

# OpenDR — Open Deep Learning Toolkit for Robotics

Project Start Date: 01.01.2020

Duration: 36 months

Lead contractor: Aristotle University of Thessaloniki

**Deliverable D3.1 First report on deep human  
centric active perception and cognition**

Date of delivery: 31 December 2020

Contributing Partners: TAU, AUTH, AU

Version: v4.0



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449.

<b>Title</b>	<b>D3.1 First report on deep human centric active perception and cognition</b>
<b>Project</b>	<b>OpenDR (ICT-10-2019-2020 RIA)</b>
<b>Nature</b>	Report
<b>Dissemination Level:</b>	<b>Public</b>
<b>Authors</b>	Anastasios Tefas (AUTH), Nikolaos Passalis (AUTH), Moncef Gabbouj (TAU), Jenni Raitoharju (TAU), Anton Muravev (TAU), Dat Thanh Tran (TAU), Negar Heidari (AU), Lukas Hedegaard Morsing (AU), Alexandros Iosifidis (AU), Nikos Nikolaidis (AUTH), Maria Tzelepi (AUTH), Paraskevi Nousi (AUTH), Pavlos Tosidis (AUTH), Kostantinos Tsampazis (AUTH), Efstratios Kakaletsis (AUTH), Charalampos Symeonidis (AUTH)
<b>Lead Beneficiary</b>	TAU (Tampere University)
<b>WP</b>	3
<b>Doc ID:</b>	OPENDR_D3.1.pdf

## Document History

<b>Version</b>	<b>Date</b>	<b>Reason of change</b>
v1.0	25/9/2020	Deliverable structure ready
v2.0	27/11/2020	Contributions from partners finalized
v3.0	18/12/2020	Final version ready
v4.0	30/12/2020	Final version to be submitted

## Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
1.1	Deep person/face/body part active detection/recognition and pose estimation (T3.1)	7
1.2	Deep person/face/body part tracking, human activity recognition (T3.2)	9
1.3	Social signal (facial expression, gesture, posture, etc.) analysis and recognition (T3.3)	11
1.4	Deep speech and biosignals analysis and recognition (T3.4)	12
1.5	Connection to Project Objectives	12
<b>2</b>	<b>Deep person/face/body part active detection/recognition and pose estimation</b>	<b>15</b>
2.1	Lightweight Face/Person Detection for Robotics	15
2.1.1	Introduction and objectives	15
2.1.2	Summary of state of the art	15
2.1.3	Description of work performed so far	16
2.1.4	Performance evaluation	17
2.2	Lightweight Face Recognition for Robotics	18
2.2.1	Introduction and objectives	18
2.2.2	Summary of state of the art	19
2.2.3	Description of work performed so far	20
2.2.4	Performance evaluation	20
2.3	Lightweight Human Pose Estimation for Robotics	21
2.3.1	Introduction and objectives	21
2.3.2	Summary of state of the art	22
2.3.3	Description of work performed so far	23
2.3.4	Performance Evaluation	23
2.4	Quadratic Mutual Information Regularization for Training Real-time Lightweight CNNs	26
2.4.1	Introduction, objectives and Summary of state of the art	26
2.4.2	Description of work performed so far	27
2.4.3	Performance evaluation	30
2.5	Fast Adaptive Inference using Early Exits	30
2.5.1	Introduction, objectives and summary of state of the art	30
2.5.2	Description of work performed so far	31
2.5.3	Performance evaluation	35
2.6	Active Control for Face Recognition	35
2.6.1	Introduction, objectives and summary of state of the art	35
2.6.2	Description of work performed so far	37
2.6.3	Performance evaluation	39
2.7	Active Face Shooting using Deep Reinforcement Learning	39
2.7.1	Introduction, objectives and summary of state of the art	39
2.7.2	Description of work performed so far	41
2.7.3	Performance Evaluation	43
2.8	Active Face Recognition using Synthesized Facial Views	43
2.8.1	Introduction, objectives, Summary of state of the art	43
2.8.2	Description of work performed so far	45

2.8.3	Performance evaluation . . . . .	47
2.9	Multilinear Compressive Learning for Face Recognition . . . . .	49
2.9.1	Introduction, Objectives and Summary of the state-of-the-arts . . . . .	49
2.9.2	Preliminaries . . . . .	50
2.9.3	Description of work performed so far . . . . .	52
2.9.4	Performance evaluation . . . . .	54
2.9.5	Future work . . . . .	57
2.10	Deepbots: A Webots-based Deep Reinforcement Learning Framework for Robotics . . . . .	58
2.10.1	Introduction, objectives and Summary of state of the art . . . . .	58
2.10.2	Description of the work . . . . .	60
<b>3</b>	<b>Deep person/face/body part tracking, human activity recognition</b>	<b>62</b>
3.1	Lightweight Human Activity Recognition . . . . .	62
3.1.1	Introduction and objectives . . . . .	62
3.1.2	Summary of state of the art . . . . .	62
3.1.3	Description of work performed so far . . . . .	63
3.1.4	Performance evaluation . . . . .	63
3.2	Temporal Attention-Augmented Graph Convolutional Network for Efficient Skeleton-Based Human Action Recognition . . . . .	65
3.2.1	Introduction and objectives . . . . .	65
3.2.2	Summary of state of the art . . . . .	65
3.2.3	Description of work performed so far . . . . .	66
3.2.4	Performance evaluation . . . . .	68
3.3	Progressive Spatio-Temporal Graph Convolutional Network for Skeleton-Based Human Action Recognition . . . . .	68
3.3.1	Introduction, objectives and summary of state of the art . . . . .	68
3.3.2	Description of work performed so far . . . . .	70
3.3.3	Performance evaluation . . . . .	71
3.4	Spatio-Temporal Bilinear Network for Skeleton-Based Human Action Recognition . . . . .	72
3.4.1	Introduction, objectives and summary of state of the art . . . . .	72
3.4.2	Description of work performed so far . . . . .	73
3.4.3	Performance evaluation . . . . .	75
3.5	Human Activity Recognition using Recurrent Bag-of-Features . . . . .	78
3.5.1	Introduction, objective and summary of state-of-the-art . . . . .	78
3.5.2	Description of work performed so far . . . . .	79
3.5.3	Performance evaluation . . . . .	80
3.6	Semi-supervised Compressive Learning with Deep Priors . . . . .	81
3.6.1	Introduction, Objectives and Summary of the state-of-the-arts . . . . .	81
3.6.2	Description of work performed so far . . . . .	82
3.6.3	Performance evaluation . . . . .	84
<b>4</b>	<b>Social signal (facial expression, gesture, posture, etc.) analysis and recognition</b>	<b>86</b>
4.1	Landmark-based facial expression recognition . . . . .	86
4.1.1	Introduction and objectives . . . . .	86
4.1.2	Summary of state of the art . . . . .	86
4.1.3	Future work . . . . .	87

4.2	Probabilistic Class-Specific Classification . . . . .	88
4.2.1	Introduction and objectives . . . . .	88
4.2.2	Summary of state of the art . . . . .	88
4.2.3	Description of work performed so far . . . . .	89
4.2.4	Performance evaluation . . . . .	90
<b>5</b>	<b>Deep speech and biosignals analysis and recognition</b>	<b>91</b>
5.1	Speech Command Recognition based on Quadratic Self-organized Operational Network . . . . .	91
5.1.1	Introduction, Summary of the state-of-the-arts and Objective . . . . .	91
5.1.2	Description of work performed so far . . . . .	92
5.1.3	Performance evaluation . . . . .	94
5.1.4	Future Work . . . . .	96
5.2	Heart Anomaly Detection using Attention-based Neural Bag-of-Features Learning	96
5.2.1	Introduction, Summary of the state-of-the-arts and Objective . . . . .	96
5.2.2	Description of work performed so far . . . . .	98
5.2.3	Performance evaluation . . . . .	100
<b>6</b>	<b>Conclusions</b>	<b>103</b>
<b>7</b>	<b>Appendix</b>	<b>121</b>
7.1	Efficient adaptive inference for deep convolutional neural networks using hierarchical early exits . . . . .	121
7.2	Efficient Adaptive Inference leveraging Bag-of-Features-based Early Exits . . .	143
7.3	Leveraging Active Perception for Improving Embedding-based Deep Face Recognition . . . . .	150
7.4	Leveraging Deep Reinforcement Learning for Active Shooting Under Open-world Setting . . . . .	157
7.5	Recurrent Bag-of-Features for Visual Information Analysis . . . . .	164
7.6	Human Action Recognition using Recurrent Bag-of-Features Pooling . . . . .	195
7.7	Temporal Attention-Augmented Graph Convolutional Network for Efficient Skeleton-Based Human Action Recognition . . . . .	210
7.8	Progressive Spatio-Temporal Graph Convolutional Network for Skeleton-Based Human Action Recognition . . . . .	219
7.9	On the spatial attention in Spatio-Temporal Graph Convolutional Networks for skeleton-based human action recognition . . . . .	225
7.10	Improving the performance of lightweight CNNs for binary classification using Quadratic Mutual Information regularization . . . . .	233
7.11	Quadratic mutual information regularization in real-time deep CNN models . .	270
7.12	Multilinear Compressive Learning . . . . .	277
7.13	Multilinear Compressive Learning with Prior Knowledge . . . . .	298
7.14	Attention-based Neural Bag-of-Feature Learning for Sequence Data . . . . .	314
7.15	Probabilistic Class-Specific Classification . . . . .	325
7.16	Deepbots: A Webots-based Deep Reinforcement Learning Framework for Robotics	341
7.17	Speech Command Recognition in Computationally Constrained Environments with a Quadratic Self-organized Operational Layer . . . . .	354

## Executive Summary

This document presents the status of the work performed for **WP3–Deep human centric active perception and cognition**. This work package contains five main tasks, of which there are four currently active. These are *Task 3.1–Deep person/face/body part active detection/recognition and pose estimation*, *Task 3.2–Deep person/face/body part tracking, human activity recognition*, *Task 3.3–Social signal (facial expression, gesture, posture, etc.) analysis and recognition*, and *Task 3.4–Deep speech and biosignals analysis and recognition*. The document starts with a general introduction, providing an overview of the individual chapters and linking them to the main objectives of the project. The introduction is followed by chapters dedicated to each of the tasks. Each chapter provides (i) an overview on the state of the art for the individual topics, (ii) details of the partners' current work as well as initial performance results (where available), and (iii) a description of the planned future steps. Finally, the conclusion section provides a closing overview of the work and the total progress of the work package.

# 1 Introduction

This document describes the work done during the first year of the project in the four active (out of five total) major research areas of WP3, namely:

- Deep person/face/body part active detection/recognition and pose estimation;
- Deep person/face/body part tracking, human activity recognition;
- Social signal (facial expression, gesture, posture, etc.) analysis and recognition;
- Deep speech and biosignals analysis and recognition.

A summary of the work done on a number of research themes within these 4 general topics is given in the next subsections. Detailed descriptions are provided in Sections 2 through 5.

## 1.1 Deep person/face/body part active detection/recognition and pose estimation (T3.1)

During the first 12 months of the project, AUTH developed a wide variety of novel methodologies and tools towards tackling the challenges that arise in deep human centric active perception and cognition and more specifically in T3.1 (Deep person/face/body part active detection/recognition and pose estimation). The work conducted by AUTH is briefly summarized below.

First, AUTH investigated the current state-of-the-art for three different application areas: a) face and person detection (Section 2.1), b) face recognition (Section 2.2) and c) human pose estimation (Section 2.3). After selecting the most prominent approaches, the selected methods were evaluated and sorted with respect to three different key aspects: a) performance on embedded hardware that is typically used for robotics, b) accuracy (or performance on widely used metrics for the task at hand) and c) stability/ease of implementation and maintenance. Extensive evaluation on different platforms have been conducted, including NVIDIA TX-2, NVIDIA AGX and other mobile and embedded platforms, while special care have been given to optimize the provided implementation and provide framework agnostic models, e.g., using ONNX<sup>1</sup> and platform-specific optimizers, such as TensorRT<sup>2</sup>. For all methods selected for inclusion in OpenDR toolkit, we have prepared optimized pre-trained models, while the developed implementations meet the performance standards, e.g., real time execution, as defined by OpenDR requirements and specifications.

Furthermore, AUTH developed a structured methodology for training and deploying regularized lightweight deep convolutional neural network models, capable of effectively operating in real-time on embedded devices, such as NVIDIA TX-2, for high-resolution video input (Section 2.4). To this end, the impact of hinge loss against the cross entropy loss on the classification performance was studied, while a novel regularization method motivated by the Quadratic Mutual Information, was also proposed in order to improve the generalization ability of the employed models. Several different DL models focusing on human-centric perception, such as CNN models for bicycle, crowd, face, and football player detection, which can be used in a wide range of robotics applications, were developed. This allowed us to perform extensive

---

<sup>1</sup><https://onnx.ai>

<sup>2</sup><https://developer.nvidia.com/tensorrt>

experiments on various binary classification problems that are often involved in autonomous robotic systems, validating the effectiveness of the proposed methodology.

An efficient early exit methodology for providing deep learning models with adaptive computational graphs that can readily adapt on-the-fly to the available resources was also proposed (Section 2.5). Note that despite their advantages, existing early exit methods suffer from many limitations which limit their performance, e.g., they ignore the information extracted from previous exit layers, they are unable to efficiently handle feature maps with large sizes, etc. To overcome these limitations we proposed a Bag-of-Features (BoF)-based method that is capable of constructing efficient hierarchical early exit layers with minimal computational overhead, while also providing an adaptive inference method that allows for early stopping the inference process when the network is confident enough for its output, leading to significant performance benefits, especially on latency critical and real-time robotics applications. To this end, the BoF model was extended and adapted to the needs of early exits by constructing additive shared histogram spaces that gradually refine the information extracted from the various layers of a network, in a hierarchical manner, while also employing a classification layer reuse strategy to further reduce the number of parameters needed per exit layer. It is worth noting that the proposed method is generic and can be readily combined with any neural network architecture, which indicates that the proposed method might have a significant impact on a wide range of tasks, that extend well over those considered in the context of this deliverable. The effectiveness of the proposed method was extensively demonstrated using several different image datasets, proving that early exits can be readily transformed into a practical tool, which can be effectively used in various real-world embedded applications for various robotics tasks.

AUTH also started investigating research topics related to active perception. More specifically, we proposed an active perception-based face recognition approach, which is capable of simultaneously extracting discriminative embeddings, as well as predicting in which direction a robot must move in order to get a more discriminative view (Section 2.6). To the best of our knowledge, this is the first embedding-based active perception method for deep face recognition, while, as it was experimentally demonstrated, the proposed method led to significant improvements, increasing the face recognition accuracy up to 9%. At the same time, the proposed method allows for using overall smaller and faster models, reducing the number of parameters by over one order of magnitude, thus meeting two important robotics requirements (i.e., using fast and lightweight models, as well as equipping models with active perception capabilities). Also, in Section 2.8 we describe a novel approach for active face recognition using synthesized facial views, which is currently under development, and allows for exploiting photorealistic facial view rendering to discover the best facial view to use for face recognition. This can lead to multiple benefits, since it can be used for both reducing the need for actually performing robot actions in order to acquire better views, as well as for providing a more effective way to perform active perception by simulating the effects of various control actions.

Furthermore, recent advances in Deep Reinforcement Learning (DRL) led to the development of powerful agents that can learn how to perform complicated tasks in an end-to-end fashion operating directly on raw unstructured data, e.g., images. However, the real world performance of such methods critically relies on the quality of the simulation environments used for training them. To enable training active perception approaches, we also developed a realistic simulation environment, by employing a state-of-the-art graphics engine, for training DRL agents to perform active shooting (Section 2.7). In contrast with previous approaches, that solely relied on simplistic constrained datasets, the environment employed in this work supports a challenging open-world setting, providing a solid step towards developing effective

active perception-enabled DL models. An appropriate reward shaping approach is also introduced in this work, ensuring that the agent will behave as expected, avoiding erratic movements, as demonstrated through the conducted experiments. Finally, we have identified a critical bottleneck in the process of developing active perception algorithms that are integrated with advanced simulation tools: a large development effort is required just to interface the learning algorithm with the corresponding simulator. To this end, we have developed a generic framework that allows for easily developing active perception methods by integrating the powerful Webots simulator (Section 2.10). The developed framework allows for simplifying the interface exposed to various active perception algorithms. We expect that this will allow to significantly speed up the development of active perception algorithms.

During the first 12 months of the project, TAU focused on tackling the problem of learning from compressed measurements for face detection/recognition. In many face recognition applications for robotics, the computational aspect of image acquisition plays an important role in the design process, especially for real-time applications. In this scenario, Compressive Sensing, which acquires images in a highly compressed form (compressed measurements), serves as a suitable signal acquisition paradigm. In order to perform inference using the compressed measurements without reconstructing the images, TAU has developed a framework to jointly learn the sensing component and the inference model, taking into account the tensor structure of images. Given the same amount of compressed measurements, the developed framework achieves state-of-the-art recognition accuracy while requiring several orders of magnitude lower computation and memory compared to the prior state-of-the-art. The framework is presented in detail in Section 2.9.

In addition, TAU also developed a surrogate performance indicator for the Multilinear Compressive Learning framework to quickly estimate the final recognition accuracy given different configurations of the sensing device. In real-life applications, given a computational budget, it often takes a huge amount of experimentation to determine the input resolution and the compression rate. The performance indicator developed by TAU is especially useful for OpenDR toolkit users when calibrating the configurations of the proposed framework given a fixed computational budget as it allows the users to speedily rank and find the optimal configuration. The details about the performance indicator developed by TAU can be found in Section 2.9.

## **1.2 Deep person/face/body part tracking, human activity recognition (T3.2)**

AU developed a variety of methods towards tackling the challenges that arise in deep learning based human activity recognition, following a human-centric approach based on body skeletons. Moreover, we conducted evaluations of efficient deep learning models for video classification in the context of human activity recognition in terms of their suitability for the OpenDR toolkit. The work conducted by AU is briefly summarized below.

Video-based human activity recognition is generally a computationally costly task, because the classification of an action requires processing of a sequence of frames. In embedded and low-power settings (such as those considered in OpenDR) the computational demand of human activity recognition using models that achieve state-of-the-art accuracy is prohibitive. These settings impose a strict computational budget, and the primary interest is in the trade-off between computational efficiency and accuracy, rather than the accuracy alone. AU benchmarked and validated the performance of four state-of-the-art video-based human activity recognition models based on 3D convolutions (Section 3.1). Our analysis includes performance evaluation under different data pre-processing strategies, classification fusion options and the efficiency

(in clips per second) that can be achieved in high-end NVIDIA GeForce RTX 2080ti GPU, as well as the NVIDIA Jetson TX2 and the NVIDIA Jetson AGX Xavier GPUs defined in the OpenDR specifications as the benchmark processing units.

Focusing on human-centric activity recognition, AU proposed multiple methods for improving the performance and the efficiency in human body skeleton-based activity recognition. The adoption of human body skeletons, which can be obtained using color image sequences or depth sequences using pose estimation methods (Section 2.3), for human activity recognition sets several challenges in methods based on deep neural networks due to the non-grid structure of skeleton data. Graph Convolutional Networks have been successfully employed to address these challenges by forming a spatio-temporal graph structure to describe the motion of the human body joints during action execution. However, the state-of-the-art skeleton-based human activity recognition methods are generally computationally complex due to the adoption of deep network architectures and the need to process a large number of human body skeletons and/or multiple-stream network topologies. To address this issue, AU proposed the Temporal Attention-Augmented Graph Convolutional Network (TA-GCN) (Section 3.2), which is able to select the most informative human body skeletons for activity recognition by applying a learnable attention mechanism. Computational cost analysis indicates that the proposed TA-GCN-based method is able to outperform with a high margin existing skeleton-based activity recognition methods, and perform on par (or even outperform) state-of-the-art methods employing the entire skeleton sequence data, while demonstrating considerable (ranging from  $\times 2.9$  to  $\times 19.3$ ) speed improvements.

One of the key factors in improving efficiency of skeleton-based human activity recognition methods is the optimization of the topology of the adopted GCN network. Most of the existing state-of-the-art methods employ a deep network topology (10 layers of spatio-temporal graph convolutional layers), while tuning the design of an optimized (in terms of both performance and efficiency) network topology requires extensive experimentation. To autonomously determine optimized network topologies for skeleton-based human activity recognition tasks, AU proposed the Progressive Spatio-Temporal Graph Convolutional Network (PST-GCN) (Section 3.3) that is able to automatically learn the network structure and train its parameters directly from data. Experimental evaluation indicates that PST-GCN is able to determine and train compact network topologies that achieve state-of-the-art performance. This leads to improvements in terms of number of parameters and FLOPs, leading to faster inference time. Moreover, we proposed the Spatio-Temporal Bilinear Network (ST-BLN) (Section 3.4) that is able to aggregate information in terms of both space (arrangement of human body joints in each skeleton) and time (motion of human body joints in successive skeletons). Experimental evaluation of the ST-BLN indicates that it can lead the same performance with state-of-the-art skeleton-based human activity recognition methods while achieving speed improvements of  $\times 2.14$ - $\times 2.78$ .

Deep Learning (DL) has provided powerful tools for visual information analysis, allowing for having models that are excelling in complex and challenging image analysis tasks by extracting meaningful feature vectors with high discriminative power. However, these powerful feature vectors are crushed through the pooling layers of the networks, that usually implement the pooling operation in a less sophisticated manner. This can lead to significant information loss, especially in cases where the informative content of the data is sequentially distributed over the spatial or temporal dimension, e.g., human activity recognition in videos, which often requires extracting fine-grained temporal information. To this end, AU proposed a novel stateful recurrent pooling approach, that can overcome the aforementioned limitations (Section 3.5). The proposed method is inspired by the well-known Bag-of-Features (BoF) model, but em-

employs a stateful trainable recurrent quantizer, instead of plain static quantization, allowing for efficiently processing sequential data and encoding both their temporal, as well as their spatial aspects. The effectiveness of the proposed Recurrent BoF model to enclose spatio-temporal information was demonstrated using several different activity recognition datasets and tasks, including a novel dataset that requires capturing fine-grained temporal information of the input video sequences.

During the first 12 months of the project, TAU focused on developing semi-supervised learning methodology for face detection/recognition from compressed measurements. In many robotic applications, the ability to learn from unlabeled data can greatly enhance the learning capacity of the robots, especially when human-provided labels are time-consuming or difficult to obtain. Although compressive learning and semi-supervised learning are complementary and useful in robotics, there exists no work that combines the two learning paradigms to have methods that can learn in both compressive and semi-supervised manner. Working towards this objective, TAU has developed a novel semi-supervised compressive learning method (Section 3.6), which is capable of leveraging unlabeled compressed data to improve the recognition accuracy.

### **1.3 Social signal (facial expression, gesture, posture, etc.) analysis and recognition (T3.3)**

AU worked towards two types of methodologies for facial expression recognition based on still images depicting facial expressions, and videos (or image sequences) depicting the facial activity leading to an expression. The work conducted by AU is briefly summarized below.

For video-based facial expression recognition, spatio-temporal changes in the expressions can be encoded by the relative position of facial landmark points in successive video frames. Such an approach exhibits several advantages compared to using color images due to its inherent invariance to face scale, illumination and head pose variations, and face appearance. However, the adoption of deep learning models operating in Euclidean structured data (like CNN and RNN models) is not able to take advantage of such information. Similar to human body skeletons used for human activity recognition (see Sections 3.2, 3.3 and 3.4), facial landmarks are non-Euclidean structured data that can be represented by graph structures, with the evolution of these graph structures through time encoding the facial activity leading to the performed facial expression. AU has developed a processing pipeline for Spatio-Temporal GCN-based facial expression recognition (Section 4.1) and is currently evaluating the performance of the corresponding state-of-the-art methods for skeleton-based action recognition in the facial expression recognition tasks, including our proposed efficient methods for spatio-temporal graph classification, i.e. TA-GCN (Section 3.2), PST-GCN (Section 3.3) and ST-BLN (Section 3.4).

AU also worked on image-based recognition applied on facial expression recognition and identification/verification problems (Section 4.2). When the objective is the discrimination of a class of interest (e.g. a facial expression) from any other possibility (other facial expressions), class-specific methodologies have shown to outperform binary classification methodologies. Class-specific approaches can be used both for identification/verification problems (i.e. to evaluate the similarity of an image to the class of interest) and classification problems (i.e. to classify the image to the class of interest or not). However, existing class-specific discriminant methods require training of an additional classifier on their outputs to tackle classification problems. Moreover, they consider all samples not belonging to the class of interest to belong to the negative class without exploiting any structural information of the negative class. Due to that

the samples forming the negative class can belong to other classes (e.g. other facial expressions in the facial expression recognition problem) it is expected that they will form subclasses. We approached the class-specific discrimination problem under a probabilistic formulation, leading to a probabilistic framework that can exploit prior information related to the class probabilities (dealing with class imbalance) and subclass information of the negative class data. We showed that the proposed probabilistic framework can incorporate existing class-specific methods as specific solutions of it. Moreover, due to that our approach is able to express the probability of the class appearance in an input image, it naturally leads to a class-specific classifier that is jointly optimized with the class-specific discrimination task. The effectiveness of the proposed Probabilistic Class-Specific Discriminant Analysis method was demonstrated using several different image-based facial expression recognition datasets both in facial expression identification/verification and recognition problems.

#### 1.4 Deep speech and biosignals analysis and recognition (T3.4)

Medical diagnosis, which plays a crucial role in ensuring human prosperity, is inherently an intricate process. This is especially true for cardiovascular disease diagnosis since the majority of heart anomaly can only be detected and diagnosed by an expert cardiologist with several years of experience and exposure. During the first 12 months, TAU has focused on developing heart anomaly detection algorithms using two types of heart signals: Electrocardiogram (ECG) and Phonocardiogram (PCG). These signals can be easily obtained by a robot, thus, suitable to be used in health-care robotic applications. The work conducted by TAU, detailed in Section 5.2, has addressed two limitations of existing works in automatic detection of heart anomaly: (i) existing end-to-end solutions rely on deep neural networks with high computational complexities, thus, unsuitable for real-time robotic applications; (ii) existing solutions are considered by the health specialists as black-box solutions, which are capable of diagnosing with a *yes* or *no* outcome without providing the *why*.

TAU has also contributed to the deep speech analysis and recognition by developing a novel neural layer, which combines the quadratic form kernels with the self-organized operational layer approach (Section 5.1). Self-organized operational neural networks generalize the neuron model and its operations by learning the appropriate transformations (via truncated Taylor series) during backpropagation. The quadratic form kernels allow the layer to take the cross-correlations within the input into account, which raises the generality and is particularly promising for audio processing. The combination of the two results in speech recognition models that have comparable or superior performance to the convolutional ones at small scales and depths, which benefits the deployment and saves computation.

#### 1.5 Connection to Project Objectives

The work performed within WP3, as summarized in the previous subsections, perfectly aligns with the project objectives. More specifically, the conducted work progressed the state-of-the-art towards meeting following objectives of the project:

- O1 *To provide a modular, open and non-proprietary toolkit for core robotic functionalities enabled by lightweight deep learning*

AUTH developed highly-optimized, lightweight and open-source implementations for face and person detection (Section 2.1), face recognition (Section 2.2), as well as human

pose estimation (Section 2.3).

TAU developed highly-optimized and open-source implementations for face recognition based on compressed measurements (Section 2.9).

AU identified and evaluated lightweight and open-source implementations for efficient video-based human activity recognition (Section 3.1).

O1a *To enhance the robotic autonomy exploiting lightweight deep learning for on-board deployment*

AU developed efficient deep learning solutions for skeleton-based human activity recognition focusing on two aspects; first the automatic selection of the most informative body skeletons in a sequence through a learnable attention mechanism leads to a reduced amount of input data that need to be processed (Section 3.2), Second, the automatic determination of a compact neural network architecture following a data-driven process (Section 3.3) leads to a reduced number of parameters needed to process the skeletons in the input sequence. In the same spirit, a network formed by bilinear spatio-temporal layers (Section 3.4) leads to a reduction of the number of parameters in the network increasing inference speed. AU also developed a processing pipeline to further exploit the benefits of the proposed methods for landmark-based facial expression recognition (Section 4.1). Moreover, a probabilistic framework for class-specific identification/verification and classification problems was developed and applied in facial expression recognition (Section 4.2).

AUTH developed an adaptive inference approach that can readily adapt the computations performed in a DL network to the available resources, meeting in the way strict speed and latency requirements that exist in many robotics applications (Section 2.5). AUTH, together with AU, also proposed a novel recurrent pooling approach that employs a stateful trainable recurrent quantizer, instead of plain static quantization, allowing for efficiently processing sequential data and encoding both their temporal, as well as their spatial aspects (Section 3.5).

TAU developed an efficient and interpretable deep learning solution for heart anomaly detection using two types of heart signals (Electrocardiogram and Phonocardiogram). Interpretability is enabled through a matrix-based attention mechanism and computational efficiency is enabled by incorporating the developed attention mechanism to the Neural Bag-of-Features models (Section 5.2). TAU also worked on enhancing the training process of multilinear compressive learning framework by developing a semi-supervised extension (Section 3.6) as well as an efficient performance indicator for this framework to rapidly estimate of learning performances of different sensor configurations (Section 2.9). TAU also contributed an efficient speech recognition model, which learns data-specific second-order operations, resulting in competitive performance with simple and compact architectures (Section 5.1).

O1b *To provide real-time deep learning tools for robotics visual perception on high-resolution data*

AUTH developed a structured methodology for training and deploying regularized lightweight deep convolutional neural network models, capable of operating in real-time on embedded devices for high-resolution video input (Section 2.4).

*O2b To provide specific deep human-centric active robot perception tools*

AUTH proposed an active perception-based face recognition approach, that can also predict in which direction a robot must move in order to get a better view of a human subject (Section 2.6). In Section 2.8 we also provide a novel approach for active face recognition using synthesized facial views which employs photorealistic facial view rendering to select the best facial view to use for face recognition. Also, a realistic simulation environment, by employing a state-of-the-art graphics engine, was developed to enable training active perception-enabled models (Section 2.7), along with an easy to use interface to the Webots simulator, allowing for more easily developing active perception algorithms (Section 2.10).

## 2 Deep person/face/body part active detection/recognition and pose estimation

### 2.1 Lightweight Face/Person Detection for Robotics

#### 2.1.1 Introduction and objectives

Face and person detection constitute crucial tasks related to human centric visual object detection. Given the proper training data, generic object detection methods can easily be applied to these tasks. However, face and person detection have been studied separately in recent literature, due in part to their importance in human centric visual analysis, and also because of the single-class nature of these tasks. The latter, in particular, allows for the development of specialized methods, which can benefit from prior knowledge surrounding the tasks, for example by using keypoints or facial and body parts to more accurately detect entire faces and bodies. Specialized face and person detection methods have also been studied to facilitate secondary analysis tasks, such as recognition and re-identification.

In robotics, face and person detection can aid in many applications, from surveillance to imposing safety related regulations. As such, accuracy and speed are very important. However, it can be challenging to find a suitable accuracy and speed trade-off, as lightweight models generally lack the discriminant ability of their heavier or specialized alternatives. Given the memory and computational constraints of embedded systems that can be attached on robots, such as the Jetson TX2, lightweight methods are preferred as larger models are typically associated with prohibitive memory and computational costs.

#### 2.1.2 Summary of state of the art

WIDER face dataset [222] is a challenging benchmark for face detection algorithms, containing faces of multiple sizes and varying difficulty. Figure 1 shows the most recent results on the WIDER dataset. The top performing algorithm on hard samples, ASFD [227], short for Automatic and Scalable Face Detector, recently proposed the use of compound scaling methods to uniformly scale the backbone net, feature module and head networks to develop a family of face detectors, motivated by the scalable model design used in EfficientDet [188], a recently proposed generic object detector.

RetinaFace [41] is a single-shot, multi-level face localisation method, using a regression target for 3D face reconstruction in a single framework which unifies face box prediction, 2D facial landmark localisation and 3D vertices regression. Their proposed face detection model achieves state-of-the-art results on WIDER face and runs at real-time on desktop GPUs (47FPS for an input size of  $640 \times 640$ ), detection faces as small as  $16 \times 16$  and as large as  $406 \times 406$ .

PyramidBox [189] and Dual Shot Face Detector (DSFD) [114] use VGG16 as the feature extraction backbone, making them slow and inefficient in terms of memory, running at 12 and less than 1 FPS on an RTX 2070 Ti GPU. PyramidBox also utilizes the Feature Pyramid Network (FPN) architecture to more accurately detect faces of multiple scales, while DSFD introduced a Feature Enhance Module, which combines the advantages of FPN and Receptive Field Blocks, first proposed in the generic object detector RFBNet [123].

Motivated by recent works in generic object detection, AInnoFace [228], MaskFace [223] and Selective Refinement Network (SRN) [233] share architectural elements with RetinaNet [63], Mask R-CNN [71] and RefineDet [232] respectively. AInnoFace utilizes the focal loss

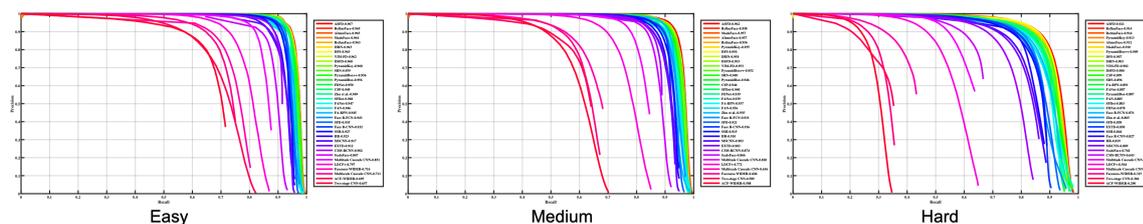


Figure 1: WIDER face test set state-of-the-art results.

introduced with RetinaNet, which weighs the training samples by their difficulty, as measured by the network’s confidence, to balance the effect of easy negative samples (i.e., background), while performing two-step classification and two-step regression, inspired by SRN, to improve the detection accuracy. MaskFace uses a ROIAlign module to produce keypoints and training for both face detection and face keypoint detection in a synergistic manner. FaceBoxes [234], combined the Region Proposal Network (RPN) [165] architecture into the Single Shot Detector framework [124], and introduced the Rapidly Digested Convolutional Layers and the Multiple Scale Convolutional Layers, designed to achieve real-time speed even on CPUs.

Taking into account the mobile applications of face detection, the Light and Fast Face Detector (LFFD) [74] utilizes a simple and efficient architecture and takes advantage of the receptive field of neurons in an anchor-free, single-stage face detector. Their proposed framework is actually based on generic one-class detection, and can be used in face detection, pedestrian detection, head detection, vehicle detection etc., while remaining very fast and able to detect objects as small as 10 pixels wide in very large resolutions like 8K.

As for person detection, existing datasets can be categorized in three main categories in terms of the point of view from which they are depicted: first-person view, aerial shots and land vehicle shots. First-person view images and annotations are present in most generic object detection datasets such as PASCAL VOC [53], MS COCO [118], and the respective task is generally tackled as part of a multiclass object detection framework, as is done in Single Shot Detector (SSD) [124] or You Only Look Once version 3 (YOLOv3) [163] for example. Datasets amassed from aerial or land vehicles typically depict humans of much smaller sizes and different perspectives (pedestrians) and are tackled separately as such. Datasets for pedestrian detection include KITTI [59], CityPersons [231] and Caltech Pedestrian [45].

In Center and Scale Prediction (CSP) [125], pedestrian detection is handled in an anchor-free fashion, as a center and scale regression task using a Deep CNN. Despite optimizations, the trade-off for the high accuracy achieved is very low inference speed. In [215], Repulsion Loss was proposed, as an alternative to Multibox Loss used in SSD and Faster R-CNN [124, 165], to better separate crowded pedestrians as well as generic objects. No other changes are required to the detectors, which run at the same speed as their Multibox counterparts during inference. Finally, LFFD [74], as a generic one-class detector, can be used in person detection and in fact provided a model for pedestrian detection on edge devices trained on Caltech Pedestrian dataset.

### 2.1.3 Description of work performed so far

The state-of-the-art methods described in the previous section are evaluated in terms of the speed/accuracy trade-off on various devices, including a desktop GPU, a CPU and two embedded devices. The performance on embedded devices is particularly important as is it crucial to

the main objectives of OpenDR.

### 2.1.4 Performance evaluation

We evaluate the performance of real-time face and person detection methods in terms of speed, by measuring the average frames per second (FPS) each detector is capable of running at, on various devices: a) an NVIDIA RTX 2070 GPU, b) Intel Core i7-9700K CPU @ 3.60GHz, c) Jetson TX2 embedded system, and finally d) Jetson AGX Xavier system.

For face detection, we measure the performance of FaceBoxes [234], MultiTask Cascaded Convolutional Networks (MTCNN) [218], a staple in face detection, as well as two versions each of the more recently proposed RetinaFace [41] and LFFD [74]. We measure the performance on large resolution images, namely at  $640 \times 480$ ,  $1280 \times 720$  and  $1920 \times 1080$  for all devices and report results in Table 1, Table 2 and Table 3 respectively. RetinaFace refers to the original model and RetinaFaceCov is a more lightweight model with a MobileNet backbone, trained to detect masked faces as well. LFFDv2 is more lightweight than LFFDv1, as it is trained to detect fewer scales of faces.

As expected, as the input resolution increases the speed drops significantly, to the point where only FaceBoxes and LFFD v2 perform at or over 25 FPS on the desktop GPU. Although FaceBoxes is the fastest on the embedded devices, its accuracy is very subpar compared to RetinaFace and LFFD, which are quite fast on images of standard resolution while being capable of detecting much smaller faces.

Table 1: Face Detection Algorithms: Speed evaluation (FPS) for input size  $640 \times 480$

Method	RTX 2070	TX2	AGX Xavier	CPU
RetinaFace [41]	47	3.2	8.5	5.6
RetinaFaceCov [41]	114	13.5	18	54
LFFD v1 [74]	92	6.6	15.5	14
LFFD v2 [74]	125	10.2	23.3	101
FaceBoxes [234]	113	19	25	73
MTCNN [218]	22.3	3.8	5.2	9.8

Table 2: Face Detection Algorithms: Speed evaluation (FPS) for input size  $1280 \times 720$

Method	RTX 2070	TX2	AGX Xavier	CPU
RetinaFace [41]	18	1.1	0.5	20
RetinaFaceCov [41]	36	4.6	2.2	1.8
LFFD v1 [74]	40	2.2	8.7	4.3
LFFD v2 [74]	54	3.5	12.5	7
FaceBoxes [234]	64.4	10	16.3	30.6
MTCNN [218]	16.6	2.2	4.1	5.5

Table 4 summarizes the speed evaluation of LFFD pedestrian on the aforementioned devices for input sizes  $640 \times 480$ ,  $1280 \times 720$  and  $1920 \times 1080$ , whereas Figure 2 compares LFFD pedestrian against generic object detectors SSD, YOLOv3 and CenterNet [163, 124, 50], trained

Table 3: Face Detection: Speed evaluation (FPS) for input size  $1920 \times 1080$ 

Method	RTX 2070	TX2	AGX Xavier	CPU
RetinaFace [41]	7.5	0.5	1.6	0.8
RetinaFaceCov [41]	19	2.2	5	9
LFFD v1 [74]	18.4	1	4.1	1.6
LFFD v2 [74]	25	1.7	6	3
FaceBoxes [234]	50	7.1	9.4	14.2
MTCN [218]	11.4	1.2	3.4	3

on datasets depicting people, on Jetson AGX and Jetson TX2 for input sizes  $512 \times 512$ ,  $1024 \times 512$  and  $1024 \times 1024$ . LFFD is slightly slower than SSD with MobileNet backbone [80], but has the potential to detect smaller objects (as small as  $10 \times 10$  in VGA resolution).

Table 4: Person Detection: Speed evaluation (FPS) for various input sizes

Input Size	Method	RTX 2070	TX2	AGX Xavier	CPU
$640 \times 480$	LFFD pedestrian [74]	92	6.6	15.5	14
$1280 \times 720$	LFFD pedestrian [74]	40	2.2	8.7	4.3
$1920 \times 1080$	LFFD pedestrian [74]	18.4	1	4.1	1.6

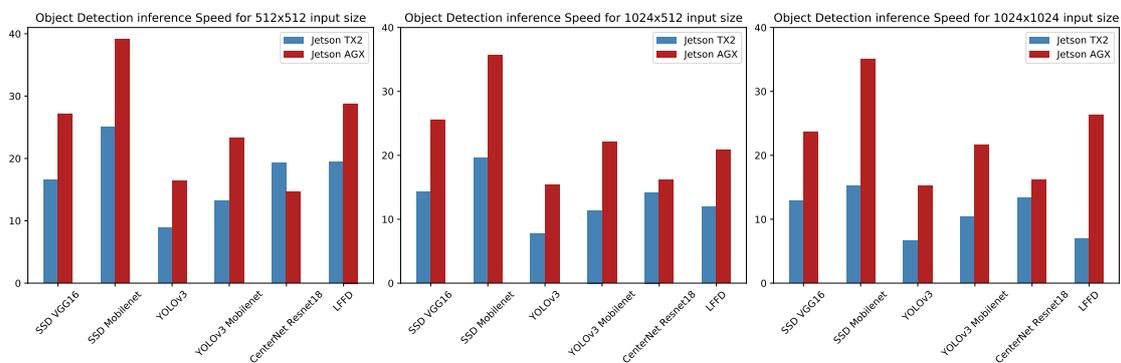


Figure 2: Person detection method speed comparison on Jetson TX2 and AGX.

## 2.2 Lightweight Face Recognition for Robotics

### 2.2.1 Introduction and objectives

Face recognition refers to analyzing images or videos where humans are depicted into order to extract their identities. Face recognition algorithms pick out specific and distinctive features of a face and either compare those to a database of known faces or directly predict the identity of a person (given that they have been appropriately trained for recognizing a set of known identities). This task is useful for many robotics applications involving human-robot interaction, such as healthcare applications (e.g., a robot should recognize the patient and provide the corresponding medicine), security applications and others. In this section, we focus on feature

learning techniques that use Deep Convolutional Neural Networks (DCNNs) and appropriate loss functions that enhance the discriminative power of the learned features [186] [171] [146]. State-of-the-art face recognition approaches work by comparing the features of a face in question, given by a feature extraction DCNN, to a database containing previously extracted features of faces of interest.

The objective of the work conducted in WP3 is to provide efficient and lightweight face recognition methods, while keeping a high accuracy, since embedded systems have limited computational resources and real-time inference is needed on many applications. To achieve this we evaluated numerous state-of-the-art algorithms, different feature extraction DCNN architectures and many loss functions that can be included as baselines tools in the OpenDR toolkit in order to meet the aforementioned requirements.

### 2.2.2 Summary of state of the art

Three modules are needed for a complete face recognition system. First, a face detector is used to localize faces in images or videos. Second, faces are aligned with a facial landmark detector. Last comes the face recognition module. The face recognition module performs a) feature extraction and b) face identification. The feature extractor is trained using an appropriate loss function and then used to extract features of faces. These features are then fed to a matching algorithm to determine the identity of the face, by computing similarity scores. Recent research on face recognition focuses both on designing effective loss functions, as well as on developing efficient architectures.

Early methods used Euclidean-distance-based loss functions to embed images into a metric space. Such commonly used loss functions are the contrastive loss and triplet loss [79]. Contrastive loss uses face image pairs, both of the same ID and of different IDs and pulls together positive pairs while pushing negative pairs apart. Triplet loss on the other hand, uses face image triplets (an anchor image, a sample of the same ID and a sample of a different ID), and minimizes the distance between the anchor and the same ID sample, while maximizing the distance between the anchor and the different ID sample. Among the drawbacks of triplet loss-based methods is their high complexity and the need to use complicated hard sample mining methods to ensure an adequate performance. Euclidean distance-based losses are largely superseded by angular/cosine-margin-based losses, such as L-Softmax [127], A-Softmax [126], CosFace [214], ArcFace [42] and AMS-Softmax [213], that aim to maximize inter-class variance and minimize intra-class variance in the resulting representation space.

The most commonly used architectures in recent face recognition systems include Residual Neural Networks (ResNets) [72], Squeeze-and-Excitation Networks (SE-Nets) [81], and Inverted Residual Networks (IR-ResNets) [17]. These are deep architectures with hundreds of layers and millions of parameters that achieve state-of-the-art accuracy on many challenging evaluation datasets. However, at the same time, these architectures are complicated and require a significant amount of resources during inference, rendering them impractical when used on mobile or embedded devices. To address this issue lighter architectures, such as MobileFaceNet [29], were proposed. To this end, in this Section we implemented and evaluated several state-of-the-art methods for face recognition in order to determine the appropriateness for inclusion in OpenDR, such as ArcFace [42] and CosFace [214], which were combined with various deep architectures like ResNets, MobileFaceNet, etc. This allows for selecting the most appropriate model that achieves the desired accuracy, while keeping the overall complexity within the required limits to ensure adequate performance on embedded and mobile devices.

Finally, it is worth noting that several datasets are publicly available for training and evaluating face recognition methods. Among the most widely used and challenging ones are the a) Microsoft Celeb dataset [67], b) Labeled Faces in the Wild dataset [83], c) VGGFace2 dataset [25], d) AgeDB30 [139] and e) CFP\_ff [168]. Most of the methods reported in the literature provide evaluation results on one (or more) of these datasets, providing a common reference point for comparing different methods.

### 2.2.3 Description of work performed so far

The state-of-the-art methods that were implemented according to OpenDR's specifications, allowing them to be later easily integrated in the OpenDR toolkit, are:

- ArcFace [42]
- CosFace [214]
- SphereFace [126]
- AM-Softmax [213]
- Residual CNN's of depth 50, 101, 152 [72]
- An improved version of the vanilla ResNet (IR) CNN's of depth 50, 101, 152 [72]
- IR CNN's with squeeze and excitation blocks of depth 50, 101, 152 [81]
- MobileFaceNet [29]

We also implemented a classification head that can be combined with the aforementioned feature extraction networks, allowing for effectively using each of these methods in a classification-based setting. Accompanying the implementation, we provide an array of demos that showcase the usage of OpenDR's face recognition methods in various use cases. These include a training and evaluation demo, a demo showcasing live inference using a webcam, a benchmarking demo to measure the inference speed and two demos utilizing Webots robot simulation and a Tiago robot equipped with a camera in order to showcase a more realistic use of the toolkit in robotics applications, as seen in Figure 3. These demos will be part of the first release of OpenDR toolkit.

### 2.2.4 Performance evaluation

Our evaluation focuses on two different models that will be provided by OpenDR toolkit, a small baseline IR(an improved version of the vanilla ResNet) [72] architecture, as well as a faster and more efficient MobileFaceNet architecture than the baseline IR [29], both trained using ArcFace loss on the cleaned MS-Celeb-1M dataset. These architectures were evaluated on four face recognition datasets that are commonly used in the literature, i.e., a) Labeled Faces in the Wild dataset [83], b) VGGFace2 dataset [25], c) AgeDB30 [139] and d) CFP\_ff [168]. We measured the accuracy of these models by comparing pairs of face images that are either of the same ID or not. Furthermore, we also measure the average FPS that the models achieve during forward pass a face image on various systems, including an Intel i5-10700k CPU, an Nvidia RTX 2070 Super GPU, a Nvidia Jetson TX2 and an Nvidia Jetson AGX Xavier. We report average inference FPS calculated using 1000 sample images.

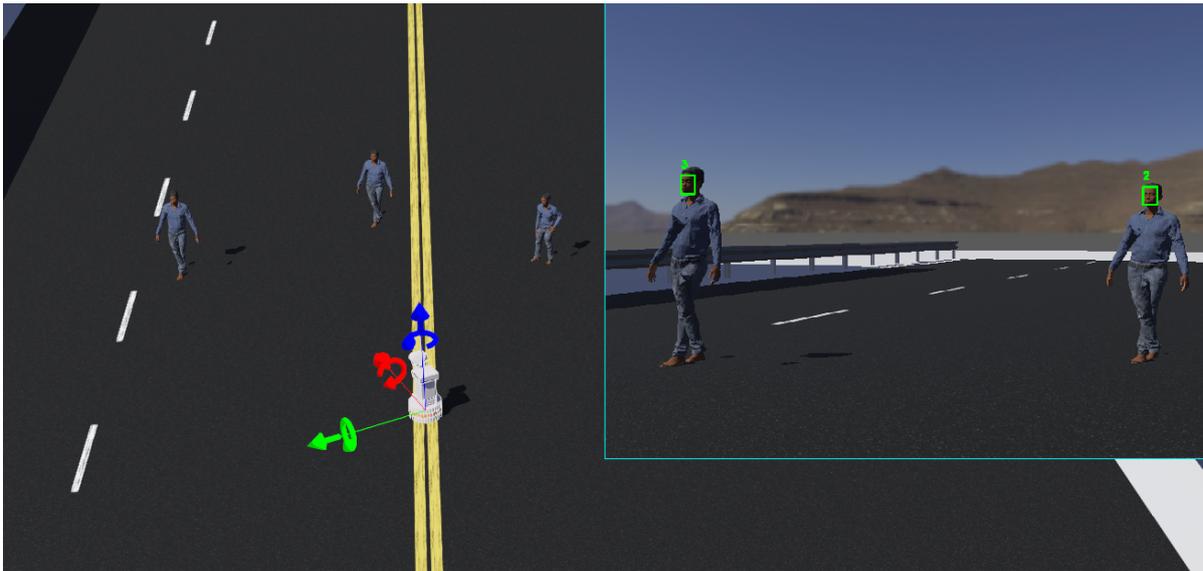


Figure 3: Demonstration of TIAGo robot identifying human models in a Webots environment.

The evaluation results are provided in Table 5 and Table 6. As shown in these tables, the MobileFaceNet architecture that will be provided in OpenDR toolkit has a slightly lower accuracy on all four datasets, compared to IR-50 architecture, but the inference speed is greatly increased. Especially on embedded systems we note that the lighter and more efficient MobileFaceNet is a better choice, when real time applications are in hand.

Table 5: Accuracy comparison between different backbones used in face recognition

Backbone	LFW	CFP_FF	VGGFace2_FP	AgeDB
IR - 50	99.84%	99.67%	95.3%	97.73%
MobileFaceNet	97.83%	97.18%	92.8%	87.03%

Table 6: Face Recognition: Speed evaluation (FPS) on various systems

Backbone	CPU	RTX2070 Super	Jetson TX2	Jetson AGX Xavier
IR - 50	18	148	18	23
MobileFaceNet	68	214	31	45

## 2.3 Lightweight Human Pose Estimation for Robotics

### 2.3.1 Introduction and objectives

Human pose estimation refers to analyzing images or videos of humans to predict their pose, which in essence entails the localization of their joints, allowing for inferring skeleton information. Pose estimation is a key step for enabling machines to understand peoples' actions and intentions in various robotics applications [239]. We focused on two-dimensional pose estimation algorithms, i.e., algorithms that predict 2D keypoints on a given RGB image, since this

consists the most popular and widely used category of DL-based pose estimation algorithms, which is expected to maximize the impact of the provided tools. These algorithms work by predicting keypoints for the joints of interest, which are then, appropriately combined to compile the final skeleton for each human that appears in an image.

Pose estimation can be single and multi-person, referring to whether it focuses on predicting poses in images containing a single person or an undefined number of multiple people. This is a basic difference because there are two approaches in pose estimation that are referred to as top-down and bottom-up. Top-down methods employ a person detector to provide them with input images that are cropped around a single person and then run pose estimation. This means that on images containing multiple persons, the pose estimation algorithm will run for each person detected separately, increasing the time required to run analogously. On bottom-up methods, the algorithm is provided with the whole image and detects all keypoints at once, not requiring multiple passes and thus its runtime is largely unaffected by the number of persons. On the other hand, the resolution per person on top-down methods is much higher and so generally get better results than bottom-up methods that often receive each person at a comparatively lower resolution.

The objective of the work conducted in WP3 is to provide efficient and lightweight human pose estimation methods, along with optimized implementations that will be released as part of the OpenDR toolkit. To this end, we evaluated various state-of-the-art algorithms to select the most appropriate ones for possible inclusion in the toolkit, as baseline tools, mainly focusing on examining the trade-offs between their accuracy and speed on mobile platforms, such as NVIDIA Jetson TX-2. Furthermore, we ensured that the selected algorithms are fully compatible with OpenDR's specifications, as detailed in D2.1, ensuring their smooth integration of the toolkit.

### 2.3.2 Summary of state of the art

Several datasets are publicly available for training and evaluating human pose estimation methods. Among the most widely used and challenging ones are a) the MPII Human Pose dataset [9] and b) the COCO dataset [119]. Most of the methods reported in the literature provide evaluation results on one (or both) of these datasets, providing a common reference point for comparing different methods.

To the best of our knowledge, at the time of writing, the latest and most accurate method on the MPII Human Pose dataset [9] for single pose estimation is the soft-gated method presented in [21], which introduces gated skip connections with per-channel learnable parameters to control the data flow for each channel. This method also employs a complex hybrid network that combines the HourGlass and U-Net architectures to further improve the accuracy of pose estimation. For the COCO dataset, both for single and multi-person pose estimation, the most accurate method currently is the Distribution-Aware coordinate Representation of Keypoint (DARK) method [229], which serves as a plugin for other pose estimation methods, improving the results by modifying the coordinate representation of the heatmap and also by changing the way the ground-truth coordinates are converted to heatmaps. Despite the ability of each of these methods to achieve state-of-the-art results on these two challenging datasets and tasks, all of them employ computationally heavy models that are difficult to run in real-time in embedded systems without extensive modification that will reduce their accuracy.

On the other hand, OpenPose method [26] uses a non-parametric representation, which is referred to as Part Affinity Fields, that associates body parts with individuals in the image in a

bottom-up approach. Furthermore, a more recent version of OpenPose, which employs a more lightweight backbone architecture [143], seems to be able to provide adequate performance, fulfilling the requirements of OpenDR toolkit, with negligible accuracy drop. Furthermore, it is worth noting that the OpenPose methodology provides multi-person pose estimation, while having real-time performance regardless of the number of individuals in the image, in contrast with other methods that suffer performance drops depending on the number of persons. This was deemed as an important aspect that led us selecting OpenPose as the basis for the main implementation that is provided in OpenDR, since this ensures that the algorithm will have constant inference time, regardless of the number of persons that appear in an image, while also more easily meeting the requirements of real-time applications.

Another bottom-up multi-person pose estimation method is SimplePose [113], which improves upon existing methods through: (i) a different representation of body parts to connect keypoints, (ii) an improved architecture of stacked hourglass network with attention mechanisms, (iii) a novel focal L2 loss dedicated to hard keypoint and keypoint association (body part) mining and (iv) a greedy keypoint assignment algorithm to get individual poses through grouping detected keypoints. SimplePose outperforms previous methods on the COCO dataset and the authors have publicly released the code. SimplePose is part of the GluonCV library for pose estimation, based on Apache's MXNet framework for training and deploying neural networks. GluonCV offers a variety of pretrained models, so a user can choose between more lightweight or more accurate architectures, depending on their use-case, with lightweight architectures being based on models that run in real-time on mobile platforms.

### 2.3.3 Description of work performed so far

The implementations of the lightweight OpenPose and SimplePose algorithms provided in OpenDR toolkit follow the abstract class structure defined in the specifications of the toolkit, i.e., we provided the appropriate implementations for the basic methods used for training new models, evaluating them and running inference on new samples, to ensure smooth integration.

Accompanying the implementation, we also developed an array of demos that showcase the usage of the developed algorithms in various use cases. Those include a webcam benchmarking demo to test the inference speed on a system with a webcam, a webcam demo that shows the video feed with the pose estimation results in a window for live testing and a training and evaluation demo. Three demos utilizing the Webots robot simulator and the TIAGo robot equipped with a camera are also provided, to showcase more realistic robotics applications of the algorithm. The first Webots demo includes the TIAGo robot and a 3D human model in an enclosed arena, with the ability to drive TIAGo using the keyboard to move around the arena while performing pose estimation through the robot's camera. The second demo draws images from a provided webcam and performs pose estimation, whose results are used to move TIAGo's arms and head around to mimic the pose as seen in Figure 4. The last demo, again performs pose estimation on webcam images and uses the angle of the elbows to drive TIAGo's motors to move it around the arena. All these demos will be part of the first version of the OpenDR toolkit.

### 2.3.4 Performance Evaluation

**Lightweight OpenPose Performance** We performed benchmarks for the pretrained model using the Mobilenet backbone provided by the authors on four different platforms: a) Nvidia RTX 2070 Super GPU, b) Intel Core i7-9700K CPU @ 3.60GHz, c) Jetson TX2 embedded

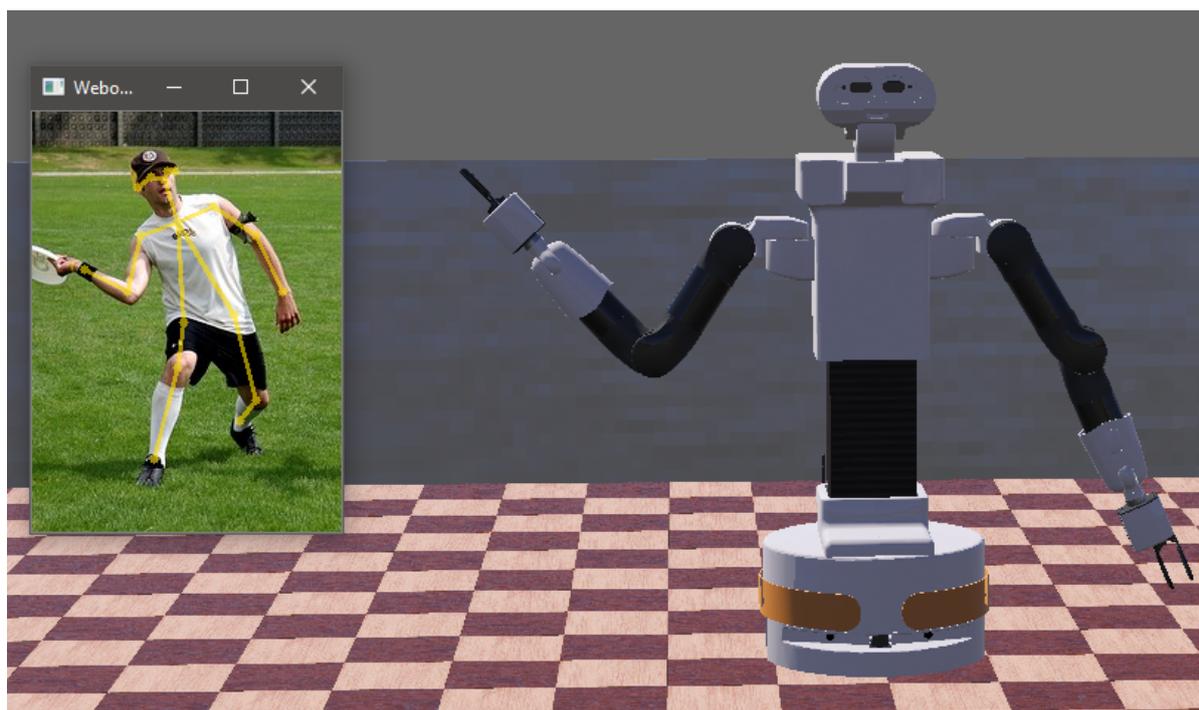


Figure 4: Demonstration of TIAGo robot mimicking arm and head pose of provided example image taken from the COCO dataset in the Webots robot simulator. Regular pose mimicking demo operates on a webcam.

system, d) Jetson AGX Xavier system. The results can be seen in detail on Table 7. The Average Precision (AP) reported is scored on a small subset of the COCO validation set using a single scale and is provided as a comparison between the different setups of the model. The model can run using no additional refinement stages as denoted by the number 0 on the table, getting the best performance in terms of FPS, but worse AP score. As more refinement stages are added, the FPS drops and the AP rises, providing a clear trade-off between speed and accuracy of the model. Moreover, the model can be optimized using ONNX providing a boost on the CPU FPS.

Prior to performing the forward pass on the model, the implementation resizes the images based on a base height parameter defined by the user, with the default value being 256. Thus, all input images have this base height defined, but with varying widths depending on their aspect ratio. As seen on Table 8, decreasing the base height increases the FPS achieved, but decreases accuracy, while increasing the base height decreases the FPS and increases accuracy.

**GluonCV SimplePose Performance** We performed additional FPS benchmarks for the GluonCV pose estimation implementation based on Apache MXNet. The algorithm used in GluonCV is SimplePose [113]. SimplePose is normally a bottom-up approach same as OpenPose, but in GluonCV it is implemented similarly to a top-down approach, employing a person detector and then passing all the bounding boxes as a batch into the pose estimator. This means that the more detections there are, the higher the processing time that is needed. The benchmarks performed are on the same hardware as before, but this time we used three different resolutions of three images that depict various numbers of people to detect; one, four and eight. This was done, to see how much performance is affected by the amount of detections. The results can be seen in Table 9. Performance is generally better with lower resolution and lower number of de-

Table 7: Comparison of FPS and AP between the provided pretrained model with Mobilenet backbone with several setups, on different platforms. FPS measurements shown in the three middle columns were made on an Intel i7-9700k CPU, an Nvidia RTX 2070 Super, a Nvidia Jetson AGX Xavier (AGX) and a Nvidia Jetson TX2. The AP reported is over a subset of the COCO validation set, using a single scale and the default image base height of 256.

<b>Model</b>	<b>Ref. Stages</b>	<b>CPU</b>	<b>CPU &amp; ONNX</b>	<b>GPU</b>	<b>AGX</b>	<b>TX2</b>	<b>AP0.5</b>
MobileNet	2	1.17	1.46	51.89	9.81	6.24	0.541
MobileNet	1	1.28	1.57	58.84	11.18	7.05	0.531
MobileNet	0	1.29	1.64	65.96	12.5	8.2	0.479

Table 8: Pose Estimation: Speed evaluation (FPS) for various base height sizes and no additional refinement stages.

<b>Base Height</b>	<b>CPU</b>	<b>CPU &amp; ONNX</b>	<b>GPU</b>	<b>AGX</b>	<b>TX2</b>	<b>AP0.5</b>
512	0.33	0.45	22.0	5.8	2.1	0.53
256	1.3	1.6	66.0	12.5	8.2	0.48
128	4.6	5.3	106.6	18.9	15.9	0.2

tections, depending on the hardware, with mobile platforms having small FPS variance between resolutions and detections, while CPU and GPU hardware show larger differences. GluonCV offers a wide variety of pretrained models both for detection and pose estimation. The benchmarks reported in Table 9 were made with Single Shot Multibox Object Detection (SSD) [124] detector with a Mobilenet [80] base feature extractor network and a SimplePose [113] pose estimator with a ResNet backbone [72].

Table 9: Pose Estimation with GluonCV and SimplePose: Speed evaluation (FPS) for various input sizes. Three FPS measurements for different number of detections: 1/4/8.

<b>Input Size</b>	<b>CPU</b>	<b>GPU</b>	<b>AGX</b>	<b>TX2</b>
1920 × 1080	25.2/19.2/14.2	30.1/29.2/24.4	7.3/6.3/6.2	3.8/3.3/3.0
1280 × 720	31.4/22.6/14.6	42.2/38.8/29.4	7.4/6.4/6.0	4.0/3.5/3.1
640 × 480	31.1/22.6/16.0	48.2/43.4/33.8	7.4/6.6/6.2	4.2/3.6/3.3

## 2.4 Quadratic Mutual Information Regularization for Training Real-time Lightweight CNNs

### 2.4.1 Introduction, objectives and Summary of state of the art

Deep Learning (DL) algorithms [66] have been established among the most effective research directions in a wide range of computer vision tasks [65]. However, state-of-the-art DL models usually owe their exceptional performance to their depth and complexity, obstructing their application in robotics. Thus, over the recent few years, the research interest has also been directed towards developing lightweight models capable of operating on devices with restricted computational resources such as embedded systems [84]. In this work, we utilize real-time deep Convolutional Neural Networks (CNN) models [204] for addressing various binary classification problems involved in autonomous systems. Our goal is to provide semantic heatmaps, by predicting for each location within the captured scene the considered object's presence. The utilized models are capable of effectively operating on devices with limited computation resources for high-resolution input (i.e. 1080p), which is of utmost importance in such applications. For example, considering the case of object detection on drone-captured images, the objects may be of extremely small size and thus image resizing, so as to meet the real-time requirement would further shrink the object rendering its detection even infeasible.

Thus, since we deal with lightweight models which usually exhibit inferior performance as compared to the more complex models, we aim at enhancing their performance. To this aim, one objective of this work, is to extensively examine the impact of hinge loss against the cross entropy loss on the classification performance of binary classification problems. Surveying the relevant literature, we observe that the cross entropy loss is widely used in deep CNNs for dealing with multi-class classification problems [72]. On the other hand, in [190] the author first proposes to replace the softmax loss layer (i.e. cross entropy loss), with a linear SVM layer. Furthermore, in [92] the authors provide comparisons among various classification losses, including the cross entropy and hinge losses, for multi-class classification problems. However, there is no other previous work extensively investigating the impact of hinge loss against cross entropy loss on the classification accuracy, with special emphasis to binary classification problems

Furthermore, another objective of this work is to propose a novel regularization method in order to improve the generalization ability of the utilized models. Generally, this constitutes a major issue in DL algorithms, since neural networks are prone to over-fitting due to their high capacity. During the past years, several regularization schemes have been proposed to achieve this goal, e.g. common regularization methods, like  $L1/L2$  regularization, and Dropout [184]. Besides, multitask-learning [28] has been proposed as a way to improve the generalization ability of a model. To this end, in this work we propose the so-called *Mutual Information (MI) regularizer*. The proposed regularizer is inspired by the Quadratic Mutual Information (QMI) measure [193], which is a variant of the Mutual Information, an information-theoretic measure of dependence between random variables. That is, apart from the classification loss, we propose to attach an additional optimization criterion based on the QMI. More specifically, our goal is to maximize the QMI, expressed using the so-called *Information Potentials* between the data samples and the corresponding classes.

A summary of this work is provided hereafter. The corresponding publications are listed below, and can be found in Appendices 7.10 and 7.11:

- [205] M. Tzelepi, and A. Tefas, “*Improving the performance of lightweight CNNs for bi-*

nary classification using Quadratic Mutual Information regularization”, Pattern Recognition, 2020.

- [206] M. Tzelepi, and A. Tefas, “Quadratic mutual information regularization in real-time deep CNN models”, IEEE International Workshop on Machine Learning for Signal Processing, 2020.

## 2.4.2 Description of work performed so far

Cross entropy loss and hinge loss functions are probably the most widely used loss functions in pattern classification. Support Vector Machines (SVM), [209], which use the hinge loss, constitute up to the present time a vivid research field [207]. SVMs, which are inherently binary classifiers, seek for the optimal hyperplane which distinctly classifies the data samples, that is the hyperplane which maximizes the margin between the two classes.

For a set of  $N$  input images  $\mathcal{X} = \{\mathbf{X}_i, i = 1, \dots, N\}$  we consider the corresponding scores with respect to each class,  $\mathbf{y}_i^{last} \in \mathfrak{R}^{N_c \times 1}$ . In the typical case the classification layer is implemented using a fully connected layer - with number of nodes equal to the number of classes - and the output is fed to the loss layer. In our case, since the objective is to develop lightweight models, and hence we propose fully convolutional architectures, instead of a fully connected layer, there is a convolutional layer with number of channels equal to the number of classes and kernel with receptive field equal to the whole input volume.

Then, the hinge loss is defined as:

$$L_{hinge} = \frac{1}{N} \sum_{i=1}^N \sum_{j=1}^{N_c} \max\left(0, 1 - \delta\{c_i = j\} y_{i,j}^{last}\right), \quad (1)$$

where  $c_i \in [1, \dots, N_c]$  indicates the correct class among the  $N_c$  classes,  $y_{i,j}^{last}$  indicates the score with respect to the  $j$ -th class for the  $i$ -th image, and

$$\delta\{condition\} = \begin{cases} 1 & , \text{if condition} \\ -1 & , \text{otherwise} \end{cases}$$

In this work, we deal with binary classification problems. That is,  $N_c = 2$ .

Cross entropy loss or softmax classifier, is extensively used in deep learning architectures [72], providing an intuitive output of normalized class probabilities. Instead of computing scores for each class, like the SVM classifier, the softmax classifier computes the scores for each class, and then applies the softmax function to transform them to a vector of values between zero and one that sum to one, in order to be interpreted as class probabilities. Finally, the classification process is realized using the cross entropy loss function.

The cross entropy loss is defined as:

$$L_{cross\_entropy} = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^{N_c} l_{i,j} \log(p_{i,j}), \quad (2)$$

where  $N_c$  is the number of classes,  $l_{i,j} \in \{0, 1\}$  is a binary indicator that takes the value 1 if the sample  $i$  belongs to class  $j$ , and  $p_{i,j}$  is the predicted softmax probability the sample  $i$  to belong to class  $j$ . For a two-class classification problem the cross entropy loss can be calculated as:

$$L_{cross\_entropy\_binary} = -\frac{1}{N} \sum_{i=1}^N \left( l_i \log(p_i) + (1 - l_i) \log(1 - p_i) \right) \quad (3)$$

In this work, we propose a novel regularizer motivated by the Quadratic Mutual Information [193]. Apart from the classification loss, we propose a regularization loss derived from the so-called information potentials of the QMI. Thus, in this section, we first introduce the Mutual Information and its quadratic variant, and then we present the proposed MI regularizer.

We assume a random variable  $Y$  representing the image representations of the feature space generated by a specific deep neural layer. We also assume a discrete-value variable  $C$  that represents the class labels. For each feature representation  $\mathbf{y}$  there is a class label  $c$ . The MI measures dependence between random variables that is, the MI measures how much the uncertainty for the class label  $c$  is reduced by observing the feature vector  $\mathbf{y}$ . Let  $p(c)$  be the probability of observing the class label  $c$ , and  $p(\mathbf{y}, c)$  the probability density function of the corresponding joint distribution.

The MI between the two random variables is defined as:

$$MI(Y, C) = \sum_c \int_{\mathbf{y}} p(\mathbf{y}, c) \log \frac{p(\mathbf{y}, c)}{p(\mathbf{y})P(c)} d\mathbf{y}, \quad (4)$$

where  $P(c) = \int_{\mathbf{y}} p(\mathbf{y}, c) d\mathbf{y}$ . MI can also be interpreted as a Kullback-Leibler divergence between the joint probability density  $p(\mathbf{y}, c)$  and the product of marginal probabilities  $p(\mathbf{y})$  and  $P(c)$ .

QMI is derived by replacing the Kullback-Leibler divergence by the quadratic divergence measure [193]. That is:

$$QMI(Y, C) = \sum_c \int_{\mathbf{y}} (p(\mathbf{y}, c) - p(\mathbf{y})P(c))^2 d\mathbf{y}. \quad (5)$$

And thus, by expanding eq. (5) we arrive at the following equation:

$$QMI(Y, C) = \sum_c \int_{\mathbf{y}} p(\mathbf{y}, c)^2 d\mathbf{y} + \sum_c \int_{\mathbf{y}} p(\mathbf{y})^2 P(c)^2 d\mathbf{y} - 2 \sum_c \int_{\mathbf{y}} p(\mathbf{y}, c) p(\mathbf{y}) P(c) d\mathbf{y}. \quad (6)$$

The quantities appearing in eq. (6), are called *information potentials* and they are defined as follows:  $V_{IN} = \sum_c \int_{\mathbf{y}} p(\mathbf{y}, c)^2 d\mathbf{y}$ ,  $V_{ALL} = \sum_c \int_{\mathbf{y}} p(\mathbf{y})^2 P(c)^2 d\mathbf{y}$ ,  $V_{BTW} = \sum_c \int_{\mathbf{y}} p(\mathbf{y}, c) p(\mathbf{y}) P(c) d\mathbf{y}$ , and thus, the quadratic mutual information between the data samples and the corresponding class labels can be expressed as follows in terms of the information potentials:

$$QMI = V_{IN} + V_{ALL} - 2V_{BTW}. \quad (7)$$

If we assume that there are  $N_c$  different classes, each of them consisting of  $J_p$  samples, the class prior probability for the  $c_p$  class is given as:  $P(c_p) = \frac{J_p}{N}$ , where  $N$  corresponds to the total number of samples. Kernel Density Estimation [172] can be used to estimate the joint density probability:  $p(\mathbf{y}, c_p) = \frac{1}{N} \sum_{j=1}^{J_p} K(\mathbf{y}, \mathbf{y}_{pj}; \sigma^2)$ , for a symmetric kernel  $K$ , with width  $\sigma$ , where we use the notation  $\mathbf{y}_{pj}$  to refer to the  $j$ -th sample of the  $p$ -th class, as well as the probability density of  $Y$  as  $p(\mathbf{y}) = \sum_{p=1}^{J_p} p(\mathbf{y}, c_p) = \frac{1}{N} \sum_{j=1}^N K(\mathbf{y}, \mathbf{y}_j; \sigma^2)$ .

Thus, eq. (7) is formulated as follows:

$$V_{IN} = \frac{1}{N^2} \sum_{p=1}^{N_c} \sum_{k=1}^{J_p} \sum_{l=1}^{J_p} K(\mathbf{y}_{pk}, \mathbf{y}_{pl}; 2\sigma^2), \quad (8)$$

$$V_{ALL} = \frac{1}{N^2} \left( \sum_{p=1}^{N_c} \left( \frac{J_p}{N} \right)^2 \right) \sum_{k=1}^N \sum_{l=1}^N K(\mathbf{y}_k, \mathbf{y}_l; 2\sigma^2), \quad (9)$$

$$V_{BTW} = \frac{1}{N^2} \sum_{p=1}^{N_c} \frac{J_p}{N} \sum_{j=1}^{J_p} \sum_{k=1}^N K(\mathbf{y}_{pj}, \mathbf{y}_k; 2\sigma^2). \quad (10)$$

The kernel function  $K(\mathbf{y}_i, \mathbf{y}_j; \sigma^2)$  expresses the similarity between two samples  $i$  and  $j$ . There are several choices for the kernel function [172]. In our experiments, we use as kernel metric a Euclidean based similarity, which also absolves us from defining the width of the kernel:  $K_{ED} = \frac{1}{1 + \|\mathbf{y}_i - \mathbf{y}_j\|_2^2}$ .

The pairwise interactions described above between the samples can be interpreted as follows:  $V_{IN}$  expresses the interactions between pairs of samples inside each class.  $V_{ALL}$  expresses the interactions between all pairs of samples, regardless of the class membership, while  $V_{BTW}$  expresses the interactions between samples of each class against all other samples.

Thus, motivated by the QMI, in this work we propose a novel regularizer in order to enhance the generalization ability of a deep model. That is, apart from the optimization criterion defined by the hinge loss function which aims at separating the samples belonging to different classes, we propose an additional optimization criterion utilizing the information potential defined in eq. (7). We assume that the hinge loss preserves the  $V_{BTW}$  information potential which aims to separate samples belonging to different classes. Then, our objective is to maximize pairwise interactions between the samples described by the  $V_{IN} + V_{ALL}$  quantities. The derived joint optimization criterion defines an additional loss function, which is attached to the penultimate convolutional layer (that is the last convolutional layer, before the one utilized for the classification task) and acts as regularizer to the main classification objective.

$$L_{MI} = -(V_{IN} + V_{ALL}), \quad (11)$$

where:

$$V_{IN} = \frac{1}{N^2} \sum_{p=1}^{N_c} \sum_{k=1}^{J_p} \sum_{l=1}^{J_p} K_{ED}(\mathbf{y}_{pk}, \mathbf{y}_{pl}), \quad (12)$$

and

$$V_{ALL} = \frac{1}{N^2} \left( \sum_{p=1}^{N_c} \left( \frac{J_p}{N} \right)^2 \right) \sum_{k=1}^N \sum_{l=1}^N K_{ED}(\mathbf{y}_k, \mathbf{y}_l). \quad (13)$$

Considering binary classification problems the above optimization criteria can be formulated as follows:

$$V_{IN} = \frac{1}{N^2} \sum_{k=1}^{J_1} \sum_{l=1}^{J_1} K_{ED}(\mathbf{y}_{1k}, \mathbf{y}_{1l}) + \frac{1}{N^2} \sum_{k=1}^{J_2} \sum_{l=1}^{J_2} K_{ED}(\mathbf{y}_{2k}, \mathbf{y}_{2l}), \quad (14)$$

and

$$V_{ALL} = \frac{1}{N^2} \left( \frac{J_1^2 + J_2^2}{N^2} \right) \sum_{k=1}^N \sum_{l=1}^N K_{ED}(\mathbf{y}_k, \mathbf{y}_l), \quad (15)$$

The total loss for the network training is defined as:

$$L_{total} = L_{Hinge} + \eta L_{MI}, \quad (16)$$

where the parameter  $\eta \in [0, 1]$  controls the relative importance of  $L_{MI}$ . We solve the above optimization problem using gradient descent. We should note that the proposed regularizer can be applied for the whole dataset, as well as in terms of mini-batch training. In our experiments

Training Approach	VGG-720p	VGG-1080p
Only Cross Entropy Loss	$0.9664 \pm 0.001$	$0.9484 \pm 0.0023$
Only Hinge Loss	$0.9785 \pm 0.0021$	$0.9684 \pm 0.0037$
Hinge Loss & MI Regularizer	<b><math>0.9884 \pm 0.0011</math></b>	<b><math>0.9696 \pm 0.0018</math></b>

Table 10: Bicycle Dataset - Test Accuracy

we utilize the mini-batch mode. We should finally note that in the regularized training we utilize the hinge loss layer since, as we show, it performs steadily better than the cross entropy one in binary classification problems, however the cross entropy loss could also be utilized.

### 2.4.3 Performance evaluation

The experimental protocols, along with extensive performance evaluation experiments are provided in Appendices 7.10 and 7.11. Some experimental results considering the task of bicycle detection are provided in Table 10, demonstrating the improvements that can be obtained using the proposed method.

## 2.5 Fast Adaptive Inference using Early Exits

### 2.5.1 Introduction, objectives and summary of state of the art

In many robotics applications we need models that will be able to dynamically adapt to the available load, e.g., by being able to provide faster (and possibly less accurate) predictions when the load is higher (e.g., when a lot of persons appear in a frame) and more refined and accurate (yet slower) predictions when the load is lower (e.g., when only a few persons appear in a frame). Among the most promising approaches to tackle these problems are DL models with adaptive computational graphs, such as [192, 212, 14]. These models provide an easy and straightforward way to dynamically adapt the model on-the-fly to the available computational resources by altering the complexity of the model’s graph, i.e., by choosing a different computational path according to the available resources.

One straightforward way to achieve this is by adding early exits at various layers of the network [82, 192, 14]. These early exits allow for estimating the final output of the network at various points of the inference process, without having to feed-forward through the whole network. Early exits provide, in theory, a solid way to adapt the inference to the available resources. Unfortunately, they do not always lead to acceptable performance [14], since they have to deal with enormous feature maps (especially for the earlier layers of a convolutional neural network). To this end, aggressive subsampling methods are usually employed, e.g., global average pooling [238]. As a result, these methods ignore both the spatial information and the distribution of the extracted feature vectors, reducing the performance of early exits (in terms of accuracy). Even though this problem is partly addressed in [82] by using a series of densely connected structures, this also requires a significant number of structural changes in the architecture of a network and cannot be easily used with existing neural networks. Furthermore, most of the existing early exit-based approaches completely ignore the information extracted by the previous exit layers, throwing away information that is readily available and can be potentially used to further increase the prediction accuracy for the subsequent exit layers.

To this end, we propose using the Bag-of-Features (BoF) model [180], for compiling compact, yet rich and discriminative representations from the feature maps of each layer where an early exit is used. The proposed method is fully adapted to the needs of early exits, overcoming most of the limitations of existing related approaches. More specifically, our contribution focuses on:

1. A BoF-based formulation [149], which is capable of a) maintaining more information regarding the distribution of the extracted feature vectors and b) keeping more spatial information in the extracted representation, than the existing early exit methods. Note that the latter is especially important for earlier exits, where the receptive field of the convolutional layers is usually smaller.
2. An efficient hierarchical aggregation scheme, that works by implicitly constructing a common histogram representation space and then gradually refining the estimations of the network. This allows for taking into account the information that was already extracted by the earlier layers. Note that most of the existing formulations ignore this information [14].
3. A classification layer reuse approach that is capable of further reducing the number of parameters required for each early exit, minimizing the cost of adding additional exit layers.
4. An adaptive classification approach, that allows for selecting the most appropriate early exit according to the difficulty of each input sample, reducing the load to the system, as well as the energy consumption of DL models.

A summary of this work is described hereafter. The corresponding publications are listed below, and can be found in Appendix 7.1 and 7.2:

- [148] N. Passalis, J. Raitoharju, A. Tefas, and M. Gabbouj “*Efficient adaptive inference for deep convolutional neural networks using hierarchical early exits*”, Pattern Recognition, 2020
- [147] N. Passalis, J. Raitoharju, M. Gabbouj, and A. Tefas, “*Efficient Adaptive Inference leveraging Bag-of-Features-based Early Exits*”, IEEE International Workshop on Multimedia Signal Processing, 2020

## 2.5.2 Description of work performed so far

The proposed method is briefly introduced in this section, along with the used notation. The notation  $f_{\mathbf{W}}(\mathbf{x}, i)$  is used to refer to the response of the  $i$ -th layer of a neural network that is composed of a total of  $m$  layers. The trainable parameters of the network are denoted by  $\mathbf{W}$ , while the input to the neural network is denoted by  $\mathbf{x}$ . In this section, we focus on convolutional neural networks. Therefore, the input to the network is an image  $\mathbf{x} \in \mathbb{R}^{W \times H \times C}$ , where its width is denoted by  $W$ , its height by  $H$ , and the number of channels by  $C$ . It is worth noting that the proposed method can be also applied for networks operating on different kinds of signals. Furthermore, let  $\mathbf{y}^{(i)} = f_{\mathbf{W}}(\mathbf{x}, i) \in \mathbb{R}^{W_i \times H_i \times C_i}$  denote the output of the  $i$ -th convolutional layer. Similarly, the notation  $W_i$ ,  $H_i$  and  $C_i$  is used to refer to the width, height and number of channels of the corresponding feature map. Finally, the output of the network, which estimates the probability of each sample belonging to a different class (out of a total of  $N_C$  classes), is denoted by  $\mathbf{y} = f_{\mathbf{W}}(\mathbf{x}, m) \in \mathbb{R}^{N_C}$ .

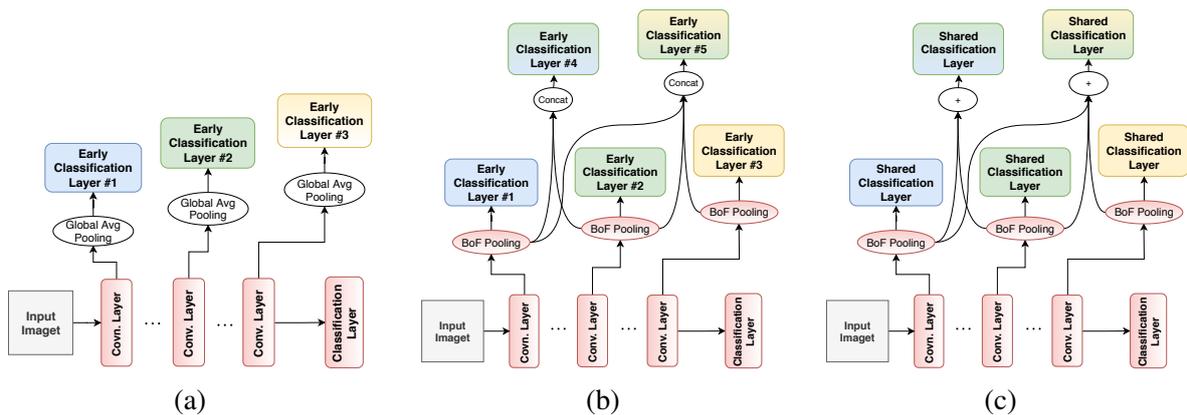


Figure 5: Comparing three different ways to use early exits. Fig. 5a demonstrates one typical way to use early exits to estimate the final output of the network at various points of its computational graph. Early exits can be improved by replacing global average pooling with the Bag-of-Features model to compile hierarchical early exits, as shown in Fig. 5b. Early exits can be further improved by sharing the same classification layer among different early exits and using additive histogram representations, as shown in Fig. 5c. This allows for reducing the number of parameters and forming implicit common representations spaces.

Also, let  $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$  denote a training set of  $N$  images. Each image is also annotated by a target (ground truth annotation) vector  $\mathbf{t}_i \in \mathbb{R}^{N_c}$ . The network can be then trained using back-propagation to minimize a loss function  $\mathcal{L}$ :

$$\mathbf{W}' = \mathbf{W} - \eta \sum_{j=1}^N \frac{\partial \mathcal{L}(f_{\mathbf{W}}(\mathbf{x}_j, m), \mathbf{t}_j)}{\partial \mathbf{W}}. \quad (17)$$

The notation  $\mathbf{W}'$  is used to refer to the parameters of the network after an update, while  $\eta$  denotes the learning rate. For this method, the cross-entropy loss function is used:

$$\mathcal{L}(\mathbf{y}, \mathbf{t}) = - \sum_{i=1}^{N_c} [\mathbf{t}]_i \log([\mathbf{y}]_i), \quad (18)$$

where the notation  $[\mathbf{y}]_i$  is used to refer to the  $i$ -th element of a vector  $\mathbf{y}$ .

In order to efficiently estimate the output of the network at various points of its computational graph, we employ an additional estimator  $g_{\mathbf{W}_i}^{(i)}(\cdot)$  on top of the feature maps extracted at the  $i$ -th layer as:

$$g_{\mathbf{W}_i}^{(i)}(\mathbf{y}^{(i)}) = g_{\mathbf{W}_i}^{(i)}(f_{\mathbf{W}_i}(\mathbf{x}, i)). \quad (19)$$

In this way, it is possible to estimate the final output of the network, without having to feed-forward through the whole architecture. We use the notation  $\mathbf{W}_i$  to refer to the parameters of the employed early exit.

Each feature vector extracted from each spatial location of feature map is first quantized using a set of  $N_K$  codewords, each one denoted by  $\mathbf{v}_{ij}$ , where  $i$  refers to the layer on which the BoF layer is used and  $j$  to the codewords. The codewords are used to represent the prototypical concepts captured by the corresponding layer [149], while a different set of codewords is used for different exits. The membership vector for each feature vector extracted from the  $i$ -th early

exit is calculated as:

$$[\mathbf{u}_{ikl}]_j = \frac{K([\mathbf{y}^{(i)}]_{kl}, \mathbf{v}_{ij})}{\sum_{m=1}^{N_K} K([\mathbf{y}^{(i)}]_{kl}, \mathbf{v}_{im})} \in [0, 1], \quad (20)$$

where  $(k, l)$  is the location from which the feature vector was extracted and  $K(\cdot)$  is a kernel function that measures the similarity between a codeword and an input vector. Then, the membership vectors  $\mathbf{u}_{ikl}$  are aggregated, leading to the final constant length histogram representation of the  $i$ -th object as:

$$\mathbf{s}^{(i)} = \frac{1}{W_i H_i} \sum_{k=1}^{W_i} \sum_{l=1}^{H_i} \mathbf{u}_{ikl} \in \mathbb{R}^{N_K}, \quad (21)$$

Note that this histogram vector provides a compact semantic summary of the features extracted from the corresponding layer, allowing for efficiently performing classification tasks, regardless of the actual size of the input feature map.

The representations extracted from the previous layers can be also readily re-used by concatenating them with the current representation, building in this way a hierarchical inference structure, as shown in Fig. 5c. Therefore, the representation  $\mathbf{s}^{(i,h)}$  extracted from the  $i$ -th layer is calculated as:

$$\mathbf{s}^{(i,h)} = \begin{cases} \mathbf{s}^{(i)} & \text{if } i = 1 \\ \mathbf{s}^{(i)} \frown \mathbf{s}^{(i-1,h)} & \text{if } i > 1 \end{cases}, \quad (22)$$

where  $\mathbf{a} \frown \mathbf{b}$  denotes the concatenation of vectors  $\mathbf{a}$  and  $\mathbf{b}$ . This approach allows for increasing the classification accuracy, with virtually zero additional cost, since the representations extracted from the previous layers can be readily cached and re-used.

In order to perform adaptive inference using a network equipped with early exits, an appropriate criterion must be used to decide whether inference should stop at a specific exit or continue until the next one. To this end, the difficulty of the input sample must be first estimated. In this work, we employ a simple, yet robust approach: the uncertainty of the network/difficulty  $r(\mathbf{x}, i)$  of an input sample  $\mathbf{x}$ , given the  $i$ -th exit, is estimated based on the confidence of the network on the class that corresponds to the neuron with the highest activation:

$$r(\mathbf{x}, i) = [g_{\mathbf{W}_i}^{(i)}(\mathbf{y}^{(i)})]_k, \quad (23)$$

where

$$k = \arg_{k'} \max [g_{\mathbf{W}_i}^{(i)}(\mathbf{y}^{(i)})]_{k'}. \quad (24)$$

Then, the most straightforward approach is to calculate a threshold for the  $i$ -th early exit based on the mean activation of the winning neurons during inference:

$$\mu_i = \frac{1}{N} \sum_{j=1}^N [g_{\mathbf{W}_i}^{(i)}(\mathbf{y}_j)]_k, \quad (25)$$

where  $g_{\mathbf{W}_i}^{(i)}$  is the output of the  $i$ -th early exit and again  $k = \arg \max g_{\mathbf{W}_i}^{(i)}(\mathbf{y}_j)$ . Then, the inference process can stop at the  $i$ -th exit whenever the confidence is higher than this threshold, i.e.,

$$r(\mathbf{x}, i) > \alpha \mu_i, \quad (26)$$

where  $\alpha$  is a hyper-parameter that allows for adapting the behavior of the inference process to the needs of each application. That is, using larger values for  $\alpha$  enables us to get more accurate

Table 11: Fast Adaptive Inference using Early Exits: Comparing the classification error and computational overhead for different variants of the proposed method. The classification error (%), computational complexity (MMAC) and required number of additional parameters is reported. The notation “AH-BoF (F, X)” refers to using the proposed method with  $X$  code-words, while “AH-BoF (S, X)” refers to employing additive histogram spaces and re-using the employed classification layer. When the “AH-SBoF” variant is used, then spatial segmentation into four regions is enabled.

Method	Early Exit - 1		Early Exit - 2		Hierarchical Exit		# Added Parameters.
	Error	MMAC	Error	MMAC	Error	MMAC	
FER-2013 Dataset (MobileNet v.2)							
AH-BoF (F, 16)	50.35%	98.60 (47%)	44.33%	143.24 (68%)	43.61%	143.39 (68%)	3.07k
AH-BoF (S, 16)	52.55%	98.60 (47%)	45.17%	143.24 (68%)	44.39%	143.39 (68%)	2.72k
AH-BoF (F, 32)	48.70%	98.75 (47%)	<b>43.66%</b>	143.46 (68%)	42.57%	143.76 (68%)	6.11k
AH-BoF (S, 32)	51.62%	98.75 (47%)	45.17%	143.46 (68%)	43.13%	143.76 (68%)	5.42k
AH-BoF (F, 128)	<b>48.39%</b>	99.64 (47%)	43.83%	144.80 (69%)	42.69%	146.00 (69%)	24.35k
AH-BoF (S, 128)	48.51%	99.64 (47%)	44.97%	144.80 (69%)	<u>42.13%</u>	146.00 (69%)	21.65k

The percentage of MMAC with the respect to the total MMAC required to feed-forward the network are reported in parenthesis. The best results for each early exit are reported in bold, while the best overall results are underlined.

Table 12: Fast Adaptive Inference using Early Exits: Comparing the proposed “AH-(S)BoF” method to two competitive early-exit approaches (ElasticNet [238] and BranchyNet [192]). The same notation as in Table 11 is used for the proposed method.

Method	Dataset	Early Exit - 1		Early/Hierarchical Exit - 2		# Added Parameters.
		Error	MMAC	Error	MMAC	
ElasticNet	FER-2013 Dataset	59.88%	98.45 (47%)	51.40%	143.01 (68%)	2.26k
BranchyNet	FER-2013 Dataset	49.21%	98.68 (47%)	44.41%	143.36 (68%)	5.96k
AH-BoF (F, 32)	FER-2013 Dataset	<b>48.70%</b>	98.75 (47%)	<b>42.57%</b>	143.76 (68%)	6.11k

The percentage of MMAC with the respect to the total MMAC required to feed-forward the network are reported in parenthesis.

Table 13: Fast Adaptive Inference using Early Exits: Classification error (%) and MFLOPs for different adaptive inference settings

Method	Setting 1	Setting 2	Setting 3	Setting 4	Class. Cost
FER-2013					
Baseline BoF-1	43.58 131.27 ± 45.54	42.94 135.89 ± 47.59	42.57 137.69 ± 48.33	42.30 139.97 ± 49.09	152.8386
Proposed BoF-1	43.58 131.27 ± 45.54	38.87 172.93 ± 47.77	38.67 183.43 ± 43.72	38.95 199.48 ± 31.20	39.0866
Baseline BoF-2	41.74 131.74 ± 45.49	41.46 136.34 ± 47.42	41.18 138.33 ± 48.22	40.87 140.27 ± 48.80	299.9848
Proposed BoF-2	41.74 131.74 ± 45.49	38.79 167.02 ± 48.87	38.84 178.49 ± 46.01	38.59 195.66 ± 35.08	60.0961

result (at the expense of higher inference time, since the corresponding early exit will be chosen less frequently), while using smaller values allows for increasing the speed, but possibly leading to less accurate classification results.

Note that the value of  $\alpha$  is not bounded, since any positive number can be used. We have experimentally found out that this approach is also especially sensitive, since even small changes in the value of  $\alpha$  can lead to significant changes in the behavior of the inference process. Furthermore, this behavior seems to be also related to the specific dataset/network used for training, with some datasets/networks requiring vastly different values compared to others. To overcome this limitation, we propose calculating the threshold based both on the mean activation of the current layer  $\mu_i$  (lower bound), as well as on the last layer of the network  $\mu_C$  (upper bound). Therefore, the threshold for selecting a specific early exit is calculated as:

$$r(\mathbf{x}, i) > (1 - \alpha)\mu_i + \alpha\mu_C, \quad (27)$$

for  $\alpha \in [0, 1]$ . In this way,  $\alpha$  is bounded between 0 and 1, while its value can be easily interpreted: as the value of  $\alpha$  increases from 0 to 1, less samples are using the current exit.

### 2.5.3 Performance evaluation

The experimental protocols, along with extensive performance evaluation experiments are can be found in the paper provided in Appendix 7.1 and Appendix 7.1, validating the effectiveness of the proposed method. Some indicative results on a face analysis dataset are provided in Table 11, where the ability of the proposed method to adapt to different scenarios is demonstrated. Furthermore, in Table 12, we demonstrate the effectiveness of the proposed method compared to other competitive methods in the literature. Some results on adaptive inference, further confirming that we can indeed reduce the time needed for inference for easier samples, are also provided in Table 13.

## 2.6 Active Control for Face Recognition

### 2.6.1 Introduction, objectives and summary of state of the art

Despite the recent achievements of DL in a wide range of different areas, most of the existing methods suffer from a significant drawback: they follow a static inference paradigm, as inherited by the traditional computer vision pipeline. More specifically, DL models perform inference on a fixed and static input, ignoring that robots, as well as many cyber-physical systems [115, 133], have the ability of *interacting* with the environment in order to better sense their surroundings. For example, consider the task of face recognition, where a robot has acquired a sub-optimal profile view of a subject. An existing static perception-based DL model might fail to recognize the subject from this view, especially if it has never been trained on profile face images. However, it is usually possible that the robot can acquire a better and more discriminative view by more appropriately repositioning itself with respect to the human subject. Therefore, in this case, the exact same DL model, will probably be able to recognize the subject, after the robot repositions itself in a more appropriate angle with respect to the subject. This approach, which is called *active perception* [7, 15, 175], allows for manipulating the robot/sensor in order to acquire a better and more clean view/signal, leading to improved situational awareness. It is worth noting that this process is very similar to the way humans and various animals interact and understand their environment. For example, humans tend to look from different angles

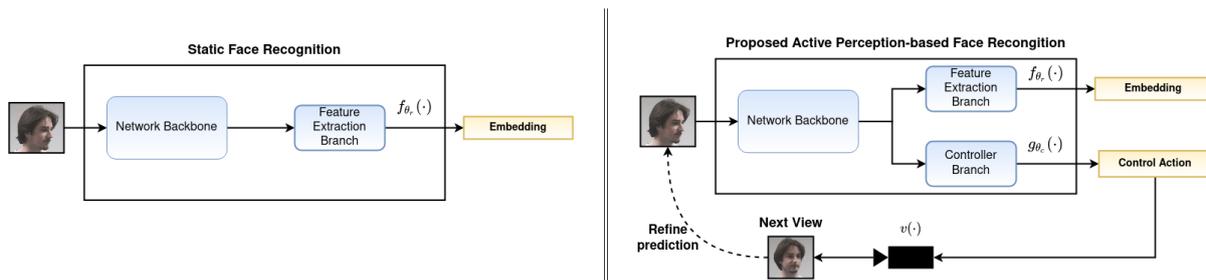


Figure 6: Comparing the proposed active perception-based approach to static face recognition. We simultaneously train a DL model to predict both a discriminative feature vector, which is used for face recognition, as well as a high-level description for the next best action, which can be used for acquiring a better and more discriminative view of the input. Note that the model is *not* trained for facial pose estimation, yet it implicitly learns the control actions that will lead to the view that will provide the best recognition results.

when trying to process complex visual stimuli, while many mammals have specialized muscles that rotate their ears toward the source of an audio signal [75].

A number of recent, yet quite primitive approaches, demonstrated that active perception can indeed increase the perception capabilities of various models. For example, in [8] it is demonstrated that developing a deep learning system that also predicts the next best move for a robot, using reinforcement learning, can significantly improve the performance of object detection, where the viewing angle, occlusions and the scale of each object can have a significant effect on the object recognition accuracy. Similar observations were also reported by more recent works [217, 69, 160]. At the same time, it is worth noting that active perception approaches often allow for developing faster and more lightweight DL models, since models are trained in order to solve a simpler problem. For example, in the case of object detection [8], a simpler model can be trained just for recognizing the objects from a limited number of angles, since a robot can usually acquire a more appropriate view that allows for accurately recognizing the corresponding object. To the best of our knowledge, despite these encouraging results in these areas, there have not been any thoroughly study on developing deep learning-based active perception models for human-centric robot perception, such face recognition.

Motivated by the aforementioned observations, we argue that the recognition accuracy of DL models can be improved by acquiring a more appropriate view, after manipulating the position of a robot inside the world. For example, moving a robot closer to a human is expected to improve the confidence of the robot when recognizing a human, as well as reduce the recognition errors. At the same time, we hypothesize that a significant part of the complexity in modern DL models arise from their ability to perform view invariant inference. Therefore, we argue that significantly smaller models can be used when active perception approaches are employed, without reducing recognition accuracy.

To this end, we proposed a DL-based active perception method for embedding-based face recognition, as well as we examined the behavior of such approached on a real multi-view face image datasets, shedding light on the aforementioned research questions. The proposed method is capable of simultaneously learning discriminative embeddings, that can disentangle the representations extracted from facial images that belong to different persons, as well as learning which should be the next control action by a robot carrying a camera in order to improve the face recognition confidence, as shown in Fig. 6.

A summary of this work is provided hereafter. The corresponding publication is listed below, and can be found in Appendix 7.3:

1. [153] N. Passalis, A. Tefas, “*Leveraging Active Perception for Improving Embedding-based Deep Face Recognition*”, IEEE International Workshop on Multimedia Signal Processing (MMSP), 2020

## 2.6.2 Description of work performed so far

In this section we briefly introduce the proposed method, along with the used notation. Even though static face recognition approach allows for achieving quite impressive face recognition results, as well as for easily training the model using a collection of static images, it comes with an important drawback: it ignores the ability of robotic systems to interact with the environment in order to get a more discriminative view for the task at hand. For example, a drone carrying a camera can fly to the appropriate direction in order to acquire a more clean frontal view of a person, allowing for analyzing the input with greater confidence. So, let  $\mathbf{x}_i \in \mathbb{R}^{W \times H \times C}$  denote a (cropped) face image, where  $W$ ,  $H$  and  $C$  are the width, height and number of channels of the corresponding image. Also, let  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_i, \dots, \mathbf{x}_N\}$  be a collection of  $N$  training images, while the binary variable  $d_{ij} \in \{0, 1\}$  is introduced to denote whether the  $i$ -th face image belongs to the same person as the one depicted in the  $j$ -th face image. To achieve the aforementioned goal, we introduce a trainable *controller*  $\mathbf{a}_t = g_{\theta_c}(\mathbf{x}^{(t)})$ , where  $\theta_c$  is a set of trainable parameters for the controller model, that receives an observation (image)  $\mathbf{x}^{(t)}$  from the environment at time  $t$  and provides an appropriate control command  $\mathbf{a}_t$  to the robot. Then, the updated observation is obtained by *executing* the corresponding action  $\mathbf{a}_t$  as:

$$\mathbf{x}^{(t+1)} = v(\mathbf{a}_t, t), \quad (28)$$

where  $v(\cdot)$  is either a model of the environment that returns the result of a simulated action  $\mathbf{a}_t$  at time  $t$ , or the real environment, in the case of deploying the model into a real system, where we execute the corresponding action and get the updated observation. In this way, the controller  $g_{\theta_c}(\cdot)$  provides a way to actively interact with the environment in order to get updated sensory stimuli, that will, in turn, lead to more accurate predictions for the embedding extractor  $f_{\theta_r}(\cdot)$ . For the rest of the section, we will refer to the environment  $v(\mathbf{a}_t, t)$  as  $v(\mathbf{a}_t)$  to avoid cluttering the used notation.

Both the feature extractor model  $f_{\theta_r}(\cdot)$ , as well as the controller model  $g_{\theta_c}(\cdot)$  must be appropriately trained for the task at hand. That is, the feature extractor model must be trained to extract discriminative embeddings, while the controller model must be trained in order to provide the appropriate control commands that will allow for getting a view that will maximize the face recognition accuracy. To this end, the optimization problem for the proposed active perception approach is defined as:

$$\theta_r, \theta_c = \arg \min_{\theta_1, \theta_2} \sum_{i=1}^N \sum_{j=1, j \neq i}^N \mathcal{L}(f_{\theta_1}(v(g_{\theta_2}(\mathbf{x}_i))), f_{\theta_1}(\mathbf{x}_j), d_{ij}). \quad (29)$$

Note that the active perception controller is allowed to manipulate only the first input to the loss function, since the other one corresponds to the fixed version that is stored in the database. Ideally, the controller should be aware of the remaining images contained in the database, since this information can be exploited in order to acquire a view that matches the view of the person

that is stored in the database. However, in this work we assumed that all images in the database are acquired under similar conditions, allowing for ignoring possible view variations of the database images.

It is also worth noting that (29) provides (at least) two different ways to minimize the employed loss: a) either by learning a powerful view-invariant feature extractor  $f_{\theta_r}(\cdot)$ , or b) by jointly learning a feature extractor along with an appropriate controller that is capable of acquiring images that makes the recognition problem easier. For the deployment, and after training both models simultaneously, the robot can either output the most probable prediction, or first appropriately control its camera (e.g., by moving itself or controlling a gimbal) in order to acquire an updated view. It is also possible that the controller will not provide any suggested action, as further explained below. In this case, we assume that the robot has already obtained an optimal view and no action should be performed.

Without loss of generality, in this worked we assumed that control will be performed in discrete steps on one (horizontal) axis, which lies along a sphere centered on the subject's face. The proposed approach can be directly extended to handle multiple axes as well. Therefore, the controller  $g_{\theta_c}(\cdot) \in \mathbb{R}^3$  support three possible actions: a) "stay", b) "go left", and c) "go right". Note that the controller  $g_{\theta_c}(\cdot)$  only provides high-level control commands, which must be appropriately translated into actual control commands by using an appropriate controller, e.g., an PID controller [156, 170].

Despite the updated formulation provided in (29), it is still not straightforward to directly optimize  $g_{\theta_c}(\cdot)$ , since the model  $v(\cdot)$  is usually not fully known and it is not differentiable. In this work we opt for a simple, yet efficient approach, which arises from the following assumption/observation: the recognition confidence is expected to monotonically increase or decrease when moving towards the same direction (at least for small continuous intervals). We call this *smooth control manifold* assumption. Even though it is possible that this assumption does not hold in practice, this approach allows for significantly simplifying the optimization process, as well as increasing the face recognition accuracy. Therefore, for each face image sampled during the training process, we propose executing all the three possible actions simultaneously (using the appropriate simulation environment/dataset) and observing the effect on the recognition confidence. Let  $\mathbf{x}_{i0}$ ,  $\mathbf{x}_{i1}$ , and  $\mathbf{x}_{i2}$  denote the updated facial image obtained after moving the robot to the left, right, or performing no action. Then, the training target for the controller can be trivially acquired by choosing the action that minimizes the distance between the representation of the correct face and the current face. Therefore, the controller target  $d_i^{(a)}$  for an image  $\mathbf{x}_i$  and a positive example  $\mathbf{x}_p$  is acquired as:

$$d_i^{(a)} = \arg \min_{k \in \{0,1,2\}} \|\mathbf{x}_{ik} - f(\mathbf{x}_p)\|_2. \quad (30)$$

For negative examples there are two options: a) either not training the controller with them, or b) training the controller in order to again minimize the distance between the embedding vectors (despite belonging to different persons). The motivation for the latter option is that the controller should always perform control in order to find the view that provides the best matching between face embeddings. In this work, the first option was selected, since it was experimentally shown to lead to slightly better recognition results.

Therefore, the loss to be minimized when optimizing the controller is defined as:

$$\mathcal{L}_g = \sum_{i=1}^N \sum_{j=1, j \neq i}^N d_{ij} \mathcal{L}_x(g_{\theta_c}(\mathbf{x}_i), d_i^{(a)}), \quad (31)$$

where  $\mathcal{L}_x$  denotes the categorical cross-entropy loss. The feature extractor can be still trained as before, i.e., by minimizing the loss:

$$\mathcal{L}_f = \sum_{i=1}^N \sum_{j=1, j \neq i}^N \mathcal{L}(f_{\theta_r}(\mathbf{x}_i), f_{\theta}(\mathbf{x}_j), d_{ij}), \quad (32)$$

Therefore, the final loss is obtained as:

$$\mathcal{L} = \mathcal{L}_g + \mathcal{L}_f \quad (33)$$

Table 14: Active Face Recognition: Experimental Evaluation using the HPID dataset

Method	Accuracy (Set 1)	Accuracy (Set 2)
Static Perception	54.1 ± 3.4%	49.9 ± 4.1%
Static Perception (finet.)	52.7 ± 4.2%	51.4 ± 4.6%
Proposed (1 step)	61.6 ± 5.1%	58.8 ± 7.0%
Proposed (3 steps)	<b>62.2 ± 5.9%</b>	<b>58.9 ± 6.5%</b>

Table 15: Active Face Recognition: Evaluating the effect of model size on face recognition accuracy

Method	Network	Accuracy	# Param.
Static Perception	0.25×	38.8 ± 6.6%	12k
Static Perception	0.5×	49.0 ± 5.5%	47k
Static Perception	1×	54.1 ± 3.4%	189k
Proposed	0.25×	<b>57.5 ± 5.8%</b>	14k
Proposed	0.5×	<b>60.2 ± 6.9%</b>	52k
Proposed	1×	<b>62.2 ± 5.9%</b>	197k

### 2.6.3 Performance evaluation

The proposed method can indeed lead to significant performance improvements, increasing both the face recognition accuracy (Table 14), as well reducing the complexity of the employed networks, reducing the hardware requirements for DL (Table 15). The experimental protocols, along with extensive performance evaluation experiments are provided in the paper provided in Appendix 7.3.

## 2.7 Active Face Shooting using Deep Reinforcement Learning

### 2.7.1 Introduction, objectives and summary of state of the art

Deep Reinforcement Learning (DRL) is capable of providing agents that operate directly on the raw input, e.g., pixels in the case of visual information, and learning how to perform various

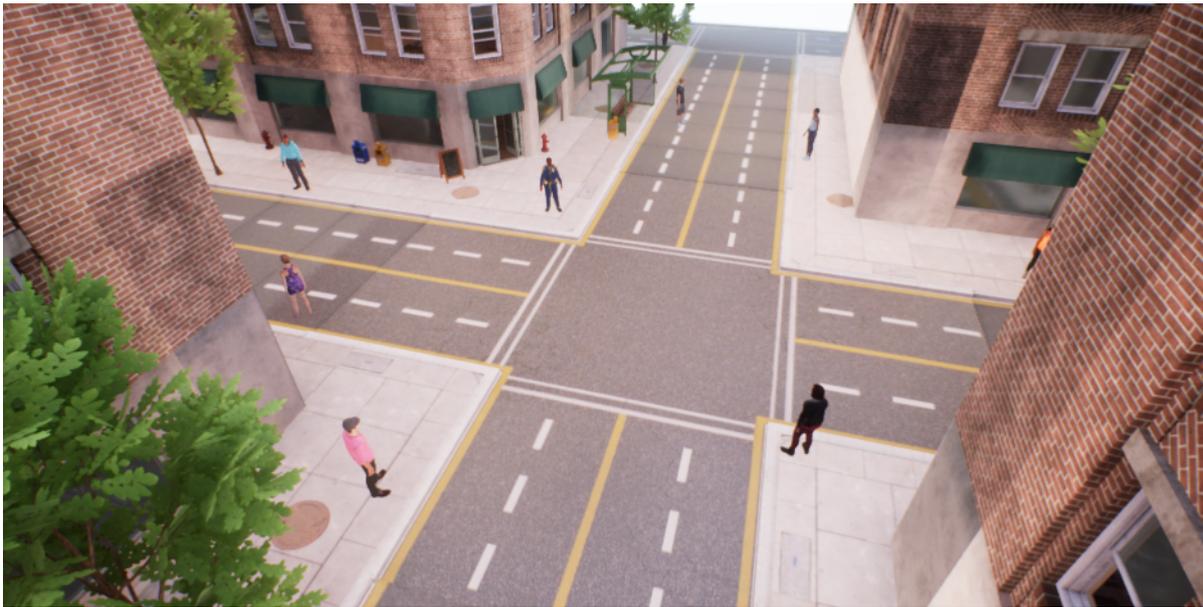


Figure 7: Open-world simulation environment

tasks, ranging from maneuvering vehicles [235], to using tools and cooperating to achieve their goals in complex environments [16]. In this work, we study the problem of controlling a drone that carries a camera in order to perform frontal view shooting of human subjects. This problem has been tackled with various approaches in the past, ranging from using face detectors and PID controllers [34], to developing DRL approaches for end-to-end control [152, 151], due to its importance for various emerging applications, such as active perception, human-robot interaction, aerial cinematography, etc. It is worth noting that applying DRL for this task is not straightforward, due to the lack of appropriate simulation environments. Existing approaches usually just employ static datasets that have been extended to support DRL, usually offering a very limited variety of view poses (since they are based on a limited collection of static images) [152], raising significant concerns on the behavior of DRL agents on less constrained environments.

To this end, in this work, we a) developed a realistic simulation environment, by employing a state-of-the-art graphics engine, that allows for training DRL agents under more challenging open-world scenarios and b) developed an appropriate DRL agent, along with a reward shaping methodology, to ensure its smooth convergence on the specific task. A snapshot of the developed simulation environment is shown in Fig. 7. Note that in this study we employed a significantly more complex and challenging environment compared to previous approaches, e.g., [152], in order to examine the behavior of DRL methods in an open-world setting, allowing to better understand how DRL methods behave and bringing us one step close to deploying such approaches on real applications. It is worth noting that, to the best of our knowledge, this is the first study in which an open-world simulator is employed for training DRL agents for performing frontal view shooting without using a static posed dataset, demonstrating that DRL can be successfully used, even under this challenging open-world setting. The developed DRL agent is trained using an established value-based DRL method and employs a specially designed reward shaping approach that was critical for successfully training the developed model.

A summary of this work is provided hereafter. The corresponding publication is listed below, and can be found in Appendix 7.4:

1. [208] A. Tzimas, N. Passalis, and A. Tefas, “*Leveraging Deep Reinforcement Learning*

*for Active Shooting under Open-World Setting*”, IEEE International Conference on Multimedia and Expo, 2020

### 2.7.2 Description of work performed so far

In this Section we briefly introduce the proposed method, including the developed simulation environment, along with the methodology used for developing the employed RL agent. The simulation environment was developed using the Unreal Engine 4. The environment represents a city that consists of four blocks, as already shown in Fig. 7. Other than the four buildings, bus stops, trees and other smaller props are also included. Furthermore, fourteen distinct 3D models of people were employed and inserted at various spots of the city. The selected models correspond to people of various ethnic groups, with a variety of clothes and features, allowing to examine the behavior of DRL methods under these more challenging conditions. Ten human 3D models were used for training, while the rest of them were used for evaluating the performance of the agent on previously unseen subjects.

A drone, equipped with an RGB camera, is also included in the simulation. The purpose of the developed agent is to appropriately control the drone in order to acquire frontal shots of the human models. Note that the drone can perform actions that only affect the position and orientation of the drone, while the orientation of the camera remains fixed, i.e., the agent does not control the gimbal on which the camera is mounted. Therefore, in order to acquire the desired shots, the agent must learn how to appropriately control the drone. The developed environment supports 9 different discrete actions:

1. *Forwards*: Move the drone towards the drone’s look direction with speed  $0.3m/s$  for  $0.3s$ ,
2. *Backwards*: Move the drone in the opposite direction than the drone’s look direction with speed  $0.3m/s$  for  $0.3s$ ,
3. *Left*: Move the drone to the left with respect to the drone’s look direction with speed  $0.3m/s$  for  $0.3s$ ,
4. *Right*: Move the drone to the right with respect to the drone’s look direction with speed  $0.3m/s$  for  $0.3s$ ,
5. *Up*: Move the drone up with speed  $0.3m/s$  for  $0.3s$ ,
6. *Down*: Move the drone down with speed  $0.15m/s$  for  $0.3s$ ,
7. *Rotate left*: Rotate the drone to the left with angular velocity  $10^\circ/s$  for  $0.3s$ ,
8. *Rotate right*: Rotate the drone to the right with angular velocity  $10^\circ/s$  for  $0.3s$ ,
9. *Stay*: Do not perform any action.

The above actions were implemented using the API provided by the AirSim plugin for Unreal Engine [173]. Also, note that the simulation was running 50 times faster than real time, so each second was equivalent to approximately 10 environment steps. Each episode lasts for 20 seconds, so about 200 steps are performed in each episode.

For every episode, one of the human models is selected at random, which will be the target human for the current episode. The selected human model is rotated to a random orientation so

that the background appearing in the shot is different in each episode. The drone is placed in a random position in front of the human model's face, pointing at it. The episode ends when either the human's face goes out of the frame, or the drone moves away from the target, or the drone collides with another object or finally when the 20 second time period is depleted from the start of the episode. The RL agent interacts with the developed environment and observes its state through the camera mounted on the drone, which provides one  $200 \times 200$  RGB frame to the agent at each step. The agent is then trained to learn how to appropriately control the drone directly from this raw pixel input.

Defining a meaningful reward function is critical for training RL agents. The agent is trained to achieve the desired distance with respect to the selected human subject, as well as to acquire a frontal shot. Therefore, the optimization objective should involve both the distance to the target, as well as the angular error with respect to the human subject. The distance error is defined as:

$$d = \sqrt{(fp_x - cp_x)^2 + (fp_y - cp_y)^2 + (fp_z - cp_z)^2}, \quad (34)$$

where  $(fp_x, fp_y, fp_z)$  denotes the optimal position in which the drone should be in order to take a frontal close-up shot and  $(cp_x, cp_y, cp_z)$  is the current camera position. The angular error is similarly defined as:

$$\theta = \arccos \left( \frac{\mathbf{td}^T \mathbf{ld}}{\|\mathbf{td}\|_2 \|\mathbf{ld}\|_2} \right), \quad (35)$$

where  $\mathbf{td} = (td_x, td_y, td_z)$  is the person's face direction with respect to the drone's camera and  $\mathbf{ld} = (ld_x, ld_y, ld_z)$  is the current look direction of the drone's camera. Therefore, the agents should be optimized with the respect to the combined distance and angular error.

However, simultaneously optimizing these two objectives when training an agent that directly operates on the raw camera input can be especially difficult [152]. To this end, we defined an appropriate reward shaping approach that a) further rewards the agent as it gets closer to the target (by providing an additional  $r_p$  reward), b) encourages the agent to select the stay action (by further providing an additional  $r_s > r_p$  reward), c) punishes the drone (by providing a negative reward equal to  $r_n$ ) when collides with the another object or losses the human subject, d) disentangles the angle from the actual returned reward, as far as the agent stays within certain limits. Therefore, the reward function used in this method is defined as:

$$r = \begin{cases} (1-d) + r_s + r_p, & \text{when } d < d_1 \text{ and } \theta < \theta_1 \text{ and the agent selects "Stay" (action 8)} \\ (1-d) + r_p, & \text{when } d < d_1 \text{ and } \theta < \theta_1 \\ -r_n, & \text{when } d > d_2 \text{ or } \theta > \theta_2 \text{ or the drone collides with another object} \\ 1-d, & \text{otherwise} \end{cases}, \quad (36)$$

where for all the conducted experiments we set:  $r_s = 90$ ,  $r_p = 10$  and  $r_n = 100$ . The error bounds for the distance and angle can be set according to the requirements of each application. In this work, we set  $d_1 = 0.1$ ,  $\theta_1 = 0.1$ ,  $d_2 = 1.13$  and  $\theta_2 = 0.4$ . Note that providing the additional  $r_s$  reward when the stay action is selected, allows not only to reach the target position and orientation, but also to train more robust agents that stay in place (select the "stay" action) when the desired shot is achieved and avoid erratic movements.

Table 16: Active Face Shooting: Drone control evaluation for frontal close-up shooting

Method	Eval. Type	Mean Reward	Mean Error
Baseline	Train	140	0.604
Proposed	Train	10,500	0.148
Baseline	Test	137	0.605
Proposed	Test	6,250	0.164

### 2.7.3 Performance Evaluation

The experimental protocols, along with extensive performance evaluation experiments are presented in the paper provided in Appendix 7.4. Some evaluation results are reported in Table 16, highlighting the effectiveness of the proposed approach, while a qualitative evaluation, which demonstrates the ability of the trained agent to control the trajectory of a drone in order to acquire a frontal shot, while avoiding erratic movements, is provided in Fig. 8.

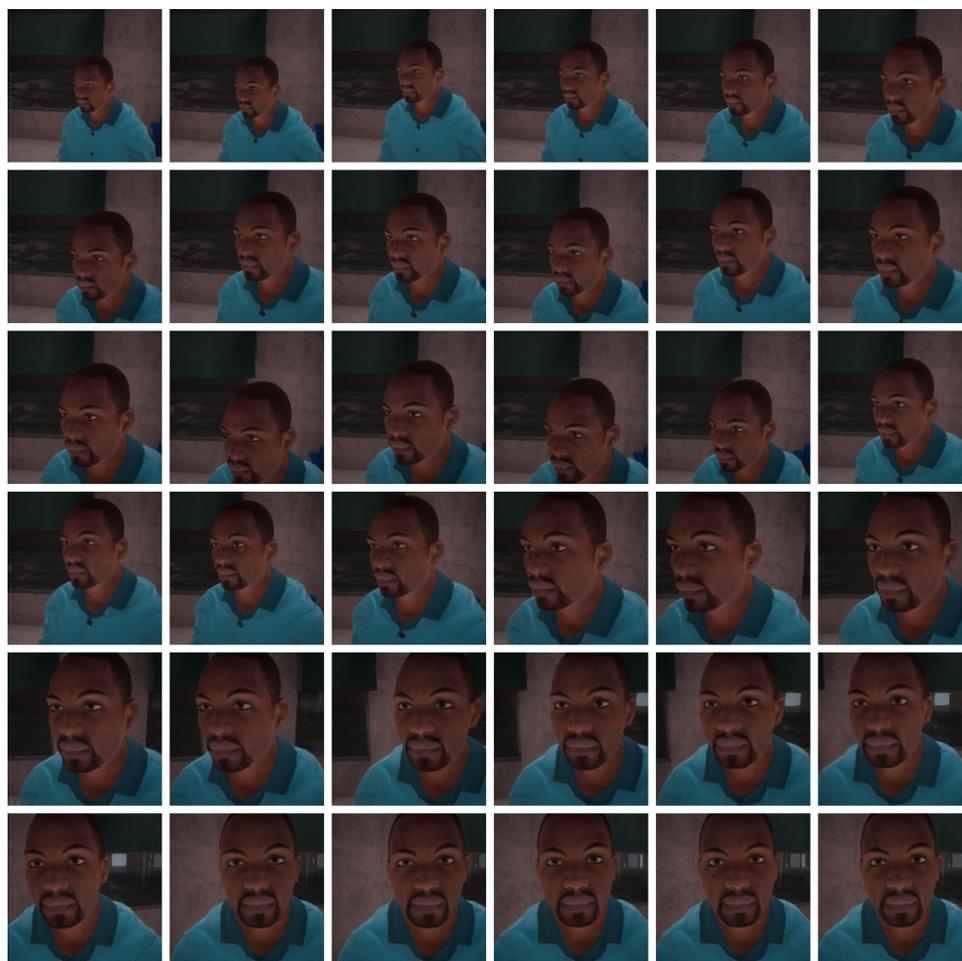
## 2.8 Active Face Recognition using Synthesized Facial Views

### 2.8.1 Introduction, objectives, Summary of state of the art

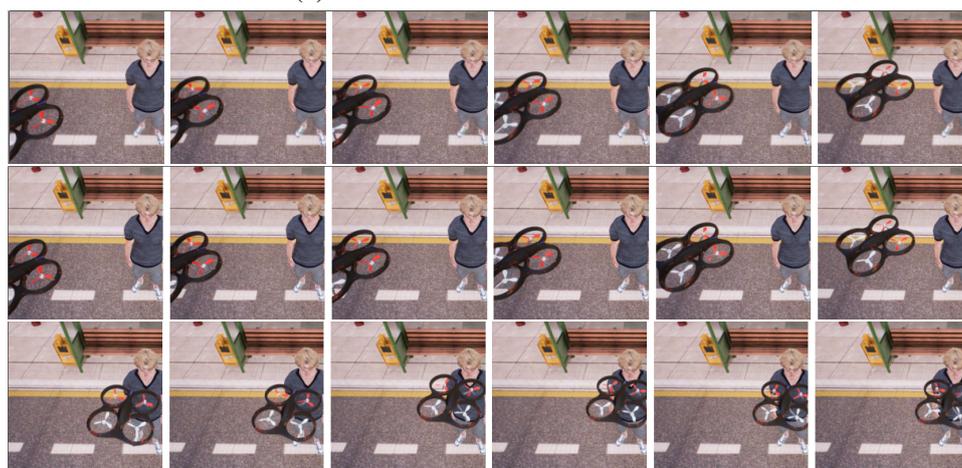
In recent years, robotics community has focused on research in the field of active vision and exploration. More specifically, considering the problems of reconstructing [135, 39] and recognizing an object from unknown a priori but spatially bounded statement [91], active vision methods are concerned with obtaining more information from the environment by actively choosing where and how to observe it using a camera [210]. This could be facilitated by camera motion which plays a fundamental role in the reconstruction/recognition of the obtained scene [56] or by moving a mobile robot in different positions which offer different sensor views of the candidate object. More specifically, active vision methods allow robots to conduct more sophisticated maneuvers with respect to limited energy consumption and motion complexity in order to take holistic views of a reconstructed/tracked [142] scene or object. In this way, through robotic sensors such as visual cameras robots could render novel views [137] of the observed scenes with photorealistic granularity [237] based on a single view image and a camera pose, so that they could replace the original ones.

An important application is the recognition of persons with deep face recognition methods. Deep Learning has lately dominated the interest of researchers in this area due to the superior performance achieved, and end-to-end deep learning approaches (i.e., approaches that perform both feature extraction and classification in an end-to-end fashion) have arisen with tremendous rates. Synthesized novel views of faces whose images were acquired through appropriate acquisition systems could be used for robot manipulation/navigation and guidance. Indeed, the face recognizer can obtain more robust results perceiving actively the respective face. Thus, using concurrently and efficiently robot's mechanics, the motion and functions of video feed shooting is being facilitated. On the other hand, a rendered facial view, produced by view synthesis procedure, could be obtained with high quality and replace the one originally captured by the robot's visual sensor; thus the robot does not necessarily have to capture it in a physical way to maximize the face recognition confidence. In this way, the recognition operation is being facilitated without conducting the respective sensor and/or robot movements.

The proposed method is dealing with active-vision-based face recognition which is rather understudied in current literature. A novel method is described in [140] which is comprised by



(a) View from the drone's camera



(b) View from a fixed point above the subject (the red rotors represent the front end of the drone)

Figure 8: Active Face Shooting: Two example control sequence of the trained DRL agent from two different perspectives

a neural network-based face recognizer along with a classifier and a decision making controller to allow the change of the examined facial snapshot viewpoint. Moreover, in [19], the frontal facial images which are fed to the face recognizer are generated by a pre-processing computer graphics step from the non-frontal input images.

### 2.8.2 Description of work performed so far

Exploiting photorealistic facial view rendering proposed in [237], the proposed active vision algorithm aims to discover the best facial view of a person in two different ways. Firstly, by considering the real cost of sensors movement and informativeness of the respective synthesized view. Secondly, through only view synthesis without actual robot movement. The proposed method is described as follows.

To begin with, as far as the first stage of the active vision algorithm is concerned, this utilizes an image acquisition system through a set of novel views. Let us denote as  $I_i$  a member of an image ensemble  $S$ , containing several viewing angles of a face that could be rendered using a single facial image captured from an arbitrary viewing angle. The robot will select the most appropriate facial rendered view that maximizes the confidence metric of the face recognition procedure with respect to informativeness of this view and the robot manipulation/movement cost in order to capture the respective real view of the face.

The information gain  $G_{I_i}$  that the rendered facial image  $I_i$  carries in comparison to other synthesized facial views could be measured by 2D image entropy [211]. According to information theory, the entropy (Eq.37) measures the amount of information carried by a signal, in our case the 2D facial image. This metric is used as an indication of randomness quantifying the texture complexity of a 2D image.

$$H(X) = - \sum_{i=1}^n p(x_i) \log_b(p(x_i)) \quad (37)$$

where  $Pr[X = x_i] = p(x_i)$  is the probability mass distribution of the signal  $X = x_i$ .

Considering the respective supposed robot movement cost  $C_{I_i}$  along with the information gain  $G_{I_i}$  of the facial view  $I_i$ , the objective utility function to evaluate the Best View (NBV) for all the remaining candidate views is  $U_{I_i}$  [91] in (Eq.38).

$$U_{I_i} = \frac{G_{I_i}}{\sum G_{I_i}} - \frac{C_{I_i}}{\sum C_{I_i}} \quad (38)$$

The proposed best rendered facial view (NBV) is the one that maximizes (Eq.39) the objective function  $U_{I_i}$  among all the candidate views, indicated as

$$NBV = \underset{I_i \in S}{argmax} \{U_{I_i}\} \quad (39)$$

where  $I_i$  the selected rendered facial image from the ensemble  $S$  of views rendered for several viewing angles and the estimation of information gain  $G_{I_i} = H(I_i)$  according to the global entropy (Eq.40) of an image  $I_i$  characterized by its histogram:

$$H(I_i) = - \sum_{j=1}^N hist_{norm}(L_{ij}) \log(hist_{norm}(L_{ij})) \quad (40)$$

where  $L_{ij}$  represents the  $N$  intensity levels of the  $m \times n$  image  $I_i(x,y)$  and  $hist_{norm}(L_{ij})$  is the histogram properly normalized to fit a probability distribution function:

$$\sum_{j=1}^N hist_{norm}(L_{ij}) = 1 \quad (41)$$

Alternatively, the proposed active vision algorithm uses only the maximization of the face recognition confidence metric in order to select the most representative rendered facial view from the image ensemble  $S$ . As result, one of the following that minimize the euclidean distance between the test facial image encoding from the train set encodings is selected: the best facial view (NBV) that maximizes the utility function in (Eq.38) or generally the test image  $I_i$  that maximizes the face recognition confidence. These two candidates results are not necessarily identical as proved by the experimental results.

Although, there are multiple solutions to compute the robot manipulation cost  $C_{I_i}$ , this will be the focus on future work as we concentrate in vision based parameters of active vision face recognition algorithm.

---

**Algorithm 1:** Active Vision Algorithm for NBV determination

---

```

1 Input: Initial facial image  $I_i$  in Pose(yaw, pitch);
   Result: Next Best Facial View - NBV
2  $I_i^{encoding} \leftarrow$  extract deep feature vector
3  $FRC \leftarrow$  do face recognition( $G$ )
4 while  $FRC \leq threshold$  do
5   for  $I_i \in S$  do
6     Position camera in  $Pose_i(yaw, pitch)$ ;
7      $I_i \leftarrow$  render the image
8      $G_{I_i} \leftarrow H(I_i)$ 
9     compute  $C_{I_i}$ ;
10     $I_i^{encoding} \leftarrow$  extract deep feature vector
11     $FRC_i \leftarrow$  do face recognition( $G$ )
12  for  $I_i \in S$  do
13    compute  $U_{I_i}$ 
14   $NBV \leftarrow \underset{I_i \in S}{argmax} \{U_{I_i}\}$ 
15   $NBV_{encoding} \leftarrow$  extract deep feature vector
16   $FRC_{NBV} \leftarrow$  do face recognition( $G$ )
17   $FRC \leftarrow \max(FRC_{NBV}, FRC_i)$ ;
18   $RESULT \leftarrow \underset{I_i \in \{S, NBV\}}{argmax} \{FRC\}$ 
19  -

```

---

**Deep Face Recognition Method:** Let us denote as gallery  $G$  a train set of facial image snapshots for the person identities that could be recognized using the face recognition approach. Similarly, the NBV images which are to be recognized are also included in the test set. The simple Deep Face Recognition method [60] is used by our proposed method. This method encodes an image in the train and test set using HOG [179] so as to create its simplified version. Using this simplified image, it finds the part of the image that most looks like a generic HOG encoding

of a face. Subsequently, it figures out the pose of the face by finding the main landmarks in the face. Once it finds those landmarks, it uses them to warp the image so that the eyes and mouth are centered. It passes the centered face image through a neural network which encodes it in a feature vector of 128 dimensions [171]. Finally, the k-NN classifier of the face recognizer checks all the faces stored in the gallery, to find which person has the closest feature vector to the test face's feature vector. More specifically, it compares the features from the train set with the feature vector of the test image and reaches a decision for the depicted person identity finding the k-nearest neighbors using euclidean distance.

### 2.8.3 Performance evaluation

Generally speaking, evaluation of existing active vision methods is difficult due to the lack of respective image datasets. This work attempts to somewhat remedy this situation, using facial image synthesis. We dealt with the creation of multi-view facial image datasets which augment existing single-view ones in the current bibliography. The new datasets are created through the unsupervised photorealistic face rotation and rendering algorithm [237] that creates novel views for several viewing angles starting from single-view facial images. To this end, performance evaluation of the proposed method is conducted utilizing the well known single-view facial image datasets **LFW**[83] and **CelebA**[128]. The first consists of 13,233 facial images for 5,749 person identities and the second consists of 202,599 facial images of 10,177 people captured "in the wild".

From a single image of the face and angles in the intervals  $[-60^\circ, 60^\circ]$  in yaw-axis and  $[0^\circ, 45^\circ]$  in pitch-axis, the high quality rendered views are produced by rotating the camera view point in front of the estimated 3D model of the head in  $5^\circ$  steps and rendering the respective synthesized images. Samples of the produced facial images are illustrated in Section 2.2 of D6.1 where the datasets are described. These images are to be contained in image ensemble  $S$  of the proposed algorithm after splitting them in train and test facial image sets in a 80%-20% manner.

In order to evaluate the contribution of any-pose facial view generation in face recognition confidence enhancement, the following experiment is performed. The face recognizer operates for each of the multi-view rendered facial views in the test set generated from respective single-views. One of the facial views contained in image ensemble  $S$  that maximizes its face recognition confidence metric (case FC in Table 17) or maximizes the facial image informativeness (case NBV in Table 17) against the other rendered facial images as previously denoted, is the candidate result facial view. It should be noted that the face recognizer is trained in two different ways. In the first way, the image gallery  $G = G_t$  consists of the single-view facial image views and in the second way, the gallery  $G = G_{t+m}$  includes additionally the respective several any-pose rendered train facial views. Thus, we can investigate how the face recognition confidence is being maximized due to the multi-view facial image generation used in training stage. Cases comparison is being held between testing single-view, any-pose or best-view facial images on a multi-view fashion with the two different training ways that aim at confidence maximization.

Experimental face recognition results are illustrated in Tables 17, 18. Table 18 shows that in most cases the face recognizer training with the rendered multi-view facial images could maximize the face recognition confidence (see Table 18) by minimizing the distance between the test and the training facial images. With respect to these confidences and generally without significant loss in face recognition accuracy (see Table 17), the latter presents better values with

the exception of some cases where it is almost the same or slightly different.

Table 17: Face Recognition Accuracy

Method\Dataset	LFW		CelebA	
	original	aligned	original	aligned
single-view FC with $G_t$	99.5%	-	97.0%	-
single-view NBV with $G_t$	99.2%	-	96.2%	-
single-view FC with $G_{t+m}$	<b>99.5%</b>	-	<b>97.5%</b>	-
single-view NBV with $G_{t+m}$	99.2%	-	96.1%	-
any-pose FC with $G_t$	89.5%	82.4%	88.3%	85.2%
any-pose FC with $G_{t+m}$	88.9%	82.3%	87.6%	81.5%
NBV-pose with $G_t$	93.9%	97.6%	92.9%	91.2%
NBV-pose with $G_{t+m}$	93.5%	97.6%	92.6%	89.7%

Table 18: Face Recognition Confidence (Minimum Distance)

Method\Dataset	LFW		CelebA	
	original	aligned	original	aligned
single-view FC with $G_t$	<b>0.01</b>	-	0.45	-
single-view NBV with $G_t$	0.04	-	0.37	-
single-view FC with $G_{t+m}$	0.19	-	<b>0.01</b>	-
single-view NBV with $G_{t+m}$	0.19	-	0.02	-
any-pose FC with $G_t$	0.46	0.31	0.36	0.37
any-pose FC with $G_{t+m}$	<b>0.01</b>	<b>0.01</b>	<b>0.01</b>	<b>0.01</b>
NBV-pose with $G_t$	0.20	0.21	0.48	0.21
NBV-pose with $G_{t+m}$	0.04	0.04	0.39	0.03

## 2.9 Multilinear Compressive Learning for Face Recognition

### 2.9.1 Introduction, Objectives and Summary of the state-of-the-arts

In many robotic applications, the computational aspect of data acquisition and processing plays an important role. This is especially important for robots that need to acquire data and make decisions in a real-time manner. Depending on the computational and energy capacity of the robot, sensor signal processing and decision making are conducted on-board, by the robot itself, or the sensor signals can be transferred to a centralized server, which can efficiently handle the intensive computation process. In the former scenario, computational efficiency and energy-efficiency are not only required for the decision making models but also for the sensor signal acquisition process. In the latter scenario, the sensor signals must be acquired and transmitted faster than real-time so that the robot can receive the decisions from a centralized server in a real-time manner. In both cases, sensor acquisition is a critical component.

The classical sample-based signal acquisition and manipulation approach usually involve separate steps of signal sensing, compression, storing or transmitting, then the reconstruction. This approach requires the signal to be sampled above the Nyquist rate in order to ensure high-fidelity reconstruction. Since the existence of spatial-multiplexing cameras, over the past decade, Compressive Sensing (CS) [24] has become an efficient and a prominent approach for signal acquisition at sub-Nyquist rates, combining the sensing and compression step at the hardware level. That is, what we obtain from a CS device is the compressed measurement of the original signal, which is several orders of magnitude more compressed than the classical sample-based acquisition paradigm at the Nyquist rate. Based on certain assumptions, which often hold for many types of data modalities such as visual or radar data, perfect reconstruction of the original signal can be done from the compressed measurements even when the acquisition rate is much lower than the Nyquist rate [23, 48].

While signal recovery plays a major role in some applications, in many application of robotics, the primary objective is of the sensor signal acquisition step is to make inference about the surrounding environments. These scenarios naturally led to the emergence of Compressive Learning (CL) concept [22, 36, 35, 161] in which the inference system is built on top of the compressively sensed measurements without the explicit reconstruction step.

While the amount of literature in CL is rather insignificant compared to signal reconstruction in CS, different attempts have been made to modify the sensing component in accordance with the learning task [13, 162], to extract discriminative features [36, 131] from the randomly sensed measurements or to jointly optimize the sensing matrix [5, 130] and the subsequent inference system. Improvements to different components of CL pipeline have been proposed, however, existing frameworks utilize the same compressive acquisition step that performs a *linear projection of the vectorized data*, thereby operating on the vector-based measurements and thus losing the tensorial structure in the measurements of multi-dimensional data.

Here we should note that the majorities of sensor data in robotics are multi-dimensional such as images, videos, point-clouds and so on. The multi-dimensional representation naturally reflects the semantic differences inherent in different dimensions or tensor modes. For example, the spatial and temporal dimensions in a video stream represent two different concepts, having different properties. Thus by exploiting this natural form of the signals and considering the semantic differences between different dimensions, one can expect to achieve superior performances compared to the vector-based approaches that are applied to multi-dimensional data.

While multilinear models have been successfully applied in Compressive Sensing or Ma-

chine Learning, to the best of our knowledge, we have not seen their utilization in Compressive Learning, which is the joint framework combining CS and ML. To this end, our work has made the following contributions:

- We proposed Multilinear Compressive Learning (MCL), a novel CL framework that consists of a multilinear sensing module, a multilinear feature synthesis component, both taking into account the multi-dimensional property of the signals, and a task-specific neural network. The multilinear sensing module compressively senses along each separate mode of the original tensor signal, producing structurally encoded measurements. Similarly, the feature synthesis component performs the feature learning steps separately along each mode of the compressed measurements, producing inputs to the subsequent task-specific neural network which has the structure depending on the inference problem.
- We showed both theoretically and empirically that the proposed MCL framework is highly cost-effective in terms of memory and computational complexity. In addition, theoretical analysis and experimental results also indicate that our framework scales well when the dimensionalities of the original signal increases, making it highly efficient for high-dimensional tensor signals.
- We conducted extensive experiments in face recognition tasks to validate the performance of our framework in comparison with its vector-based counterpart.
- We empirically analyzed the relationship between signal resolution, compression rate and learning performance and provided a performance indicator that can help practitioners to speedily rank different acquisition configurations without training the entire system.

A summary of this work is provided hereafter. The corresponding publications are listed below, and can be found in Appendix 7.12:

1. [202] D. T. Tran, M. Gabbouj, A. Iosifidis, “*Multilinear Compressive Learning*”, IEEE Transactions on Neural Networks and Learning Systems 2020.
2. [198] D. T. Tran, M. Gabbouj, A. Iosifidis, “*Performance Indicator in Multilinear Compressive Learning*”, IEEE Symposium Series on Computational Intelligence 2020.

## 2.9.2 Preliminaries

In this section, we denote scalar values by either lower-case or upper-case characters ( $x, y, X, Y \dots$ ), vectors by lower-case bold-face characters ( $\mathbf{x}, \mathbf{y}, \dots$ ), matrices by upper-case or Greek bold-face characters ( $\mathbf{A}, \mathbf{B}, \dots$ ) and tensor as calligraphic capitals ( $\mathcal{X}, \mathcal{Y}, \dots$ ). A tensor with  $K$  modes and dimension  $I_k$  in the mode- $k$  is represented as  $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_K}$ . The entry in the  $i_k$ th index in mode- $k$  for  $k = 1, \dots, K$  is denoted as  $\mathcal{X}_{i_1, i_2, \dots, i_K}$ . In addition,  $\text{vec}(\mathcal{X})$  denotes the vectorization operation that rearranges elements in  $\mathcal{X}$  to the vector representation.

**Kronecker Product:** The Kronecker product between two matrices  $\mathbf{A} \in \mathbb{R}^{M \times N}$  and  $\mathbf{B} \in \mathbb{R}^{P \times Q}$  is denoted as  $\mathbf{A} \otimes \mathbf{B}$  having dimension  $MP \times NQ$ , is defined by:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} \mathbf{A}_{11}\mathbf{B} & \dots & \mathbf{A}_{1N}\mathbf{B} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{M1}\mathbf{B} & \dots & \mathbf{A}_{MN}\mathbf{B} \end{bmatrix} \quad (42)$$

**Mode- $k$  product:** mode- $k$  product between a tensor  $\mathcal{X} = [x_{i_1}, \dots, x_{i_K}] \in \mathbb{R}^{I_1 \times \dots \times I_K}$  and a matrix  $\mathbf{W} \in \mathbb{R}^{J_k \times I_k}$  is another tensor of size  $I_1 \times \dots \times J_k \times \dots \times I_K$  and denoted by  $\mathcal{X} \times_k \mathbf{W}$ . The element of  $\mathcal{X} \times_k \mathbf{W}$  is defined as  $[\mathcal{X} \times_k \mathbf{W}]_{i_1, \dots, i_{k-1}, j_k, i_{k+1}, \dots, i_K} = \sum_{i_k=1}^{I_k} [\mathcal{X}]_{i_1, \dots, i_{k-1}, i_k, \dots, i_K} [\mathbf{W}]_{j_k, i_k}$ .

The following relationship between the Kronecker product and  $k$ -mode product is the cornerstone in Compressive Sensing and Multilinear Compressive Sensing:

$$\mathcal{Y} = \mathcal{X} \times_1 \mathbf{W}_1 \times \dots \times_N \mathbf{W}_N \quad (43)$$

can be written as

$$\mathbf{y} = (\mathbf{W}_1 \otimes \dots \otimes \mathbf{W}_N) \mathbf{x} \quad (44)$$

where  $\mathbf{y} = \text{vec}(\mathcal{Y})$  and  $\mathbf{x} = \text{vec}(\mathcal{X})$

**Compressive Sensing (CS)** [24] is a signal acquisition and manipulation paradigm that performs simultaneous sensing and compression on the hardware level, leading to large reduction in computation cost and the number of measurements. The signal  $\mathbf{y}$  working under CS is assumed to have a sparse or compressible representation  $\mathbf{x}$  in some basis or dictionary  $\Psi \in \mathbb{R}^{I \times I}$ , that is:

$$\mathbf{y} = \Psi \mathbf{x} \quad \text{with} \quad \|\mathbf{x}\|_0 \leq K \quad \text{and} \quad K \ll I \quad (45)$$

where  $\|\mathbf{x}\|_0$  denotes the number of non-zero entries in  $\mathbf{x}$ . While the dictionary presented in Eq. (45) is complete, i.e., the number of columns in  $\Psi$  is equal to the signal dimension  $I$ , we should note that signal models with over-complete dictionaries can also work, i.e.,  $\Psi \in \mathbb{R}^{J \times I}$  with some modifications [6].

With the assumption on the sparsity, CS performs the linear sensing step using the sensing operator  $\Phi \in \mathbb{R}^{M \times I}$ , acquiring a small number of measurements  $\mathbf{z} \in \mathbb{R}^M$  with  $M < I$ , from signal  $\mathbf{y}$ :

$$\mathbf{z} = \Phi \mathbf{y} \quad (46)$$

Eq. (46) represents both the sensing and compression step that can be efficiently implemented at the sensor level. Thus, what we obtain from CS sensors is a limited number of measurements  $\mathbf{z}$  that is used for other processing steps. By combining Eq. (45, and 46), the CS model is usually expressed as:

$$\mathbf{z} = \Phi \Psi \mathbf{x} \quad \text{with} \quad \|\mathbf{x}\|_0 \leq K \quad \text{and} \quad K \ll I \quad (47)$$

**Multilinear Compressive Sensing (MCS):** given a multi-dimensional signal  $\mathcal{Y} \in \mathbb{R}^{I_1 \times \dots \times I_N}$ , a direct application of the sparse representation in Eq. (45) requires vectorizing  $\mathbf{y} = \text{vec}(\mathcal{Y})$  and the calculations on  $\Phi \Psi \in \mathbb{R}^{M \times (I_1 \dots I_N)}$ , which is a very big matrix with the number of elements scales exponentially with  $N$ . Instead of assuming  $\text{vec}(\mathcal{Y})$  is sparse in some basis or dictionary, MCS adopts a sparse Tucker model [37] as follows:

$$\mathcal{Y} = \mathcal{X} \times_1 \Psi_1 \times \dots \times_N \Psi_N \quad (48)$$

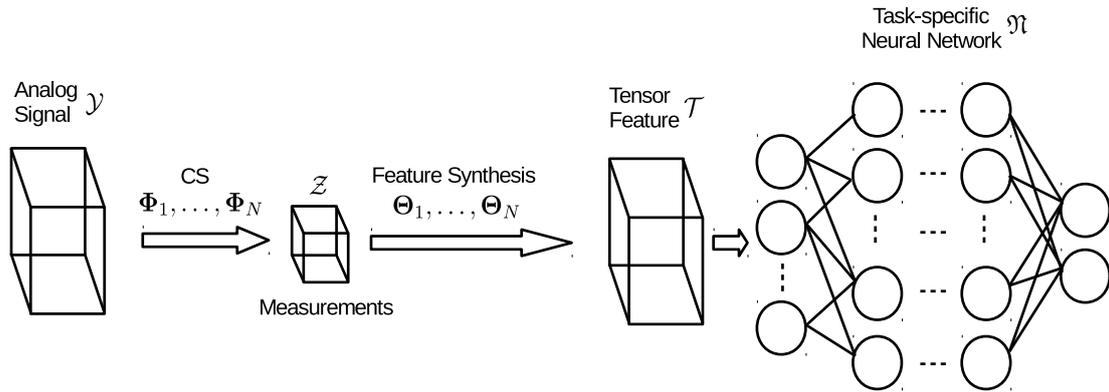


Figure 9: Illustration of Multilinear Compressive Learning framework

which assumes that the signal  $\mathcal{Y}$  is sparse with respect to a set of bases or dictionaries  $\Psi_n, n = 1, \dots, N$ . Since in some cases, the sensing step can be taken in a multilinear way, i.e., by using a set of linear operators along each mode separately, also known as separable sensing operators:

$$\mathcal{Z} = \mathcal{Y} \times_1 \Phi_1 \times \dots \times_N \Phi_N \quad (49)$$

that allows us to obtain the measurements  $\mathcal{Z}$  with retained multi-dimensional structure. From Eq. (43, 44, 48 and 49), the MCS model is often expressed as:

$$\mathbf{z} = (\mathbf{B}_1 \otimes \dots \otimes \mathbf{B}_N) \mathbf{x} \quad \text{with} \quad \|\mathbf{x}\|_0 \leq K \quad (50)$$

where  $\mathbf{z} = \text{vec}(\mathcal{Z})$ , and  $\mathbf{B}_n = \Phi_n \Psi_n$  ( $n = 1, \dots, N$ ). That is, any Multilinear Compressive Sensing model can be implemented (realized on the hardware level) as a (linear) Compressive Sensing model, but not vice versa.

### 2.9.3 Description of work performed so far

As mentioned in Section 2.9.1, the contributions of our work in this section include a Multilinear Compressive Learning (MCL) framework as well as our empirical study on a performance indicator of MCL. We will first describe the Multilinear Compressive Learning framework and its optimization procedure, then move on to the description of the performance indicator study.

**Multilinear Compressive Learning (MCL)** consists of three components as follows:

- **CS component:** the data acquisition step of the multi-dimensional signals is done via separable linear sensing operators  $\Phi_n, n = 1, \dots, N$  as in Eq. (49). As mentioned previously, in cases where the actual hardware implementation only allows vector-based sensing scheme, Eq. (50) allows the simulation of this multilinear sensing step. This component produces measurements  $\mathcal{Z}$  with encoded tensor structure, having the same number of tensor modes ( $N$ ) as the original signal.
- **Feature Synthesis (FS) component:** from  $\mathcal{Z}$ , this step performs feature extraction along  $N$  modes of the measurements  $\mathcal{Z}$  with the set of learnable matrices  $\Theta_n$ . Since the measurements typically have many fewer elements compared to the original signal  $\mathcal{Y}$ , the FS

component expands the dimensions of  $\mathcal{Z}$ , allowing better separability between the sensed signals from different classes in a higher multi-dimensional space that is found through optimization. While the sensing step performs linear interpolations for computational efficiency, the FS component can be either multilinear or nonlinear transformations. A typical simple nonlinear transformation step is to perform zero-thresholding, i.e., ReLU, on  $\mathcal{Z}$  before multiplying with  $\Theta_n, n = 1 \dots, N$ , i.e.,  $\text{ReLU}(\mathcal{Z}) \times_1 \Theta_1 \times \dots \times_N \Theta_N$ . In applications which require the transmission of  $\mathcal{Z}$  to be analyzed, this simple thresholding step can, before transmission, increase the compression rate by sparsifying the encoded signal and discarding the sign bits. While nonlinearity is often considered beneficial for neural networks, adding the thresholding step as described above further restricts the information retained in a limited number of measurements  $\mathcal{Z}$ , thus, adversely affects the inference system. Our experiments indicated that using nonlinear thresholding such as ReLU can indeed lead to performance degradation. Thus, our FS component only consists of the multilinear transformations.

- Task-specific Neural Network  $\mathfrak{N}$ : from the tensor representation  $\mathcal{T}$  produced by FS step, a neural network with task-dependent architecture is built on top to generate the regression or classification outputs. For example, when analyzing visual data, the  $\mathfrak{N}$  can be a Convolutional Neural Network (CNN) in case of static images or a Convolutional Recurrent Neural Network in case of videos. In CS applications that involve distributed arrays of sensors that continuously collect data, specific architectures for time-series analysis such as Long-Short Term Memory Network should be considered for  $\mathfrak{N}$ . Here we should note that the size of  $\mathcal{T}$  is also task-dependent and should match with the neural network component. For example, in object detection and localization task, it is desirable to keep the spatial aspect ratio of  $\mathcal{T}$  similar to  $\mathcal{Y}$  to allow precise localization.

The MCL framework is illustrated in Figure 9.

**Optimization:** as for deep neural networks, it is desirable to optimize all three components, i.e.,  $\Phi_n$ ,  $\Theta_n$  and  $\mathfrak{N}$ , with respect to the inference task. A simple and straightforward approach is to consider all components in this framework as a single computation graph, then randomly initialize the weights according to some popular initialization scheme [61, 73] and perform stochastic gradient descend on this graph with respect to the loss function defined by the learning task. However, this approach does not take into account any existing domain knowledge of each component that we have.

Instead of doing so, we propose to apply Higher Order Singular Value Decomposition (HOSVD) [37] and initialize  $\Phi_n$  with the left singular vectors that correspond to the largest singular values in mode  $n$ . The sensing matrices are then adjusted together with other components during the stochastic optimization process. This initialization scheme resembles the one proposed for vector-based CL framework which utilizes Principal Component Analysis (PCA). In a general case where one has no prior knowledge on the structure of  $\mathfrak{F}$ , a transformation that retains the most energy in the signal such as PCA or HOSVD is a popular choice when reducing dimensionalities of the signal. While for higher-order data, HOSVD only provides a quasi-optimal condition for data reconstruction in the least-square sense [64], since our objective is to make inferences, this initialization scheme works well.

With the aforementioned initialization scheme of CS component for a general setting, it is natural to also initialize  $\Theta_n$  in FS component with the right singular vectors corresponding to the largest singular values in mode  $n$  of the training data. With this initialization of  $\Theta_n$ , during the

initial forward steps in stochastic gradient descent, the FS component produces an approximate version of  $\mathcal{Y}$ , and in cases where a classifier  $\mathcal{C}$  pre-trained on  $\mathcal{Y}$  or its approximated version  $\hat{\mathcal{Y}}$  exists, the weights of neural network  $\mathfrak{N}$  can be initialized with that of  $\mathcal{C}$ .

After performing the aforementioned initialization steps, all three components in our MCL framework are jointly optimized using Stochastic Gradient Descent method, i.e., end-to-end training after the initialization.

**Surrogate Performance Indicator:** when building a MCL model for deployment, the computational requirements determine the range of feasible dimensions for  $\mathcal{Y}$  and  $\mathcal{L}$ . That is:

$$\begin{aligned} I_k^{\min} &\leq I_k \leq I_k^{\max} \\ M_k^{\min} &\leq M_k \leq M_k^{\max} \\ \forall k &= 1, \dots, K \end{aligned} \quad (51)$$

where  $I_k^{\min}$  and  $I_k^{\max}$  denote the lower- and upper-bounds of the feasible values for  $I_k$ .

When  $K$ ,  $(I_k^{\max} - I_k^{\min})$  or  $(M_k^{\max} - M_k^{\min})$  are large, the number of possible combinations of  $I_k$  and  $M_k$  can be enormous. Thus, we attempt to efficiently determine an optimal configuration of CS device (i.e., the choice of  $I_k$  and  $M_k$ ), without the need of conducting the entire optimization process of MCL for every feasible combination of  $I_k$  and  $M_k$ . One might guess that the higher the resolution  $I_k$  and number of compressed measurements  $M_k$  are, the higher the learning performance will be. However, this is not necessarily true as it will shown in Section 2.9.4.

By varying the number of compressed measurements and the device resolutions, we found out that higher sensor resolutions and higher number of measurements do not always lead to better learning performance. In addition, the compression rate also showed no clear linear relationship with the final learning performance. On the other hand, the Mean Squared Error (MSE) obtained during initializing the CS and FS components of MCL strongly correlates with the final learning performance. This suggests that this metric can be used as a surrogate measure of the final learning performance to gauge between different configurations of the CS device without conducting the entire optimization procedure, which is often time-consuming.

## 2.9.4 Performance evaluation

The experiment protocols, detailed description and analysis of different object recognition problems can be found in the Appendix 7.12. In the following, we provide representative results to demonstrate the effectiveness of the MCL framework in face recognition problems, which were conducted via two datasets: CelebA [128] and PubFig83 [158].

### CelebA Experiments

CelebA [128]: is a large-scale face attributes dataset with more than 200K images at different resolutions from more than 10K identities. In our experiment, we used a subset of 100 identities in this dataset which corresponds to 7063, 2373, and 2400 samples for training, validation, and testing, respectively. This dataset is used to benchmark the performance of MCL framework compared to its vector-based counterpart. In addition, this dataset is also used to study the scalability of MCL framework with different signal resolution. In order to evaluate the scalability, we resized the original images to different set of resolutions, including:  $32 \times 32$ ,

$48 \times 48$ ,  $64 \times 64$ , and  $80 \times 80$  pixels, which are subsequently denoted as CelebA-32, CelebA-48, CelebA-64, and CelebA-80, respectively.

The recognition performances with different compressed measurement configurations are shown in Table 19. In this table, we compared MCL framework with the state-of-the-art vector-based framework proposed by [240] at different levels of compression. It is obvious that using MCL framework, the face recognition accuracy is superior compared to its vector-based counterpart at all levels of compression using different types of backbone architectures with (ResNet) or without (AllCNN) residual connections.

In addition, in Table 20, we compare the memory complexity (the number of parameters) and computational complexity (the number of FLOPs) required during inference between the MCL framework and the vector-based framework. It is clear that the MCL framework is highly efficient in terms of memory and computational complexity compared to the vector-based framework.

Finally, the scalability of the MCL framework is demonstrated in Table 21 and Figure 10

Table 19: Test Accuracy with AllCNN and ResNet architecture as the task-specific network. Configuration refer to the dimension of the compressed measurements. “Oracle” refers to the task-specific network that was trained using uncompressed data

Configuration	CelebA-32 (ResNet)	CelebA-32 (AllCNN)
Oracle	93.08	92.58
$256 \times 3$ [240]	$87.29 \pm 00.00$	$89.25 \pm 00.00$
$20 \times 19 \times 2$ (our)	$92.00 \pm 00.48$	$92.36 \pm 00.07$
$28 \times 27 \times 1$ (our)	<b><math>93.54 \pm 00.32</math></b>	<b><math>92.74 \pm 00.31</math></b>
$102 \times 3$ [240]	$76.32 \pm 02.35$	$67.04 \pm 02.36$
$14 \times 11 \times 2$ (our)	$88.50 \pm 00.26$	$87.01 \pm 00.99$
$18 \times 17 \times 1$ (our)	<b><math>90.82 \pm 00.14</math></b>	<b><math>91.26 \pm 00.13</math></b>
$18 \times 3$ [240]	$67.29 \pm 00.44$	$63.76 \pm 00.22$
$9 \times 6 \times 1$ (our)	<b><math>69.90 \pm 00.41</math></b>	<b><math>68.49 \pm 00.28</math></b>
$6 \times 9 \times 1$ (our)	$67.39 \pm 00.34$	$65.92 \pm 00.77$

## PubFig83 Experiments

PubFig83 [158] is a medium-size dataset that contains 13002 facial images of 83 public figures. Since the photos were collected from the internet, the dataset represents the task of recognizing identities in uncontrolled situations using near-frontal faces. This dataset is used to study the performance indicator of the MCL framework. In order to simulate different resolutions of the CS device, we resized the images to 5 different resolutions, ranging from  $256 \times 256 \times 3$  to  $128 \times 128 \times 3$ . That is, we experimented with the set of feasible resolutions of  $\mathcal{Y}: I_1 \times I_2 \times I_3 \in \{256 \times 256 \times 3, 224 \times 224 \times 3, 192 \times 192 \times 3, 160 \times 160 \times 3, 128 \times 128 \times 3\}$ . Regarding compressed measurements  $\mathcal{Z}$ , we considered the following set of 6 feasible dimensions:  $M_1 \times M_2 \times M_3 \in \{30 \times 30 \times 1, 28 \times 28 \times 1, 26 \times 26 \times 1, 24 \times 24 \times 1, 22 \times 22 \times 1, 20 \times 20 \times 1\}$ . This leads to 30 combinations for the sizes of  $\mathcal{Y}$  and  $\mathcal{Z}$ . For each combination, the experiment was run 5 times and the average performance on the test set is reported.

In Table 22, we show the test accuracy with different input resolutions and compressed

Table 20: Computational complexity of the MCL framework and the vector-based framework, excluding the task-specific classifier component

Configuration	#Parameters	#FLOPs
$256 \times 3$ [240]	786K	1573K
$20 \times 19 \times 2$ (our)	<b>2.5K</b>	<b>5K</b>
$28 \times 27 \times 1$ (our)	3.5K	7K
$102 \times 3$ [240]	313K	627K
$14 \times 11 \times 2$ (our)	<b>1.6K</b>	<b>3.2K</b>
$18 \times 17 \times 1$ (our)	2.2K	4.5K
$18 \times 3$ [240]	55K	111K
$9 \times 6 \times 1$ (our)	<b>1.0K</b>	<b>1.9K</b>
$6 \times 9 \times 1$ (our)	<b>1.0K</b>	<b>1.9K</b>

Table 21: Test Performance &amp; Complexity of the MCL framework at different resolutions of the original CelebA dataset, with AllCNN as the task-specific classifier

Configuration	ACCURACY				MEASUREMENT RATE			
	CelebA-32	CelebA-48	CelebA-64	CelebA-80	CelebA-32	CelebA-48	CelebA-64	CelebA-80
Oracle	92.58	93.37	94.75	95.04	1.0	1.0	1.0	1.0
$20 \times 19 \times 2$	$92.36 \pm 0.07$	$91.24 \pm 0.23$	$92.62 \pm 0.36$	$93.01 \pm 0.32$	0.247	0.110	0.062	0.040
$28 \times 27 \times 1$	$92.74 \pm 0.31$	$92.43 \pm 0.19$	$93.39 \pm 0.52$	$93.42 \pm 0.37$	0.246	0.109	0.062	0.039
$14 \times 11 \times 2$	$87.01 \pm 0.99$	$87.00 \pm 0.87$	$87.75 \pm 0.27$	$88.67 \pm 0.26$	0.100	0.045	0.025	0.016
$18 \times 17 \times 1$	$91.26 \pm 0.14$	$90.17 \pm 0.30$	$91.36 \pm 0.43$	$91.89 \pm 0.46$	0.1	0.044	0.025	0.016
$6 \times 9 \times 1$	$65.92 \pm 0.77$	$66.56 \pm 0.05$	$66.69 \pm 0.46$	$66.03 \pm 0.37$	0.018	0.008	0.004	0.003
$9 \times 6 \times 1$	$68.49 \pm 0.28$	$68.69 \pm 0.96$	$67.75 \pm 0.62$	$67.31 \pm 0.49$	0.018	0.008	0.004	0.003
Configuration	#FLOP				#PARAMETER			
	CelebA-32	CelebA-48	CelebA-64	CelebA-80	CelebA-32	CelebA-48	CelebA-64	CelebA-80
$20 \times 19 \times 2$	5.0K	7.5K	10.0K	12.5K	2.5K	3.8K	5.0K	6.3K
$28 \times 27 \times 1$	7.0K	10.6K	14.1K	17.6K	3.5K	5.3K	7.0K	8.8K
$14 \times 11 \times 2$	3.2K	4.8K	6.4K	8.0K	1.6K	2.4K	3.2K	4.0K
$18 \times 17 \times 1$	4.5K	6.7K	9.0K	11.2K	2.2K	3.4K	4.9K	5.6K
$6 \times 9 \times 1$	1.9K	2.9K	3.9K	4.8K	1.0K	1.4K	1.9K	2.4K
$9 \times 6 \times 1$	1.9K	2.9K	3.9K	4.8K	1.0K	1.4K	1.9K	2.4K

measurement configurations on the PubFig83 dataset. In addition, the lower section of Table 22 shows the corresponding Mean-Squared-Error (MSE) obtained during the initialization of the CS and FS components. We further visualize the input resolutions, the classification error (CE) and the MSE obtained during initialization of the CS and FS components in Figure 11. As we can see from Table 22 and Figure 11, the MSE obtained during the initialization of the CS and FS component is highly correlated with the final learning performance, while there is no clear relationship between the learning performances and the input resolution or the compression rate. This suggests that we can rapidly gauge and rank different configurations of the input resolutions and the compression rate by only conducting the initialization phase of the CS and FS component in the MCL framework to narrow down potential candidate configurations.

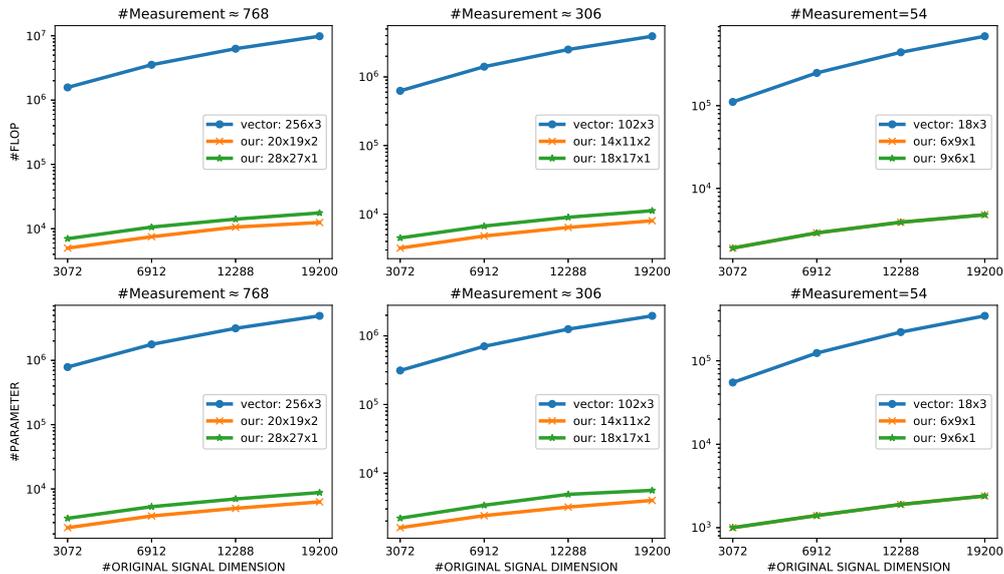


Figure 10: #FLOP and #PARAMETER versus the original dimensionalities of the signal, measured in the MCL framework and the vector-based framework, excluding the task-specific classifier. The x-axis represents the original dimension of the input signal. The y-axis on the first row represents the number of FLOPs in log scale while the y-axis on the second row represents the number of parameters

Table 22: Test Performances of PubFig83 Dataset. The upper section shows test accuracy while the lower section shows Mean Squared Error (MSE) measured on test set when initializing the CS and FS components. **Bold-face** numbers indicate the top-3 accuracy and the corresponding MSE

Test Accuracy (%)		$\mathcal{Y}$ Dimension ( $I_1 \times I_2 \times I_3$ )				
		$256 \times 256 \times 3$	$224 \times 224 \times 3$	$192 \times 192 \times 3$	$160 \times 160 \times 3$	$128 \times 128 \times 3$
$\mathcal{X}$ Dimension ( $M_1 \times M_2 \times M_3$ )	$30 \times 30 \times 1$	66.01	74.38	67.69	78.36	44.49
	$28 \times 28 \times 1$	<b>80.86</b>	<b>79.46</b>	57.17	72.12	58.17
	$26 \times 26 \times 1$	77.53	<b>79.32</b>	67.47	47.03	47.59
	$24 \times 24 \times 1$	58.54	53.70	54.43	62.00	58.85
	$22 \times 22 \times 1$	57.53	71.16	72.23	77.39	75.67
	$20 \times 20 \times 1$	58.44	68.17	38.35	43.23	71.45
MSE		$\mathcal{Y}$ Dimension ( $I_1 \times I_2 \times I_3$ )				
		$256 \times 256 \times 3$	$224 \times 224 \times 3$	$192 \times 192 \times 3$	$160 \times 160 \times 3$	$128 \times 128 \times 3$
$\mathcal{X}$ Dimension ( $M_1 \times M_2 \times M_3$ )	$30 \times 30 \times 1$	0.1368	0.1161	0.0408	0.0256	0.2490
	$28 \times 28 \times 1$	<b>0.0185</b>	<b>0.0196</b>	0.0395	0.1357	0.1957
	$26 \times 26 \times 1$	0.0192	<b>0.0173</b>	0.0320	0.2008	0.2135
	$24 \times 24 \times 1$	0.0480	0.1078	0.1675	0.0567	0.0425
	$22 \times 22 \times 1$	0.1927	0.0241	0.0306	0.0167	0.0189
	$20 \times 20 \times 1$	0.0323	0.0254	0.1218	0.0616	0.0199

## 2.9.5 Future work

At the moment, we are developing methods that can train a single MCL model that can operate at multiple compression rates, thereby enabling the adaptive rate processing capacity, especially when the acquired signals are transmitted and processed in a centralized server. Depending on



Figure 11: PubFig83 classification error (CE) versus compression rates, input resolutions and MSE obtained during initialization of CS and FS components

the status of the transmission bandwidth, the robot can adaptively acquire the sensor signals at an optimal rate and the task-specific network can also adapt to the variable amount of compressed measurements.

## 2.10 Deepbots: A Webots-based Deep Reinforcement Learning Framework for Robotics

### 2.10.1 Introduction, objectives and Summary of state of the art

A critical bottleneck in the process of developing active perception algorithms that are integrated with advanced simulation tools is the large development effort that is required just to interface the learning algorithm with the corresponding simulators, especially when Deep Reinforcement Learning (DRL) algorithms are used. Indeed, DRL is increasingly used to train robots to perform complex and delicate tasks. Despite the potential of DRL on robotics, such approaches usually require an enormous amount of time to sufficiently explore the environment and manage to solve the task at hand, often suffering from low sample efficiency [185]. Furthermore, during the initial stages of training, the agents take actions at random, potentially endangering the robot's hardware. To circumvent these restrictions, researchers usually first run training sessions on realistic simulators, such as Gazebo [104], and OpenRAVE [44], where the simulation can run at accelerated speeds and with no danger, only later to transfer the trained agents on physical robots. However, this poses additional challenges [52], due to the fact that simulated environments provide a varying degree of realism, so it is not always possible for the agent to observe and act exactly as it did during training in the real world. This led to the development of more realistic simulators, which further reduce the gap between the simulation and the real world, such as Webots [136], and Actin [4]. It is worth noting that these simulators not only simulate the physical properties of an environment and provide a photorealistic representation of the world, but also provide an easily parameterizable environment, which can be

adjusted according to the needs of every different real life scenarios.

Even though the aforementioned simulators provide powerful tools for developing and validating various robotics applications, it is not straightforward to use them for developing DRL methods, which typically operate over a higher level of abstraction that hides low-level details, such as how the actual control commands are processed by the robots. This limits their usefulness for developing DRL methods, since their steep learning curve and the enormous amount of development required to interface with DRL methods, considerably restricts their use by DRL researchers. To this end, OpenDR has developed an open-source framework that can overcome the aforementioned limitations by supplying a DRL interface that is easy to use. More specifically, the developed framework, called “deepbots”, combines the well known OpenAI Gym [20] interface with the Webots simulator in order to establish a standard way to employ DRL in real case robotics scenarios. Deepbots aims to enable researchers to use RL in Webots for many different scenarios (e.g., developing active perception algorithms). In essence, deepbots acts as a middleware between Webots and the DRL algorithms, exposing a Gym style interface with multiple levels of abstraction. The framework uses design patterns to achieve high code readability and re-usability, allowing to easily incorporate it in most research pipelines. The aforementioned features come as an easy-to-install Python package that allows developers to efficiently implement environments that can be utilized by researchers or students to use their algorithms in realistic benchmarking. At the same time, deepbots provides ready-to-use standardized environments for well-known problems. Finally, the developed framework provides some extra tools for monitoring, e.g., tensorboard logging and plotting, allowing to directly observe the training progress.

There is an increasing number of works that attempt to formalize and facilitate the usage of RL in robotic simulators. However, none of these works target the state-of-the-art Webots simulator. For example, Gym-Ignition [55] is a work which aims to expose an OpenAI Gym interface to create reproducible robot environments for RL research. The framework has been designed for the Gazebo simulator and provides interconnection to external third party software, multiple physics and rendering engines, distributed simulation capabilities and it is ROS compliant. Other than that, [226] extends the Gym interface with ROS compliance and it uses the Gazebo simulator as well. The latest version of this work [132] is compatible with ROS 2 and is extended and applied in more real world oriented examples. All of these works are limited by the low quality graphics provided by the Gazebo simulator, rendering them less useful for DRL algorithms that rely on visual input. Finally, Isaac Gym is a powerful software development toolkit providing photorealistic rendering, parallelization and is packed as a unified framework for DRL and robotics. However, its closed source nature can render it difficult to use, especially on scenarios that deviate from its original use cases. To the best of our knowledge this is the first work which provide a generic OpenAI Gym interface for Webots, standardizing the way DRL algorithms interface with Webots and provide easy access to a state-of-the-art simulator.

A summary of this work is provided hereafter. The corresponding publication is listed below, and can be found in Appendices 7.16:

- [103] Kirtas M., Tsampazis K., Passalis N., Tefas A., “Deepbots: A Webots-Based Deep Reinforcement Learning Framework for Robotics”, IFIP International Conference on Artificial Intelligence Applications and Innovations, 2020.

### 2.10.2 Description of the work

Deepbots follows the same agent-environment loop as the well-known and established OpenAI Gym framework [20], with the only difference being that the agent, which is responsible for choosing an action, runs on the supervisor and the observations are acquired by the robot. This master-minion protocol is not problem-specific and thus has the advantage of generalization, due to the fact that it can be used in more than one examples. That makes it easier to construct various use cases and utilize them as benchmarks. In this way, the deepbots framework acts as a wrapper, meaning that it wraps up and hides certain operations from the users, so that they are able to focus on the DRL task, rather than handling all the technical simulator-specific details. At the same time, deepbots also enriches the training pipeline with live monitoring features, which helps researchers get early observations about the fundamental parts of the training process. All these features contribute into providing a powerful DRL-oriented abstraction over Webots, allowing researchers to quickly model different use cases and simulation environments, as well as employ them to develop sophisticated DRL algorithms.

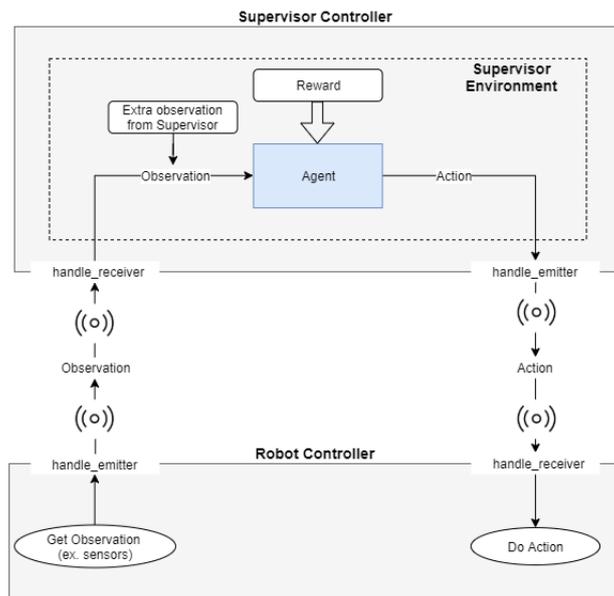


Figure 12: Deepbots supervisor-controller communication

Before describing deepbots, it is useful to briefly review the way Webots handles various simulation tasks. A robot should be a node in the simulation world under the root node which contains a controller. Controllers are scripts responsible for the node's functionality. For RL purposes, it is necessary to include a special supervisor node which has full knowledge of the world and it can get information for and modify every other node.

With respect to the aforementioned logic, deepbots gives the ability to easily construct DRL tasks with minimal effort. The basic structure of deepbots communication scheme is depicted in Fig. 12, where the supervisor controller is the script of the supervisor and the robot controller is the script of the robot. The communication between supervisor and robot is achieved via emitters/receivers, which broadcast and receive messages respectively. Without loss of generality, the supervisor is a node without mass or any physical properties in the simulation. For example, in a real case scenario, a supervisor could be a laptop which transmits actions to the robot, but

without interacting with the actual scene. Furthermore, the emitter and the receiver could be any possible device, either cable or wireless, properly set up for this task.

In order to implement an agent, the user has to implement two scripts at each side of the communication channel and the framework handles the details. On the supervisor side, the user has to create a Gym environment with the well known abstract methods and train/evaluate the DRL model. While on the other side, a simple script has to be written for reading values from sensors and translating messages to the actual values needed by the actuators. The deepbots framework runs all the essential operations needed by the simulation, executes the step function and handles the communication between the supervisor and the robot controller in the background.

Deepbots also contains a collection of ready-to-use environments, which showcase uses of the framework in toy or complicated examples. On the one hand, the community can contribute new environments and use cases to enrich the existing collection. On the other hand, this collection can be used by researchers to benchmark RL algorithms in Webots. Three environments of varying complexity are included in the framework at its release. For example, one of them is a typical find target and avoid obstacles task with a simple world configuration. For this task the E-puck robot is used [62], which is a compact mobile robot developed by GCTronic and EPFL and is included in Webots. The world configuration contains an obstacle and a target ball. Different world configurations with incremental difficulty have been used in the training sessions for better generalization.

## 3 Deep person/face/body part tracking, human activity recognition

### 3.1 Lightweight Human Activity Recognition

#### 3.1.1 Introduction and objectives

Human Activity Recognition is the broad task of recognising the actions of humans in a digital medium such as video. While there are multiple sub-tasks in Human Activity Recognition, including localisation of actions in space and time, and with different modalities such as accelerometer data, the most common one is Trimmed Action Recognition on videos, which is also the focus of this effort. This task amounts to the classification of a trimmed video, which is spatially and temporally trimmed to the action of interest, and constitutes an important building block for downstream applications such as spatio-temporal action localisation or as part of the visual cognition module in an autonomous robot. However, video-data and neural network operating on them, have a large computational cost. In embedded and low-power settings (such as in autonomous robots) the computational demand of human activity recognition using models that achieve state-of-the-art accuracy is prohibitive. These settings pose a situation with a strict computational budget, and rather than the absolute achievable accuracy, the trade-off between computational efficiency and accuracy is of primary interest. For inclusion in the OpenDR toolkit, we provide implementations and comprehensive benchmarks on four state-of-the-art model configurations for lightweight Trimmed Action Recognition in the X3D model family [54].

#### 3.1.2 Summary of state of the art

Many approaches to HAR in videos using deep neural networks have been explored, but the main themes can be broadly categorised in two families: 2D CNN + RNN architectures, where a CNN is applied to each image frame and a Recursive Neural Network (RNN) accumulates knowledge along the time [46, 94]; and 3D CNNs and their extensions, in which spatio-temporal convolutional kernels are used in place of the 2D (spatial) kernels traditionally employed for tasks on still images [194, 27, 196]. While the former is appealing due to simple reuse of well-performing image classification networks pretrained on ImageNet [40] and easily lend themselves to online processing, the latter family of models have achieved better classification accuracies in large-scale HAR benchmarks. In the 3D CNN family, C3D [194] was among the first to successfully employ 3D convolutions for HAR, by using a deep convolutional architecture similar to VGG-16 but with 3D convolutions in place of their 2D counterparts. At the time, however, the limited size of available video datasets, did not allow C3D to surpass the best 2D CNN + RNN methods. Inflated 3D (I3D) [27] solved the problem of limited data by reusing the weights of pretrained image classifiers and "inflating" them to 3D, thereby achieving new state-of-the-art results. In an effort to make more efficient 3D kernels, the R(2+1)D [196] method decomposed the 3D convolutional kernels as a 2D convolution along the spatial dimensions, and a 1D convolution in time. The newfound savings were used to increase the number of neurons in each layer, once again setting record results on HAR benchmarks.

The primary limitation of the above-described models in context of an embedded application such as robotics, is their computational cost and large memory consumption. For the image classification task, there has been great progress in the reduction of model sizes, while keeping pre-

diction accuracy at a high level. In architectures such as MobileNet [80, 169], SqueezeNet [84], ShuffleNet [236], and EfficientNet [187], major savings were made by using cheap grouped 2D convolutions, and bottleneck projections by 1D convolutions, that integrate information across channels. A recent strain of work in video classification and Trimmed Action Recognition, has extended the MobileNet, SqueezeNet, ShuffleNet [108], and EfficientNet [54] models using grouped 3D convolutions, to achieve better cost/accuracy trade-off than prior method. X3D is the extension of EfficientNet and holds the current best cost/accuracy trade-off in multiple computational regimens, ranging from low to higher budget settings.

### 3.1.3 Description of work performed so far

The focus of this effort in Human Activity Recognition has been to benchmark and validate the performance of the four publicly available models in the X3D family. These are denoted as XS, S, M, and L, c.f. their computational cost, and their inclusion in the OpenDR toolkit will serve as a strong baseline for a wide range of computational requirements. In the toolkit, each model is available with weights that were pretrained on the large-scale Kinetics-400 [97] dataset, from which good performance on other datasets can be achieved by means of fine-tuning. To server as realistic accuracy benchmarks, the models were tested using a varying number of clips to reflect the range of applications from offline processing, where multiple clips can be used for inference on a video, to low latency online processing, where only a single clip can be used for inference. Moreover, we have measured the floating point operations per second (FLOPS), model sizes and out-of-the-box clips-per-second that toolkit user can expect on respectively desktop CPU, GeForce RTX 2080 Ti GPU, and the embedded devices Nvidia Jetson TX2 and Nvidia Jetson Xavier.

### 3.1.4 Performance evaluation

In most literature on Human Activity Recognition, it is commonplace to report metrics only for multi-clip testing on a video. Here, a clip denotes a subset of the full video of a limited number from frames at a certain frame-rate and spatial resolution. In fact, the number of frames, frame rate and resolution are important hyperparameters, which have a considerable influence on both the accuracy and computational cost the classifier. This was exploited in the creation of the different X3D model configurations as seen in Table 23. Notably, the only difference between the X3D-XS and X3D-S models is the number of frames and their sampling rate, though their 1-clip accuracy on Kinetics400 differs by 5.5 %-points and FLOPS by  $3.2\times$ .

Table 23: Temporal and spatial resolutions for different X3D model configurations.

Model	Num. Frames	Frame Rate	Num. Pixels
X3D-XS	4	12	$160^2$
X3D-S	13	6	$160^2$
X3D-M	16	5	$224^2$
X3D-L	16	5	$312^2$

For 1-clip testing, a single clip in time is sampled from a spatial center-crop of the video, and the accuracy for that clip in isolation is reported. This corresponds to the inference that would take place in a real-time robotics application, if latency tolerance is low. For 10-clip

testing, ten clips are uniformly sampled in time from the video, processed individually, and finally averaged over their softmax scores to produce a video-level results. For 30-clip testing, ten temporal locations are samples, each with three spatial clips over the horizontal axis (left, center, and right). For offline applications where efficiency is of lesser importance or if a higher latency is tolerated in the online setting, 10- and 30-clip inference can be used to achieve higher accuracy. An overview of the achieved test accuracy on the Kinetics400 dataset for the considered test regimens and models is presented in Table 24.

Table 24: Kinetics400 test top-1 (top-5) accuracy % for various X3D model configurations with a varying number of clips for video classification. For 1-clip testing, a single clip was randomly sampled for each video. For the 10-clip testing, 10 clips were uniformly selected in time from a video. For 30-clip testing, 10 temporal locations were also selected, and a clip was extracted from the left-most, central, and right-most spatial regions.

Model	1-clip	10-clip	30-clip
X3D-XS	55.73 (78.47)	66.31 (86.94)	67.08 (87.28)
X3D-S	61.25 (82.91)	70.29 (89.41)	70.91 (89.72)
X3D-M	65.58 (85.46)	73.66 (91.51)	74.48 (91.67)
X3D-L	67.00 (86.33)	75.21 (92.19)	75.78 (92.47)

To benchmark the efficiency for the different X3D models, a batch of random data with the shape of a clip was generated on the CPU of each platform. The clips-per-second were measured as the time it took to transfer the data to the GPU (if applicable), perform the inference using the model, and to transfer it back to the CPU. For CPU testing, a single core was used on a MacBook Pro (16-inch, 2019) with a 2.6 GHz i7 processor, alongside a batch size of 1. On the Nvidia Jetson TX2, a batch size of 32 was used. The X3D-L configuration was not tested on TX2 due to memory limitations. For testing on Nvidia Jetson Xavier and the Nvidia RTX 2080 Ti graphics card, a batch size of 64 was used for all models but X3D-L which used a batch size of 32. The calculation of floating point operations per second (FLOPS) was likewise computed for a single clip, and is independent of the underlying hardware, as is the parameter count.

Table 25: Comparison of inference speed and model size for different X3D configurations. For measurements on CPU, TX2, Xavier and RTX 2080 Ti, the clips per second (cps)  $\pm$  standard deviation are reported. In comparison of the models, it should be noted that there is a significant different in clip size between the models (see Table 23).

Model	CPU (cps)	TX2 (cps)	Xavier (cps)	RTX 2080 Ti (cps)	FLOPS (G)	Params (M)
X3D-XS	8.26 $\pm$ 0.11	8.20 $\pm$ 0.09	26.37 $\pm$ 0.03	430.15 $\pm$ 9.29	0.61	3.79
X3D-S	2.23 $\pm$ 0.11	2.68 $\pm$ 0.01	8.07 $\pm$ 0.12	138.04 $\pm$ 1.69	1.96	3.79
X3D-M	0.83 $\pm$ 0.04	1.47 $\pm$ 0.003	3.69 $\pm$ 0.02	55.27 $\pm$ 0.67	4.73	3.79
X3D-L	0.25 $\pm$ 0.01	N/A	0.88 $\pm$ 0.003	16.58 $\pm$ 0.13	18.37	6.15

The results of the benchmark are presented in Table 25, where it can be seen that super real-time performance can be achieved without issue on the RTX 2080 Ti for all but the X3D-L model. Among embedded devices, only Xavier with the X3D-XS surpasses 25 clips per second. However, it should be noted that many frames are processed within each clip c.f. Table 23, giving rise to many redundant computations if applied directly to the online setting. While the

current 3D convolutional architectures are not able to compute a prediction output per frame, the conversion of the clip-wise computation to a frame-wise recurrent computation would enable both CPU and TX2 to perform inference using X3D-S at super real-time (in the best case for X3D-S on CPU we have  $2.23 \text{ frames/sec} \times 13 \text{ frames/clip} = 29.0 \text{ frames/sec}$ ). This is an active area of our research and constitutes future work in this direction.

## 3.2 Temporal Attention-Augmented Graph Convolutional Network for Efficient Skeleton-Based Human Action Recognition

### 3.2.1 Introduction and objectives

Skeleton-based human activity recognition has been a very popular research topic in recent years. Compared to other data modalities such as RGB videos, depth images and optical flow, Human body skeleton is invariant to illumination variations, human appearance, view-point variations, context noise and body scale [68]. Besides it encodes high level information representing the human pose and joints motions in a compact graph structure. Many deep learning methods have been recently proposed to model both the spatial and temporal evolution in a sequence of body skeletons and among them, Graph Convolutional Networks (GCNs) have reached significant performance when applied to skeleton-based human activity recognition task [178, 221, 157, 191, 117, 176, 57]. However, most of the recently proposed methods suffer from long training process and heavy computational complexity. Considering that these methods process all the body skeletons in a sequence depicting the performed action, these approaches is not efficient in terms of memory consumption and computation time.

Therefore, minimizing the number of floating point operations (FLOPs) by processing a fewer number of body skeletons in a sequence is a large step towards increasing the computational efficiency in both training and inference processes of human activity recognition task. For each action class, sufficient information can be extracted by focusing on the skeletons of a subset of body poses which are the most informative for that action. We proposed a trainable temporal attention module in a GCN-based model which measures the importance of each skeleton in a sequence and selects a subset of them in the early layers of the network. This module is employed in a GCN-based spatio-temporal network to increase the models' efficiency. The experimental results confirm that our method performs on par with the state-of-the-art methods while increasing computational efficiency by a factor of up to  $\times 9.6$  and it makes it suitable for real time human activity recognition.

A summary of this work is provided hereafter. The corresponding publication is listed below, and can be found in Appendix 7.7:

1. [78] N. Heidari, and A. Iosifidis, “*Temporal Attention-Augmented Graph Convolutional Network for Efficient Skeleton-Based Human Action Recognition*”, International Conference on Pattern Recognition, 2020

### 3.2.2 Summary of state of the art

Skeleton-based human activity recognition methods employing deep learning model are mainly categorized into RNN-based, CNN-based and GCN-based methods. In RNN-based methods [49, 121, 174, 182, 230, 116], the sequence of skeletons are introduced into the model as sequence of vectors which are formed by concatenating 3D coordinates of all body joints of skeletons, and CNN-based methods [120, 100, 98, 122, 111, 112], re-organize the joints' coordinates

into a 2D map and employ a state-of-the-art CNN model to extract temporal and spatial features. Since all these RNN-based and CNN-based methods convert the skeletons into a regular grid or a sequence, they cannot benefit from the non-Euclidean structure of skeleton data. The recently proposed GCN-based methods have achieved promising performance in skeleton-based human activity recognition. Their successful performance can be explained by their ability in capturing the embedded features in non-Euclidean data structures and treat the skeleton data as a graph which represents the body joints (graph nodes) and the natural connections (graph edges) between them. Spatio-temporal graph convolutional network (ST-GCN) [221] is the first GCN-based method proposed for action recognition and is our baseline. It receives the sequence of body skeletons directly as input and employs the GCNs' aggregation rule to extract the spatial features of each skeleton in a sequence, while the temporal dynamics are modeled by using a temporal graph with fixed connections. 2s-AGCN [178] is one of the state-of-the-art methods proposed on top of ST-GCN and adaptively updates the graph structure of the skeleton in each layer by learning a spatial attention mask and a data-dependant graph in an end-to-end manner. Besides, they train two models with the same topology for two different data streams to utilize both joint and bone features and fuse the SoftMax scores of two models. DGNN [176] represents the skeletons as directed acyclic graphs to benefit from the relationship between joints and bones based on their kinematic dependency. It also utilizes the motion data in addition to joints and bones and trains 4 models with different data streams. AS-GCN [117] method captures the richer action-specific correlations and structural links between the skeleton joints by proposing an inference module based on encoder-decoder structure. GCN-NAS [157] is a neural architecture search method which uses ST-GCN model as baseline and explores the search space to determine the best graph structure at each layer of the network. The most similar work to our method is DPRL+GCNN [191] which aims to select the most representative skeletons from the input sequence using deep reinforcement learning. It adjusts the chosen skeletons progressively by evaluating the models' performance in activity recognition task.

### 3.2.3 Description of work performed so far

An undirected spatio-temporal graph on a sequence of skeletons is denoted as  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , where the set of nodes  $\mathcal{V}$  indicates 2D or 3D coordinates of  $N$  body joints of a skeleton in a sequence of  $T$  time steps and  $\mathcal{E}$  is the set of spatial (intra-skeleton) and temporal (inter-skeleton) connections. The graph structure is captured by the adjacency matrix  $\mathbf{A} \in \mathbb{R}^{N \times N}$  which is a symmetric binary matrix. Figure 13 (right) shows the spatio-temporal graph on a sequence of skeletons.

According to the baseline method ST-GCN [221], and its extensions like 2s-AGCN, the model receives the feature tensor  $\mathbf{X} \in \mathbb{R}^{C_{in} \times T \times N}$  as input, where  $C_{in}$  denotes the number of input channels,  $T$  is the number of skeletons and  $N$  is the number of body joints in each skeleton and updates the nodes' feature vectors by applying the spatial convolution to produce  $\mathbf{X}' \in \mathbb{R}^{C_{out} \times T \times N}$  with  $C_{out}$  channels as follows:

$$\mathbf{X}' = ReLU \left( \sum_p (\hat{\mathbf{A}}_p + \mathbf{M}_p) \mathbf{X} \mathbf{W}_p \right), \quad (52)$$

According to the spatial partitioning process [221], each node has 3 subsets of neighbors, which is illustrated in Figure 13 (left) where the nodes in different neighboring subsets (partitions) are shown with different colors and the center of gravity is shown as a red dot. Based on this spatial partition process, the adjacency matrix  $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N = \sum_p \mathbf{A}_p$  is defined as the summation of 3

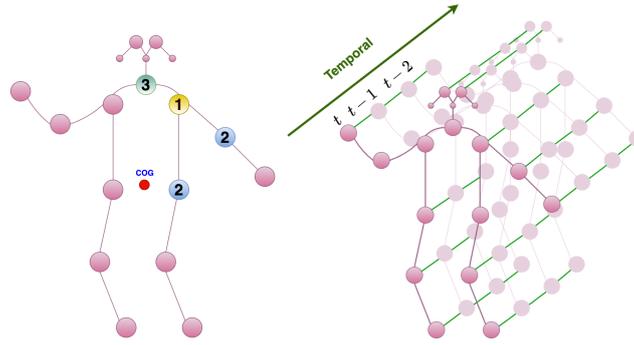


Figure 13: Illustration of an example human body poses encoded as a Spatio-temporal graph (right), and the neighboring subsets in spatial partitioning process in different colors (left).

different adjacency matrices which are indexed by  $p$  and  $\mathbf{A}_0 = \mathbf{I}_N$  represents the nodes' self connections. The normalized adjacency matrix for each subset is defined as:

$$\hat{\mathbf{A}}_p = \mathbf{D}_p^{-\frac{1}{2}} \mathbf{A}_p \mathbf{D}_p^{-\frac{1}{2}}, \quad (53)$$

where  $\mathbf{D}_p$  denotes the degree matrix and  $\mathbf{M}_p \in \mathbb{R}^{N \times N}$  is a learnable attention map which highlights the elements of each adjacency matrix.  $\mathbf{W}_p \in \mathbb{R}^{C_{out} \times C_{in}}$  denotes the weight matrix which transforms the node features of each partition.

To capture the temporal dynamics in a skeleton sequence and consider the motions taking place in an action, a temporal convolution is applied on the output tensor of the spatial convolution step. In order to extract discriminative features in temporal dimension of data, we proposed TAM which highlights the most informative skeletons in a sequence. Based on this process, we apply skeleton selection to highly reduce the overall computational cost both during training and inference.

The TAM takes a tensor  $\mathbf{H}^{(l)} \in \mathbb{R}^{C_l \times T \times N}$  as input, which can be the input data, i.e.  $\mathbf{H}^{(0)} = \mathbf{X}$ , or the output of the  $l^{th}$  hidden layer of the network. First, two average pooling operations in both feature and spatial dimensions is performed to produce the  $\mathbf{h}^{(l)} \in \mathbb{R}^{1 \times T \times 1}$ , which denotes the average feature value of each skeleton in the sequence. Then, this tensor is introduced to a fully connected layer which transforms features in temporal dimension  $T$  to produce the attention tensor  $\mathbf{a} \in \mathbb{R}^{1 \times T \times 1} = \{a_0, a_1, \dots, a_T\}$  as follows:

$$\mathbf{a} = \text{Sigmoid} \left( \mathbf{h}^{(l)} \Theta \right), \quad (54)$$

where  $\Theta \in \mathbb{R}^{T \times T}$  denotes the learnable transformation matrix, and the resulted attention tensor  $\mathbf{a}$  indicates the importance of each skeleton in the sequence. To highlight the most informative skeletons, we create the attention tensor  $\Lambda \in \mathbb{R}^{C_l \times T \times N}$ , a duplicated version of attention map  $\mathbf{a}$  with  $C_l \times N$  copies, which is subsequently dot multiplied to  $\mathbf{H}^{(l)}$  as follows:

$$\hat{\mathbf{H}}^{(l)} = \text{ReLU}(\mathbf{H}^{(l)} \otimes \Lambda), \quad (55)$$

where  $\otimes$  denotes element-wise multiplication. To select a subset of  $T'$  skeletons from  $\hat{\mathbf{H}}^{(l)}$ , the values in the attention map  $\mathbf{a}$  are sorted in descending order and the skeletons corresponding to the  $T'$  highest attention values are selected to be introduced into next layers of the network. To improve the computational efficiency of model in both training and inference phases, it would be preferable to use the TAM in the early layers of the network so that the next layers will process less number of skeletons.

Table 26: Comparisons of the classification accuracy with state-of-the-art methods on the test set of NTU-RGB+D dataset

Method	CS(%)	CV(%)	#Streams
ST-GCN [221]	81.5	88.3	1
DPRL+GCNN [191]	83.5	89.8	1
AS-GCN [117]	86.8	94.2	2
2s-AGCN [178]	88.5	95.1	2
GCN-NAS [157]	89.4	95.7	2
DGNN [176]	89.9	96.1	4
<b>TA-GCN (<math>T' = 150</math>)</b>	87.7	94.2	1
<b>2s-TA-GCN (<math>T' = 150</math>)</b>	88.5	95.1	2
<b>4s-TA-GCN (<math>T' = 150</math>)</b>	89.91	95.8	4

### 3.2.4 Performance evaluation

The performance of the proposed method is evaluated on the NTU-RGB+D [174] and Kinetics-Skeletons [97] datasets. The results for the NTU-RGB+D dataset in Table 29 show that the proposed method, TA-GCN outperforms ST-GCN, which is the baseline in GCN-based methods, and DPRL+GCNN by a large margin in both CV and CS benchmarks. Compared to AS-GCN, the proposed method achieves higher accuracy in CS benchmark and a similar performance in CV benchmark. The only competing GCN-based method that performs skeleton selection, i.e. DPRL+GCNN [191], performs poorly compared to all variants of the proposed method. For Kinetics-Skeleton dataset (Table 30), the proposed method outperforms ST-GCN and AS-GCN methods and it has competitive performance with 2s-AGCN. When the proposed model is trained using two or four different data streams like joints and bones, joint-motion and bone-motion (2s-TA-GCN, 4s-TA-GCN), it achieves competitive or better performance compared to state-of-the-arts like DGNN, 2s-AGCN and GCN-NAS with around 4.8 times less computational complexity.

Table. 31 shows the computational complexity comparison in terms of FLOPs and model parameters (Params) between the GCN-based competing methods on NTU-RGB+D (CV) dataset. The results show that not only does our method outperform the baseline ST-GCN in terms of classification accuracy, but also it has 2.9 times less FLOPs and 1.3 times less number of parameters. Therefore, TA-GCN can be a strong and efficient baseline for GCN-based human action recognition which achieves better performance than ST-GCN. In comparison with state-of-the-art, our proposed TA-GCN achieves good performance with much less computational complexity which makes TA-GCN applicable for many practical tasks with limited computational capacity.

## 3.3 Progressive Spatio-Temporal Graph Convolutional Network for Skeleton-Based Human Action Recognition

### 3.3.1 Introduction, objectives and summary of state of the art

Many Graph Convolutional Network (GCN)-based methods have been proposed recently for skeleton-based human action recognition and they have achieved very good performance. However, most of the state-of-the-art methods in this area have high computational complexity and it is not feasible to employ them in restricted computation applications. In order to make these methods more efficient in terms of computational complexity and memory usage, one solution

Table 27: Comparisons of the classification accuracy with state-of-the-art methods on the test set of Kinetics-Skeleton dataset

Method	Top1(%)	Top5(%)	#Streams
ST-GCN [221]	30.7	52.8	1
AS-GCN [117]	34.8	56.5	2
2s-AGCN [178]	36.1	58.7	2
DGNN [176]	36.9	59.6	4
GCN-NAS [157]	37.1	60.1	2
1s-TA-GCN ( $T' = 250$ )	34.95	57.28	1
2s-TA-GCN ( $T' = 250$ )	36.1	58.72	2
4s-TA-GCN ( $T' = 250$ )	36.9	59.77	4

Table 28: Comparisons of the computational complexity with state-of-the-art methods on the on CV benchmark of NTU-RGB+D dataset. The number of FLOPs and Params for 1s-TA-GCN ( $T' = 150$ ) are 5.64G and 2.24M, respectively.

Method	FLOPs	# Params
1s-TA-GCN ( $T' = 150$ )	$\times 1$	$\times 1$
2s-TA-GCN ( $T' = 150$ )	$\times 2$	$\times 2$
4s-TA-GCN ( $T' = 150$ )	$\times 4$	$\times 4$
ST-GCN	$\times 2.9$	$\times 1.3$
AS-GCN	$\times 6.3$	$\times 3.2$
2s-AGCN	$\times 6.6$	$\times 3$
DGNN	$\times 12.6$	$\times 3.6$
GCN-NAS	$\times 19.3$	$\times 8.9$

is to find an optimized and compact network topology for the model which is able to reach the state-of-the-art performance with less number of parameters and floating point operations. The neural architecture search methods [241, 200] have been proposed for other types of deep learning models like MLPs and they are able to find the optimized network topology for the target model by exploring a huge search space. For ST-GCN methods, the only proposed NAS-based method which is also an state-of-the-art method in this area is GCN-NAS [157]. However, GCN-NAS uses ST-GCN model, with a fixed number of layers and channel sizes in each layer, as baseline and explores the search space to determine the best graph structure at each layer of the network. Therefore, the existing NAS method for ST-GCN is not able to find the optimized network topology in terms of width and depth and still an extensive set of experiments are needed to train different model architectures and select the one with best performance.

In this work, we proposed a Neural Architecture Search method for ST-GCN which finds a compact and problem-specific topology for the network by progressively growing the network topology and at the same time, optimizing the model parameters on a objective function defined on a non-Euclidean data structure. According to the experimental results, the proposed method performs on par with state-of-the-art methods while it builds a compact network topology for ST-GCN which has 15 times less number of parameters compared to the existing methods and therefor, it leads much faster inference process than the competing methods.

A summary of this work is provided hereafter. The corresponding preprint is listed below, and can be found in Appendix 7.8:

- [77] N. Heidari, and A. Iosifidis, “*Progressive Spatio-Temporal Graph Convolutional Network for Skeleton-Based Human Action Recognition*”, arXiv:2011.05668, 2020

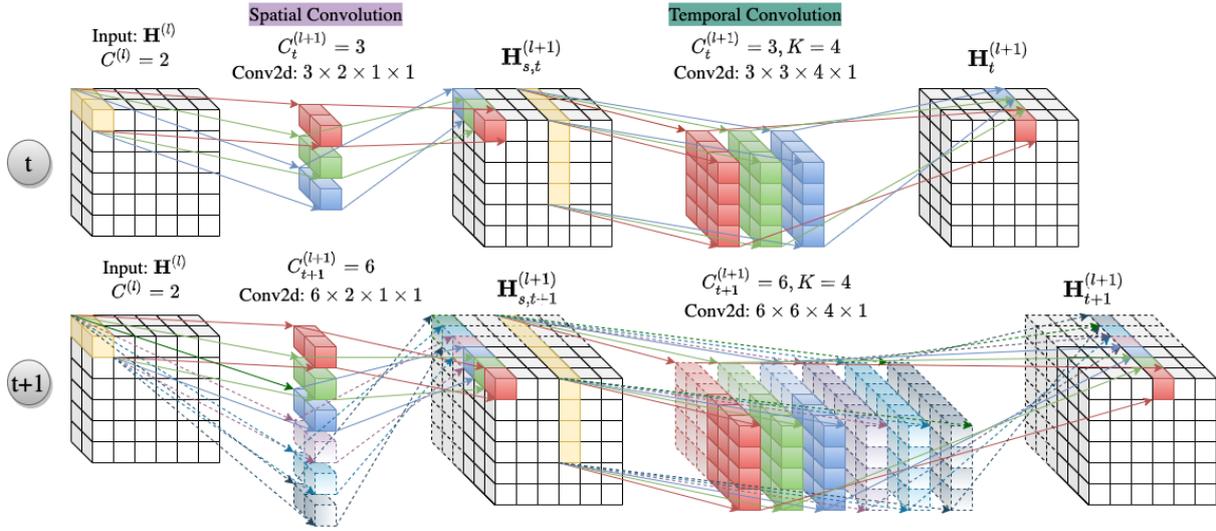


Figure 14: Illustration of the proposed ST-GCN-AM building the model in layer  $l$  at iteration  $t$  and  $t + 1$ . It is assumed that  $C^{(l)} = 2$ ,  $S = 3$  and  $K = 4$ . Conv2d is the abbreviation for standard 2D convolutional operation.

### 3.3.2 Description of work performed so far

To learn a problem-dependant topology for ST-GCN, we proposed a ST-GCN augmentation module (ST-GCN-AM) which builds each layer of the ST-GCN model by increasing the receptive field in spatial and temporal convolutions progressively. The progressive ST-GCN algorithm (PST-GCN) starts with an empty network and returns the ST-GCN model with an optimized topology in terms of both width and depth by employing the proposed ST-GCN-AM.

Let us assume the ST-GCN model is already formed by  $l$  layers. ST-GCN-AM builds a new layer by receiving the existing model and its output  $\mathbf{H}^{(l)} \in \mathbb{R}^{C^{(l)} \times T^{(l)} \times V}$  as input and applies an iterative training process (indexed by  $t$ ). In the training process, the receptive fields in spatial and temporal convolutions are increased progressively. Inspired by 2s-AGCN [178], the spatial convolution is defined as follows:

$$\mathbf{H}_{s,t}^{(l+1)} = \text{ReLU} \left( \sum_p (\hat{\mathbf{A}}_p + \mathbf{M}_{p,t}^{(l+1)}) \mathbf{H}^{(l)} \mathbf{W}_{p,t}^{(l+1)} \right). \quad (56)$$

The attention mask  $\mathbf{M}_{p,t}^{(l+1)}$  which is added to the normalized adjacency matrix  $\hat{\mathbf{A}}_p$ , both highlights the existing connections between the joints and adds potentially important connections between the disconnected nodes. In practice, the spatial convolution is a standard 2D convolutional operation which performs 2D convolutions with  $C^{(l+1)}$  different filters each of size  $C^{(l)} \times 1 \times 1$  on  $\mathbf{H}^{(l)}$  to map the features of the  $l^{\text{th}}$  layer into a new space with  $C^{(l+1)}$  channels. At the iteration  $t = 1$ , we set  $C_t^{(l+1)} = S$ , where  $S$  is a hyper-parameter of the method. The spatial convolution is followed by a batch normalization layer and a ReLU activation function. In the next step, the temporal convolution captures the temporal dynamics through the sequence of skeletons by receiving  $\mathbf{H}_{s,t}^{(l+1)}$  as input and performing standard 2D convolution with  $C_t^{(l+1)}$  different filters, each of size  $C_t^{(l+1)} \times K \times 1$ , to aggregate the features through  $K$  consecutive skeletons. The temporal convolution is also followed by a batch normalization layer, residual connection. Each layer has a residual connection to stabilize the model and its output is fol-

lowed by a ReLU activation function. A global average pooling (GAP) operation is applied on the extracted features  $\mathbf{H}_t^{(l+1)}$  and the produced feature vectors are passed to a fully connected classification layer. The model is trained using back-propagation and the classification accuracy  $\mathcal{A}_t^{(l+1)}$  on validation data is recorded.

At iteration  $t > 1$ , the number of output channels in the current layer are increased by  $S$  in order to grow the layer's width. Accordingly, the filter size in all the 2D convolutional operations (spatial, temporal and residual) is increased. The spatial convolution and the residual connection perform  $C_t^{(l+1)}$  convolutional filters of size  $C^{(l)} \times 1 \times 1$  on  $\mathbf{H}^{(l)}$  so that the first  $(t-1)S$  filters are initialized by the learned parameters at iteration  $t-1$  and the second newly added  $S$  filters are initialized randomly. The temporal convolution performs  $C_t^{(l+1)}$  convolutional filters of size  $C_t^{(l+1)} \times 1 \times 1$  on the output of spatial convolution. Figure. 14 illustrates the width progression in each layer of the ST-GCN performed by the ST-GCN-AM and it can be seen that both the width and length of the filters are increased in temporal convolution. The first  $(t-1)S$  filters of size  $(t-1)S \times 1 \times 1$  are initialized using the learned parameters at iteration  $t-1$  (illustrated with solid structure) and the newly added parameters are initialized randomly (illustrated with dotted structure). In a similar way, the first  $(t-1)S \times N_{class}$  parameters of the classification layer are initialized using the finetuned parameters in the previous iteration and the remaining parameters are initialized randomly. At each iteration, the attention matrix  $\mathbf{M}_{p,t}^{(l+1)}$  is fully initialized by its finetuned values obtained at iteration  $t-1$ .

In order to evaluate the width progression of the  $(l+1)^{th}$  layer, the model parameters are fine-tuned by back-propagation and the new classification accuracy  $\mathcal{A}_t^{(l+1)}$  of the model on validation data is recorded. The performance improvement rate, i.e.  $\alpha_w = (\mathcal{A}_t^{(l+1)} - \mathcal{A}_{t-1}^{(l+1)}) / \mathcal{A}_{t-1}^{(l+1)}$  shows whether the width progression improved the performance of the model or not, i.e.  $\alpha_w < \varepsilon_w$  with  $\varepsilon_w > 0$ . If the performance is improved, the parameters are saved and the next iteration augmenting the layer by adding  $S$  new channels starts. If the width progression doesn't improve the model's performance, the newly added parameters are removed and all the model parameters are set to those obtained at iteration  $t-1$ . At this point, the width progression for the  $(l+1)^{th}$  layer stops.

Let us assume that the width progression is terminated at  $(l+1)^{th}$  layer. In order to evaluate the depth progression of the network, the model's performance with  $l+1$  layers is recorded and the rate of improvement is given by  $\alpha_d = (\mathcal{A}^{(l+1)} - \mathcal{A}^{(l)}) / \mathcal{A}^{(l)}$ , where  $\mathcal{A}^{(l)}$ ,  $\mathcal{A}^{(l+1)}$  denote the model's performance before and after adding the  $(l+1)^{th}$  layer to the model, respectively. If the addition of the last layer does not improve the model's performance, i.e. when  $\alpha_d < \varepsilon_d$  with  $\varepsilon_d > 0$ , the newly added layer is removed and the algorithm stops growing the networks' topology. As the final step, the model is fine-tuned using both training and validation sets.

### 3.3.3 Performance evaluation

The performance of the proposed method is evaluated by conducting experiments on the NTU-RGB+D [174] and Kinetics-Skeletons [97] datasets. Tables 29 and 30 show the comparison of the proposed method with the state-of-the-art methods in terms of classification accuracy on NTU-RGB+D and Kinetics-Skeleton datasets, respectively. Besides, Table.31 shows the comparison of computational complexity of our method with GCN-based methods in terms of floating point operations (FLOPs) and model parameters. Since the source code of DPRL+GCN is not provided by the authors, its computational complexity is not reported in Table 31. 2s-PST-GCN shows the performance of our method when it is trained with two data streams, joints

Table 29: Comparisons of the classification accuracy with state-of-the-art methods on the test set of NTU-RGB+D dataset

Method	CS(%)	CV(%)	#Streams
ST-GCN [221]	81.5	88.3	1
DPRL+GCNN [191]	83.5	89.8	1
AS-GCN [117]	86.8	94.2	2
2s-AGCN [178]	88.5	95.1	2
1s-TA-GCN [78]	87.97	94.2	1
2s-TA-GCN [78]	88.5	95.1	2
GCN-NAS [157]	89.4	95.7	2
<b>PST-GCN</b>	87.9	94.33	1
<b>2s-PST-GCN</b>	88.68	95.1	2

Table 30: Comparisons of the classification accuracy with state-of-the-art methods on the test set of Kinetics-Skeleton dataset

Method	Top1(%)	Top5(%)	#Streams
Deep LSTM [174]	16.4	35.3	1
TCN [100]	20.3	40.0	1
ST-GCN [221]	30.7	52.8	1
AS-GCN [117]	34.8	56.5	2
2s-AGCN [178]	36.1	58.7	2
1s-TA-GCN [78]	34.95	57.28	1
2s-TA-GCN [78]	36.1	58.72	2
GCN-NAS [157]	37.1	60.1	2
<b>PST-GCN</b>	34.71	57.08	1
<b>2s-PST-GCN</b>	35.53	58.2	2

and bones, and it is compared with other competing methods which utilize two different data streams such as 2s-AGCN, AS-GCN and GCN-NAS.

The optimized network topologies found by PST-GCN for NTU-RGB+D dataset is a 8 layer network with channel sizes (100, 80, 100, 40, 60, 80, 60, 80), (100, 80, 60, 100, 60, 100, 140, 80) for CV and CS benchmarks, respectively. The network topology found for Kinetics-skeleton dataset has 9 layers of sizes (80, 100, 100, 120, 100, 120, 140, 160, 180), respectively. The network topology which is used in ST-GCN, 2s-AGCN and GCN-NAS methods, compromise of 10 ST-GCN layers with fixed channel sizes (64, 64, 64, 64, 128, 128, 128, 256, 256, 256). The results show that in most cases, the proposed method performs on par with state-of-the-art methods while it has less number of parameters and FLOPs which makes the inference process much faster.

### 3.4 Spatio-Temporal Bilinear Network for Skeleton-Based Human Action Recognition

#### 3.4.1 Introduction, objectives and summary of state of the art

The recently proposed GCN-based methods which have been very successful in skeleton-based human action recognition [221, 176, 178, 157] represent the sequence of skeletons as a spatio-temporal graph to encode both utilize the graph structure of the skeleton data to capture both the spatial structure of human body joints with their natural connections and the temporal dynamics of each action. The baseline method ST-GCN [221] receives a sequence of skeletons as input

Table 31: Comparisons of the computational complexity with GCN-based state-of-the-art methods.

Methods	#G FLOPs	#M Params
ST-GCN [221]	16.7	3.12
AS-GCN [117]	35.92	7.24
2s-AGCN [178]	37.32	6.9
1s-TA-GCN (NTU-CV-CS) [78]	5.64	2.24
2s-TA-GCN (NTU-CV-CS) [78]	11.28	4.48
1s-TA-GCN (Kinetics) [78]	9.17	2.24
2s-TA-GCN (Kinetics) [78]	18.34	4.48
GCN-NAS [157]	109.26	20.13
<b>PST-GCN (NTU-CV)</b>	7.2	0.63
<b>2s-PST-GCN (NTU-CV)</b>	14.4	1.26
<b>PST-GCN (NTU-CS)</b>	9.57	0.92
<b>2s-PST-GCN (NTU-CS)</b>	19.14	1.84
<b>PST-GCN (Kinetics)</b>	5.67	1.96
<b>2s-PST-GCN (Kinetics)</b>	11.34	3.92

and employs GCN layers to capture both spatial and temporal features in actions by exploiting the graph structure of input data. Moreover, it employs an attention mask to highlight the most informative connections between the body joints in the spatio-temporal graph for the action prediction. Several methods, like 2s-AGCN [178], GCN-NAS [157] and DGNN [176], are proposed to improve the performance of the ST-GCN method by learning the graph structure in each GCN layer adaptively in an end-to-end manner, based on the features of input data [178, 157, 176].

In this work, the attention mechanism in existing methods built on top of ST-GCN method is analyzed and based on this analysis we argue that the attention mechanism in GCN layers using an additive formulation should lead to a symmetric attention mask to reflect the symmetric relationship of the corresponding human body joint positions in a human body pose. Accordingly, we propose a symmetric attention mechanism that is shown to perform on par with the original one for different structures of the model. Besides, we propose a spatio-temporal bilinear layer which allows for more flexible design of the model for skeleton-based human action recognition. We show that models with spatio-temporal bilinear layers perform on par with spatio-temporal GCN networks without requiring the design of graph structures to encode relationships of the joints of the body skeletons.

A summary of this work is provided hereafter. The corresponding preprint is listed below, and can be found in Appendix 7.9:

1. [76] N. Heidari, and A. Iosifidis, “*On the spatial attention in Spatio-Temporal Graph Convolutional Networks for skeleton-based human action recognition*”, arXiv:2011.03833, 2020

### 3.4.2 Description of work performed so far

One of the main drawbacks of ST-GCN method which is addressed by more recently proposed methods, such as 2s-AGCN [178], is that it uses a fixed graph structure which is heuristically predefined and represents the natural physical connections between the body joints in a skeleton. In ST-GCN method, each GCN layer updates the human body features through both spatial and

temporal domains by applying spatial and temporal convolutions on the output of the previous layer. The spatial convolution in this method is defined as follows:

$$\mathbf{H}_s^{(l)} = \text{ReLU} \left( \sum_p \left( \hat{\mathbf{A}}_p \otimes \mathbf{M}_p^{(l)} \right) \mathbf{H}^{(l-1)} \mathbf{W}_p^{(l)} \right), \quad (57)$$

where  $\mathbf{H}^{(l-1)} \in \mathbb{R}^{C^{(l-1)} \times T^{(l-1)} \times V}$  denotes the output of the  $(l-1)^{th}$  layer and the input of first layer is defined as  $\mathbf{H}^0 = \mathbf{X}$ . The number of human body joints in each skeleton is  $V$ . Based on a spatial partitioning process, the neighboring nodes of each body joint are partitioned into  $p = 3$  subsets and the GCN's propagation rule is applied on each graph with a different weight matrix  $\mathbf{W}_p^{(l)}$ , where  $C^{(l-1)}$  and  $C^{(l)}$  denote the number of channels in layers  $l-1$  and  $l$ , respectively.  $\mathbf{M}_p^{(l)} \in \mathbb{R}^{V \times V}$  is a learnable attention matrix which highlights the most important connections in a skeleton for each action. It is initialized as an all-one matrix and is element-wise multiplied to its corresponding normalized adjacency matrix  $\hat{\mathbf{A}}_p$  of each graph partition. Therefore, the attention mechanism in ST-GCN method can only highlight or diminish existing connections between the body joints which is not guaranteed to be optimal for action classification task. For some actions such as "touching head", the connection between the hand and head is important for recognizing the action, while this connection does not exist naturally in the body and, thus, is not considered in the predefined graph structure.

The 2s-AGCN method addressed this issue by defining the spatial convolution as follows:

$$\mathbf{H}_s^{(l)} = \text{ReLU} \left( \sum_p \left( \hat{\mathbf{A}}_p + \mathbf{M}_p^{(l)} \right) \mathbf{H}^{(l-1)} \mathbf{W}_p^{(l)} \right), \quad (58)$$

where  $\mathbf{M}_p^{(l)}$  is a learnable attention matrix added to the graph in order to both highlight/diminish existing connections between the skeleton joints and also add potentially important connections between the disconnected ones. This matrix is initialized by zeros and it is learned in an end-to-end manner along with the other model's parameters.

Considering that the nodes of the graph correspond to human body skeleton joints and during action execution their relative positions are symmetric, we would expect that their pairwise attentions should be the same. However, the final form of the attention matrix  $\mathbf{M}_p^{(l)}$  will be an asymmetric matrix containing both positive and negative values because it is optimized in an end-to-end manner without imposing any constraints. In order to enforce this property in the attention mechanism, we defined the attention mask to be a symmetric matrix, i.e.  $\mathbf{M}_p^{(l)} = \mathbf{L}_p^{(l)} \mathbf{L}_p^{(l)T}$ . Accordingly, we define the spatial convolution as follows:

$$\mathbf{H}_s^{(l)} = \text{ReLU} \left( \sum_p \left( \hat{\mathbf{A}}_p + \mathbf{L}_p^{(l)} \mathbf{L}_p^{(l)T} \right) \mathbf{H}^{(l-1)} \mathbf{W}_p^{(l)} \right). \quad (59)$$

Once the training process ends, the matrix  $\mathbf{M}_p^{(l)}$  is calculated and used for inference, thus, the space and time complexities remain the same as in the case of using Eq. (58).

According to the optimization of the attention-based Adjacency matrices defined in Eq. (58), the final form of the attention-based Adjacency matrix depends primarily on the learned values of the mask. In other words, after the first few training epochs, the GCN blocks of the ST-GCN using an additive attention mask do not follow the graph structure anymore, but they

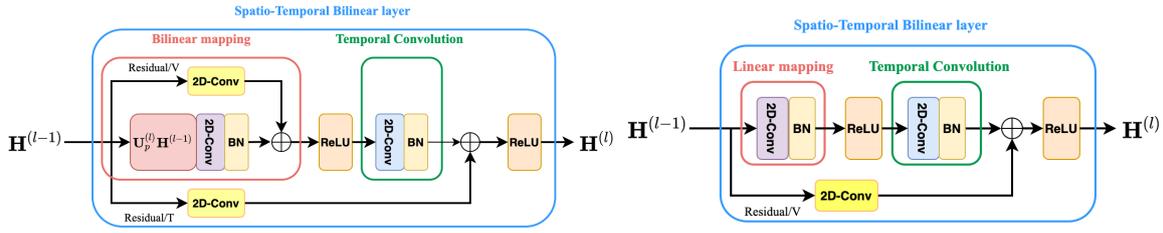


Figure 15: Left: illustration of spatio-temporal bilinear layer  $l$  receiving as input  $\mathbf{H}^{(l-1)}$  of size  $C^{(l-1)} \times T^{(l-1)} \times V^{(l-1)}$  providing an output  $\mathbf{H}^{(l)}$  of size  $C^{(l)} \times T^{(l)} \times V^{(l)}$ . 2D-Conv blocks correspond to standard 2D convolutions. Residual/V indicates the residual connection which adds the input to the output of bilinear mapping, and Residual/T denotes the residual connection which adds the layer’s input to the output of temporal convolution. BN and ReLU represent the batch-normalization and ReLU activation function, respectively. Right: illustration of spatio-temporal bilinear layer  $l$  when  $V^{(l-1)} = V^{(l)} = 1$ .

are rather equivalent to bilinear layers [58, 199, 99]. That is, Eqs. (58) and (59) can take the form:

$$\mathbf{H}_s^{(l)} = \text{ReLU} \left( \sum_p \mathbf{U}_p^{(l)} \mathbf{H}^{(l-1)} \mathbf{W}_p^{(l)} \right), \quad (60)$$

where  $\mathbf{U}_p^{(l)} \in \mathbb{R}^{V^{(l)} \times V^{(l-1)}}$  is a learnable matrix which is optimized in an end-to-end manner jointly with the parameters of the entire network. By using the bilinear layer definition in Eq. (60) one can freely decide on the dimensions of the transformation to be applied using  $\mathbf{U}_p^{(l)}$ . That is, selection of  $V^{(1)} < V$  in the first layer of the Spatio-Temporal Bilinear network will lead to aggregation of joints’ information to create a new set of  $V^{(1)}$  nodes, and selection of  $V^{(1)} > V$  will lead to the creation of new nodes to be processed by the second layer. A similar explanation can be given to the selection of the values  $V^{(l)}$  for layer  $l$ .

A special case in this setting is the one where  $V^{(l)} = 1$ , leading to the creation of one node encoding information of the entire input skeleton data. The structure of a bilinear layer is shown in Figure. 15 (left). If at layer  $l - 1$  a value of  $V^{(l-1)} = 1$  is used, the input  $\mathbf{H}^{(l-1)}$  to layer  $l$  becomes of size  $C^{(l-1)} \times T^{(l-1)} \times 1$ . In that case there is no need of using  $\mathbf{U}_p^{(l)} \in \mathbb{R}$  as the scaling factor that would be learned by using it can be absorbed in  $\mathbf{W}_p^{(l)}$ . Thus, the bilinear mapping can be replaced with an equivalent linear mapping. Moreover, in this case there is no need of using a residual connection for the linear mapping block. The architecture of such a spatio-temporal bilinear layer is shown in Figure 15 (right).

### 3.4.3 Performance evaluation

We conducted two sets of experiments on NTU-RGB+D dataset [174] which is the largest indoor-captured action recognition dataset and it is widely used for evaluating the skeleton-based human action recognition methods. In the first set of experiments, we compared the performance of spatio-temporal networks using GCN layers equipped with attentions in Eqs. (58) and (59) and using bilinear layer as defined in Eq. (60). The performance of the three models is reported in Table 32. As can be seen, the three networks achieved very similar performance. This indicates that there is no difference in using the predefined graph structure encoded by the matrices  $\hat{\mathbf{A}}_p$  in Eqs. (58) and (59) for initializing the mapping in the bilinear layer (60).

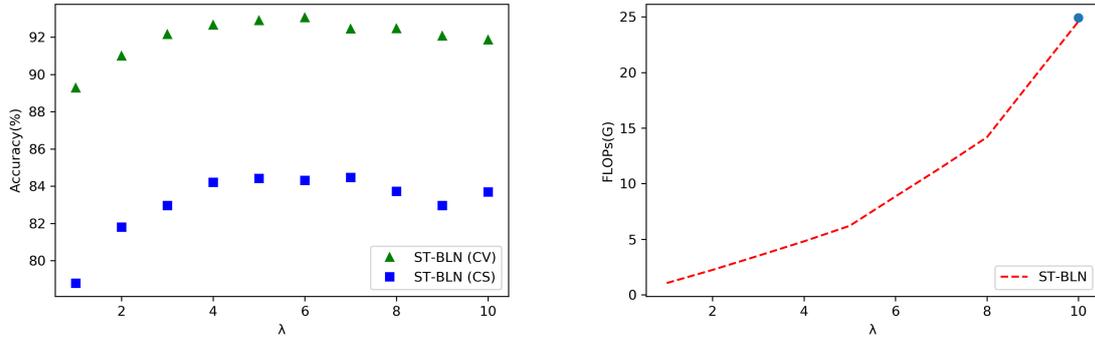


Figure 16: Left: performance of ST-BLN model in terms of classification accuracy, when joints' features are aggregated in different layers  $\lambda$  of the network. Right: Floating Point Operations of the ST-BLN model when joints' features are aggregated in different layers  $\lambda$  of the network. The case of a ST-BLN with  $V^{(l)} = V$ ,  $l = 1, \dots, 10$  is illustrated with a dot.

Table 32: Comparison of classification accuracy of spatio-temporal networks equipped with asymmetric and symmetric spatial attentions, and the bilinear mappings on the test set of CV benchmark of NTU-RGB+D dataset. The models are trained using joints data.

Attention/Mapping	CS(%)	CV(%)
$\hat{\mathbf{A}}_p + \mathbf{M}_p^{(l)}$	85.29	93.78
$\hat{\mathbf{A}}_p + \mathbf{L}_p^{(l)} \mathbf{L}_p^{(l)T}$	87.14	93.86
$\mathbf{U}_p^{(l)}$	85.71	93.80

Table 33: Classification accuracy comparison of spatio-temporal bilinear model with state-of-the-art GCN-based methods on the test set of NTU-RGB+D dataset.

Method	CS(%)	CV(%)	#Streams
ST-GCN [221]	81.5	88.3	1
DPRL+GCNN [191]	83.5	89.8	1
TA-GCN [78]	87.97	94.2	1
AS-GCN [117]	86.8	94.2	2
2s-AGCN [178]	88.5	95.1	2
2s-TA-GCN [78]	88.5	95.1	2
GCN-NAS [157]	89.4	95.7	2
<b>ST-BLN</b>	85.71	93.80	1
<b>2s-ST-BLN</b>	87.8	95.1	2

The performance of state-of-the-art methods in skeleton-based human action recognition on the two benchmarks of the NTU-RGB+D dataset are provided in Table 29. The 2s-ST-BLN model is formed by two streams, the first of which is trained by joint data and the second one is trained by bone data and the predicted SoftMax scores of the two streams are fused to obtain the final classification outcome. The results show that ST-BLN and 2s-ST-BLN perform on par with other state-of-the-art methods while they do not require a predefined graph structure to encode the relationships between the joints (and bones) of the human body skeleton.

To evaluate the need of combining information in the dimension of human body skeleton joints, we conducted a second set of experiments in which we used a 10-layer spatio-temporal bilinear network with  $V^{(l)} = V, l = 1, \dots, \lambda - 1$  and  $V^{(l)} = 1, l = \lambda, \dots, 10$ , instead of using bilinear mappings with  $V^{(l)} = V$  for all 10 layers of the model. That is, the first  $\lambda - 1$  layers of the model use Spatio-Temporal Bilinear layers as the one illustrated in Figure 15 (left) with values  $V^{(l)} = V$ . At layer  $\lambda$  a Spatio-Temporal Bilinear layer in the form of Figure 15 (left) is used with  $V^{(\lambda)} = 1$ , while the remaining layers take the form of a Spatio-Temporal Bilinear layer as the one illustrated in Figure 15 (right). We applied experiments using both the CV and CS benchmarks of the NTU-RGB+D dataset using the joints data, and the results in terms of classification accuracy and number of FLOPs are indicated in Figure 16. As can be seen in Figure 16 (left), the performance values are on par with the results reported in Table 32, achieved by using the Spatio-Temporal Bilinear/GCN layers using  $V^{(l)} = V, l = 1, \dots, 10$ . By observing the number of Floating Point Operations needed to evaluate an input skeleton sequence for different choices of  $\lambda$  in Figure 16 (right), we can see that one can increase efficiency of the model by using lower values of  $\lambda$ . Considering that the number of model's parameters at the deeper layers is higher compared to the first layers, it can be seen that a considerable improvement in the model's efficiency can be achieved. Overall, a network using a value of  $\lambda = 6$  ( $\lambda = 7$ ) operates  $\times 2.78$  ( $\times 2.14$ ) faster compared to a network with  $\lambda = 10$ .

## 3.5 Human Activity Recognition using Recurrent Bag-of-Features

### 3.5.1 Introduction, objective and summary of state-of-the-art

The typical pipeline of a human action recognition approach involves at least the following two steps: a) feature extraction, where low-level information is extracted from small spatial or temporal segments of the data, and b) feature aggregation, where the information extracted in the previous step is aggregated in a representation that can be used for the subsequent tasks, e.g., classification, retrieval, etc. DL unified, to some extent, these two steps by employing deep trainable feature extraction layers, e.g., convolutional layers, which are then used in combination with simple pooling layers, e.g., max or average pooling. These pooling operators are capable of providing some degree of translation invariance, as well as lowering the complexity of the model. Indeed, Convolutional Neural Networks (CNNs) were able to extract discriminative features, that led to extraordinary performance for various complex image analysis tasks [109]. However, at the same time, these naive pooling layers discard crucial spatial and/or temporal information that is carried by the extracted feature vectors. This can lead to significant information loss, especially in cases where a significant part of the information is sequentially distributed over the temporal dimension, reducing the performance of such approaches on tasks that require capturing fine-grained temporal information, such as action recognition, as we will also demonstrate through this Section.

To this end, we proposed a novel stateful recurrent pooling approach that is capable of overcoming these limitations. The proposed method builds upon the well-known Bag-of-Feature (BoF) model [180], which is capable of creating a constant-length representation of multimedia objects, e.g., video [93, 88], audio [144], etc., by generating a histogram over the features extracted from the corresponding object. It is worth noting that despite its remarkable success in various tasks and its ability to handle inputs of variable size, BoF-based methods still lead to loss of valuable spatial and temporal information, which can be crucial for various tasks, such as human action recognition, where capturing the temporal succession of events is often needed for successfully recognizing different activities.

To this end, we propose a powerful stateful trainable recurrent quantization approach that can be used instead of the plain static quantization employed by the traditional BoF formulations. In this way, the proposed method overcomes a critical limitation of existing BoF formulations, by learning *temporal codewords* that capture not only the semantic content of each feature vector, but also the interrelation between them. The employed approach still maintains all the advantages of BoF-based formulations, e.g., invariance to various distribution shifts [149], while at the same time is capable of capturing the temporal information contained in the input data by employing a powerful recurrent quantizer. The proposed method, which is called Recurrent BoF (“ReBoF”), is formulated as a *recurrent* neural layer that is used between the last information analysis layer (e.g., last convolutional layer) and the classification layers of a deep neural network. Therefore, instead of using other naive pooling layers, that can lead to significant loss of temporal information, the extracted feature vectors are quantized to a number of temporal codewords in a recurrent manner, enabling to encode the fine-grained temporal information contained in the original feature vectors. ReBoF is fully differentiable allowing the resulting model to be fully optimized using gradient descent in an end-to-end fashion. To the best of our knowledge, in this method we propose the first stateful recurrent Bag-of-Features model that is capable of effectively modeling the temporal dynamics of video sequences for human action recognition. It is also worth noting that existing BoF formulations, e.g., [149, 150, 167], provide models that cannot effectively encode the temporal information contained in the input

data.

A summary of this work is provided hereafter. The corresponding publications are listed below, and can be found in Appendices 7.5 and 7.6:

1. [107] M. Krestenitis, N. Passalis, A. Iosifidis, M. Gabbouj and A. Tefas, “*Recurrent Bag-of-Features for Visual Information Analysis*”, Pattern Recognition, 2020
2. [106] M. Krestenitis, N. Passalis, A. Iosifidis, M. Gabbouj and A. Tefas, “*Human Action Recognition using Recurrent Bag-of-Features Pooling*”, Proceedings of the International Conference on Pattern Recognition, 2020

### 3.5.2 Description of work performed so far

The proposed method is briefly introduced in this section, along with the used notation. Let  $\mathcal{X} = \{x_i\}_{i=1}^N$  be a set of  $N$  videos to be represented using the standard BoF model. From each video,  $N_i$  feature vectors are extracted:  $\mathbf{x}_{ij} \in \mathbb{R}^D (j = 1, \dots, N_i)$ , where  $D$  is the dimensionality of each feature vector. BoF provides a way to efficiently aggregate these features into a fixed-length histogram. To this end, each feature vector is first quantized into a predefined number of codewords, by employing a codebook  $\mathbf{V} \in \mathbb{R}^{N_K \times D}$ , where  $N_K$  is the number of codewords. Several ways have been proposed to learn the codebook. Perhaps the simplest, yet among the most efficient ones, is to cluster the feature vectors into  $N_K$  clusters [180]. Note that any clustering algorithm, e.g.,  $k$ -means, can be used to this end, with each centroid,  $\mathbf{v}_k \in \mathbb{R}^D (k = 1, \dots, N_K)$ , corresponding to a codeword. Then, the quantized feature vectors are aggregated to extract a histogram that describes the semantic content of the corresponding video. Also, let  $\mathbf{x}_{ij}$  denote the  $j$ -th feature vector extracted from the  $i$ -th video. The histogram extracted using the regular BoF formulation discards any temporal information encoded by the order in which the feature vectors arrive. To overcome this limitation, in this work we propose using a recurrent stateful quantizer, which allows for capturing and effectively encoding the temporal information expressed by the order in which the feature vectors arrive to the model. We assume that the feature vector  $\mathbf{x}_{ij}$  is extracted from the  $j$ -th timestep of the  $i$ -th video sequence.

Before deriving the proposed recurrent quantization approach, it is worth examining, from a probabilistic perspective, the quantization process involved in the BoF model. Using Kernel Density Estimation [18], we can estimate the probability of observing the feature vector  $\mathbf{x}_{ij}$ , given an input object  $x_i$ , as:

$$p(\mathbf{x}_{ij}|x_i) = \sum_{k=1}^{N_K} [\mathbf{s}_i]_k K(\mathbf{x}_{ij}, \mathbf{v}_k) \in [0, 1], \quad (61)$$

where  $K(\cdot)$  the employed kernel function and the histogram  $\mathbf{s}_i \in \mathbb{R}^{N_K}$  actually controls the parameters of the density estimation. These parameters can be then calculated using a maximum likelihood estimator:

$$\mathbf{s}_i = \arg \max_{\mathbf{s}} \sum_{j=1}^{N_i} \log \left( \sum_{k=1}^{N_K} [\mathbf{s}]_k K(\mathbf{x}_{ij}, \mathbf{v}_k) \right). \quad (62)$$

The involved parameters (histogram) can be estimated as  $\mathbf{s}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \mathbf{u}_{ij}$ , as shown in [18], where

$$[\mathbf{u}_{ij}]_k = \frac{K(\mathbf{x}_{ij}, \mathbf{v}_k)}{\sum_{l=1}^{N_K} K(\mathbf{x}_{ij}^{(t)}, \mathbf{v}_l)} \in [0, 1], \quad (63)$$

which leads to the regular soft formulation of BoF, as described in the previous Section. Also, note that we can replace the Gaussian kernel typically used for density estimation, which also requires tuning the width  $\sigma$ , with an easier to use hyperbolic kernel, which does not require tuning any hyper-parameters [152]. The hyperbolic kernel also proved to be stabler and easier to use when the proposed method was combined with deep neural networks. Therefore, we can now measure the similarity between each feature vector and the codewords in order to quantize the feature vectors as:

$$[\mathbf{d}_{ij}]_k = \sigma(\mathbf{x}_{ij}^T \mathbf{v}_k) \in \mathbb{R}, \quad (64)$$

where  $\sigma(x) = \frac{1}{1+\exp(-x)}$  is the logistic sigmoid function.

It is worth noting that temporal information is still discarded in this formulation. To overcome this limitation, we extend (64) in order to take into account the temporal information, as expressed by the histogram extracted until the current step, as:

$$[\mathbf{d}_{ij}]_k = \sigma(\mathbf{V}\mathbf{x}_{ij} + \mathbf{V}_h(\mathbf{r}_{ij} \odot \mathbf{s}_{i,j-1})) \in \mathbb{R}^{N_K}, \quad (65)$$

where  $\mathbf{V}_h \in \mathbb{R}^{N_K \times N_K}$  is a weight matrix that is used to transfer the gated histogram vector into the quantization space,  $\mathbf{s}_{i,j-1}$  is the histogram extracted from previous quantizations (state) and  $\mathbf{r}_{ij} \in \mathbb{R}^{N_K}$  is the output of a reset gate, introduced to ensure the long-term stability of the model. The additional parameters introduced in this formulation are learned during the training process using back-propagation. The proposed method employs a reset gate, inspired by the GRU model [31], to ensure the stability of the model, which is defined as:

$$\mathbf{r}_{ij} = \sigma(\mathbf{W}_r \mathbf{x}_{ij} + \mathbf{U}_r \mathbf{s}_{i,j-1}) \in \mathbb{R}^{N_K}, \quad (66)$$

where  $\mathbf{W}_r \in \mathbb{R}^{N_K \times D}$  and  $\mathbf{U}_r \in \mathbb{R}^{N_K \times N_K}$  denote the weight parameters used by the reset gate.

Then, the  $l_1$  normalized membership vector is computed similarly to the regular BoF model as:

$$\mathbf{u}_{ij} = \frac{\mathbf{d}_{ij}}{\|\mathbf{d}_{ij}\|_1}. \quad (67)$$

Note that the initial state  $\mathbf{s}_{i,0}$  is set to  $\mathbf{s}_{i,0} = \frac{1}{N_K} \mathbf{1}$ , where  $N_K$  refers to the number of codewords, while  $\mathbf{1} \in \mathbb{R}^{N_K}$  denotes a vector of all ones. This ensures that the quantizer's output will be always a properly normalized membership vector. Therefore, the histogram is recurrently updated as:

$$\mathbf{s}_{ij} = (\mathbf{1} - \mathbf{z}_{ij}) \odot \mathbf{s}_{i,j-1} + \mathbf{z}_{ij} \odot \mathbf{u}_{ij} \in \mathbb{R}^{N_K}. \quad (68)$$

The output of the update gate  $\mathbf{z}_{ij}$ , which controls how much the current histogram will be updated, is similarly calculated as:

$$\mathbf{z}_{ij} = \sigma(\mathbf{W}_z \mathbf{x}_{ij} + \mathbf{U}_z \mathbf{s}_{i,j-1}) \in \mathbb{R}^{N_K}, \quad (69)$$

where  $\mathbf{W}_z \in \mathbb{R}^{N_K \times D}$  and  $\mathbf{U}_z \in \mathbb{R}^{N_K \times N_K}$  are weight matrices for the update gate. Finally, to compile the final histogram we average all the intermediate histograms, as before:  $\mathbf{s}_i = \sum_{j=1}^{N_i} \mathbf{s}_{ij} \in \mathbb{R}^{N_K}$ .

### 3.5.3 Performance evaluation

The experimental protocols, along with extensive performance evaluation experiments are presented in the paper provided in Appendix 7.5. Some evaluation results are reported in Table 34 and Table 35, highlighting the effectiveness of the proposed approach using two different human action recognition datasets, i.e., UTKinect-Action3D and UCF101 datasets.

Table 34: Human Activity Recognition using Recurrent Bag-of-Features: UTKinect-Action3D Evaluation

Method	# Codewords / GRU Units	Test Accuracy (%)
Average Pooling	-	40.83
Linear BoF	256	44.47
GRU	512	47.71
ReBoF	512	<b>54.64</b>

Table 35: Human Activity Recognition using Recurrent Bag-of-Features: UCF101 Evaluation

Method	# Codewords / GRU Units	Test Accuracy (%)
Average Pooling	-	70.32 $\pm$ 0.43
Linear BoF	1024	71.11 $\pm$ 0.35
GRU	2048	71.04 $\pm$ 0.20
ReBoF	1024	<b>72.02 <math>\pm</math> 0.68</b>

## 3.6 Semi-supervised Compressive Learning with Deep Priors

### 3.6.1 Introduction, Objectives and Summary of the state-of-the-arts

Compressive Learning (CL) and Multilinear Compressive Learning (MCL) [5, 202] are resource-constrained learning paradigms in which the sensor signals are acquired in a compressed manner on the hardware level and the learning model performs inference directly based on the compressed measurements. This learning paradigm is especially suitable for many robotic applications that have computational bottleneck. As the sensor signals are acquired and compressed into few measurements, they can be easily transmitted faster-than-real-time to a centralized server, which can process the data from many agents and return the inference results in real-time.

Semi-supervised learning is another learning paradigm in which the models can leverage unlabeled data and learn from both labeled and unlabeled data. Semi-supervised learning is also very useful in many robotic applications, especially in those scenarios where robots have to continuously operate in new environments and perform online learning. By having semi-supervised learning methods, the robots can ask for the label of few samples and leverage the available unlabeled data in the same domain to learn about the environment.

Although compressive learning and semi-supervised learning are complementary, there exists no work that combines the two learning paradigms to have methods that can learn in both compressive and semi-supervised manner. To this end, our work has made the following contributions:

- We propose a novel method to discover and incorporate prior knowledge into existing

end-to-end Compressive Learning systems. The proposed method can enhance the learning performance of any Compressive Learning algorithms. For the illustrative purpose, we demonstrate the method on the Multilinear Compressive Learning framework that is presented in Section 2.9.

- The proposed method naturally leads to a semi-supervised extension for Compressive Learning algorithms. Our experiments indicate that when having the ability to learn from unlabeled data, the model can indeed improve its performance although having seen only a few labeled samples.

A summary of this work is provided hereafter. The corresponding preprint is listed below, and can be found in Appendix 7.13:

1. [197] D. T. Tran, M. Gabbouj, A. Iosifidis, “*Multilinear Compressive Learning with Prior Knowledge*”, arXiv:2002.07203, 2020

The preliminaries and related works regarding Compressive Learning and the Multilinear Compressive Learning framework can be found in Section 2.9

### 3.6.2 Description of work performed so far

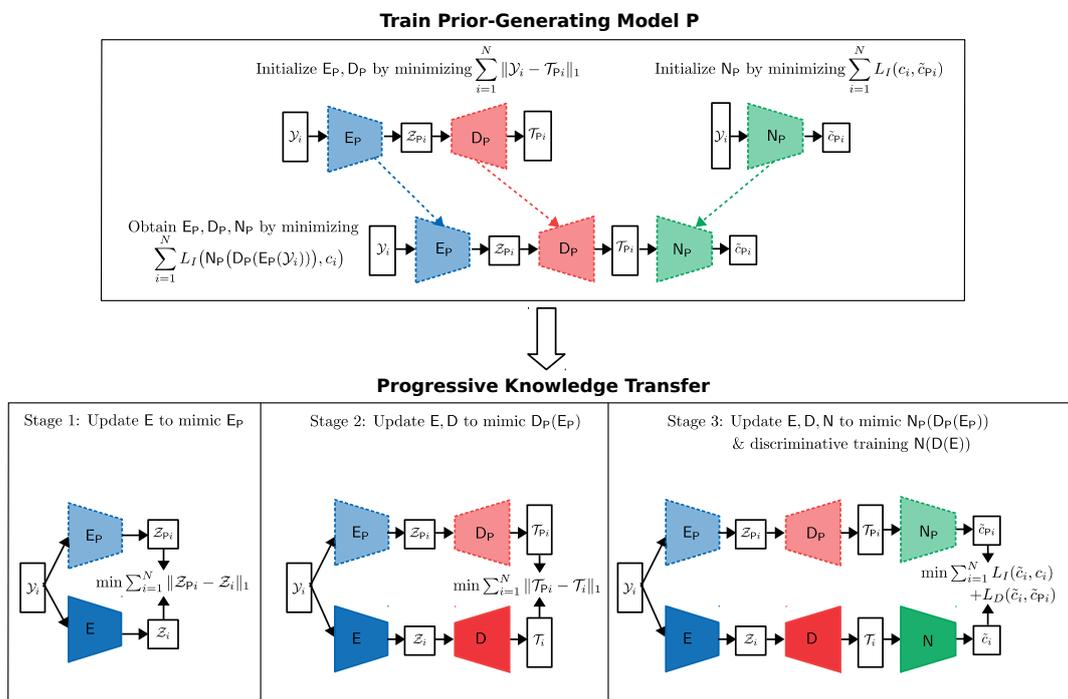


Figure 17: Illustration of the proposed algorithm: we first train prior-generating model P. The knowledge in each component of P is then progressively transferred to the MCL model. Lighter color blocks indicate components of P while darker color ones indicate the corresponding components in MCL model.

### Finding and incorporating prior knowledge

The Multilinear Compressive Learning (MCL) framework [202] consists of three components: the compressive sensing component, the feature synthesis component and the task-specific network. The three components are described in details in Section 2.9. The compressive sensing component, which is implemented on the sensor, performs the following sensing and compression operation:

$$\mathcal{Z} = \mathcal{Y} \times_1 \Phi_1 \times \dots \times_N \Phi_N \quad (70)$$

where  $\mathcal{Y}$  denotes the discrete uncompressed signal.

A common constraint which exists in all CL models is that the sensing operator is linear or multilinear. This limits the capacity of CL models and the amount of information that can be captured from the original signal. Thus, in CL in general and MCL in particular, given a fixed structure of the Feature Synthesis (FS) component and the task-specific neural network  $\mathcal{N}$ , there is always an upper-bound on the capacity of the whole learning system, which makes the problem of learning from compressed measurements  $\mathcal{Z}$  generally harder than other unconstrained learning tasks.

Although we have no direct prior knowledge of a *good* tensor subspace that  $\mathcal{Z}$  should be in, we do know that if we can use highly complex, nonlinear sensing modules and sufficient training data, we can learn to extract and preserve more relevant information from the original signal, compared to a linear/multilinear sensing module. Thus, the tensor subspace learned by a nonlinear sensing module can provide us some supervisory signals about the information that should be contained in compressed measurements  $\mathcal{Z}$ .

Using this intuition, our method to enhance the training of MCL models contains two steps:

- **Finding prior knowledge:** by training a prior-generating network  $\mathcal{P}$  that also has three components: a nonlinear sensing component, a feature synthesis component and a task-specific neural network component. The feature synthesis component and the task-specific neural network of the prior-generating network  $\mathcal{P}$  have exactly the same architectures as the MCL model we want to train. The only difference is that the prior-generating network  $\mathcal{P}$  has a nonlinear sensing component, rather than a linear/multilinear one.
- **Incorporating prior knowledge:** by using 3-stage knowledge distillation (KD) technique. In the first stage, the compressed measurements  $\mathcal{Z}_{\mathcal{P}}$  generated by the prior-generating network are used as supervisory signals to pretrain the multilinear compressive sensing module of the target MCL model. In the second stage, the features  $\mathcal{D}_{\mathcal{P}}$  synthesized by the prior-generating network (the outputs of the feature synthesis component) are used as supervisory signals to pretrain both the sensing and feature synthesis components of the target MCL model. In the final stage, the whole MCL model is trained with the predictions generated by the prior-generating network, as well as the target labels.

Our algorithm is illustrated in Figure 17.

### Semi-supervised Compressive Learning

Let us denote  $\mathcal{L} = \{(\mathcal{Y}_i, c_i) | i = 1, \dots, N\}$  the labeled training set and  $\mathcal{U} = \{\mathcal{Y}_i | i = N + 1, \dots, M\}$  the unlabeled training set. To take advantage of  $\mathcal{L} \cup \mathcal{U}$ , we propose the following modifications to the training procedure of prior-generating model  $\mathcal{P}$ :

- Initialization of the sensing ( $E_P$ ) and feature synthesis ( $D_P$ ) components of the prior-generating model  $P$ : the weights of the sensing ( $\theta_{E_P}$ ) and feature synthesis ( $\theta_{D_P}$ ) component are initialized with values obtained from minimizing the reconstruction error on  $\mathcal{L} \cup \mathcal{U}$ , i.e.,  $\sum_{i=1}^M \|\mathcal{Y}_i - D_P(E_P(\mathcal{Y}_i))\|_1$ .
- Incremental optimization of three components ( $E_P, D_P, N_P$ ) of the prior-generating model via self-labeling: after the initialization step, all parameters of the prior-generating model  $P$  are optimized with respect to the inference loss, which is calculated on the enlarged labeled set  $\tilde{\mathcal{L}}$ :

$$\arg \min_{\theta_{E_P}, \theta_{D_P}, \theta_{N_P}} \sum_{(\mathcal{Y}_i, c_i) \in \tilde{\mathcal{L}}} L_I(N_P(D_P(E_P(\mathcal{Y}_i))), c_i) \quad (71)$$

where  $L_I$  denotes the inference loss.

Initially, the enlarged labeled set is formed from the labeled data, i.e.,  $\tilde{\mathcal{L}} = \mathcal{L}$ . After every  $T$  backpropagation epochs,  $\tilde{\mathcal{L}}$  is augmented with those data instances (with their predicted labels) in  $\mathcal{U}$  that have the most confident predictions from the current  $P$ , given a probability threshold  $\rho$ , i.e.,  $\tilde{\mathcal{L}} = \tilde{\mathcal{L}} \cup \mathcal{C}$ :

$$\mathcal{C} = \{(\mathcal{Y}, c) \mid \mathcal{Y} \in \mathcal{U} \wedge N_P(D_P(E_P(\mathcal{Y})))_{\max} \geq \rho, \\ c = \operatorname{argmax}(N_P(D_P(E_P(\mathcal{Y}))))\} \quad (72)$$

After the enlargement of  $\tilde{\mathcal{L}}$  with the most confident instances, they are removed from the unlabeled set  $\mathcal{U}$ , i.e.,  $\mathcal{U} = \mathcal{U} \setminus \mathcal{C}$ . The training terminates when the enlargement of  $\tilde{\mathcal{L}}$  stops, i.e.,  $\mathcal{C} = \emptyset$ .

Self-labeling is a popular technique in semi-supervised algorithms. While there are many sophisticated variants of this technique [203], the simple modifications proposed above work well in our experiments. Given a prior-generating model  $P$  trained on  $\mathcal{L} \cup \mathcal{U}$ , we use both the labeled and unlabeled data ( $\mathcal{L} \cup \mathcal{U}$ ) to transfer knowledge from the prior-generating knowledge.

### 3.6.3 Performance evaluation

The experiment protocols, detailed description and analysis of different object recognition problems can be found in the Appendix 7.13. In the following, we provide representative results to demonstrate the effectiveness of the MCL framework in face recognition problems, which were conducted via the CelebA dataset [128].

CelebA is a large-scale human face image dataset with more than 200K images at different resolutions from more than 10K identities. In our experiment, we created three versions of CelebA with increasing difficulties by increasing the set of identities to be recognized: CelebA-100, CelebA-200, and CelebA-500 having 100, 200, and 500 identities respectively. Here we note that CelebA-100 is a subset of CelebA-200, and CelebA-200 is a subset of CelebA-500. In addition, we also created a semi-supervised version called CelebA-500S, which has the same number of training instances as CelebA-500, but only 20% of them are labeled.

The results indicating the improvements obtained by using the prior-generating network are shown in Table 36. As we can see from Table 36, as the difficulty of the problems increases

Table 36: Learning performances of MCL [202] and MCLwP (our proposed algorithm) measured on test set (accuracy in %). The last four rows show the performances of prior-generating model P for reference

Measurements	Models	CelebA-100	CelebA-200	CelebA-500
$6 \times 6 \times 1$	MCL [202]	53.96	44.52	38.07
	MCLwP (our)	<b>55.31</b>	<b>47.61</b>	<b>42.51</b>
	$\Delta$ (MCLwP–MCL)	+1.35	+3.09	+4.44
$9 \times 7 \times 1$	MCL [202]	72.08	67.40	61.53
	MCLwP (our)	<b>75.50</b>	<b>72.02</b>	<b>68.73</b>
	$\Delta$ (MCLwP–MCL)	+3.42	+4.62	+7.20
$13 \times 12 \times 1$	MCL [202]	86.27	83.54	83.63
	MCLwP (our)	<b>87.35</b>	<b>86.39</b>	<b>85.97</b>
	$\Delta$ (MCLwP–MCL)	+1.08	+2.85	+2.34
$14 \times 11 \times 2$	MCL [202]	86.90	84.36	83.93
	MCLwP (our)	<b>88.17</b>	<b>86.99</b>	<b>87.29</b>
	$\Delta$ (MCLwP–MCL)	+1.27	+2.63	+3.36

(going from 100 to 500 classes), the recognition performances decrease for both MCL and MCLwP. However, the enhancement obtained by our method also increase compared to MCL.

In Table 37, we show the learning performances of the proposed semi-supervised variant (MCLwP-S). As can be seen from this table, our proposed semi-supervised learning method can indeed achieve better performances when taking advantages of the unlabeled data.

Table 37: Learning performances in Semi-supervised Setting of MCL [202], MCLwP (trained with labeled data only), and MCLwP-S (trained with labeled and unlabeled data)

Measurements	Models	CelebA-500S
$6 \times 6 \times 1$	MCL	12.93
	MCLwP	17.70
	MCLwP-S	<b>21.91</b>
$9 \times 7 \times 1$	MCL	27.46
	MCLwP	25.72
	MCLwP-S	<b>34.16</b>
$13 \times 12 \times 1$	MCL	49.09
	MCLwP	44.56
	MCLwP-S	<b>51.55</b>
$14 \times 11 \times 2$	MCL	41.47
	MCLwP	42.07
	MCLwP-S	<b>58.34</b>

## 4 Social signal (facial expression, gesture, posture, etc.) analysis and recognition

### 4.1 Landmark-based facial expression recognition

#### 4.1.1 Introduction and objectives

Facial expression recognition has been widely studied in the past several years and it is of great importance in different areas of computer vision such as sociable robotics and human-computer interaction (HCI). Various machine learning and deep learning methods have been proposed for facial expression recognition (FER) which utilize different data modalities such as video streams and audio signals to encode 7 facial expressions which are anger, sadness, happiness, fear, disgust, surprise and neutral. FER methods are mainly categorized into two groups: 1) *static methods* which use an image as input to classify the facial expression depicted in it, and 2) *dynamic methods* which use videos or a sequence of images as input to classify the facial expression by considering both spatial and temporal features for classification. In this section, we focus on dynamic methods for video-based facial expression recognition, while in the next Section we describe a static method operating on images.

It has been shown that the FER performance can be improved by utilizing the localized facial landmark coordinates for face alignment [138]. The facial landmarks do not only encode high-level features representing the most informative face locations in a compact structure, but they are also invariant to the face scale, illumination and head pose variations, and face appearance. Deep learning algorithms which employ facial landmark features instead of images, videos or other data modalities have been rarely studied recently. In some studies [95, 105, 70], multi-modal data fusion based on facial landmarks and images or videos are proposed to show the effectiveness of facial landmarks in FER performance improvement. The existing deep learning approaches which utilize facial landmark features, typically concatenate the facial landmark coordinates to form a sequence of vectors to be used by Recurrent Neural Networks, or reorganize them to form a grid map so that they can be in a form suitable to become the input of Convolutional Neural Networks. Therefore, these methods are not capable to capture the dynamic spatial and temporal features encoded in the facial landmarks in a sequence of frames.

Similar to human body skeletons used for human activity recognition (see Sections 3.2, 3.3 and 3.4), facial landmarks are non-Euclidean structured data that can be modeled by a graph in which the landmark points are the graph nodes and the relationship between them denoted by the spatial arrangement of the landmark points during the execution of facial expressions are the edges connected graph nodes. The objective of this study is to employ Graph Convolutional Networks (GCNs) to extract informative features from a sequence of graphs, encoding facial landmarks through different time steps, for facial expression recognition. This will be our focus in for this task in the next months.

#### 4.1.2 Summary of state of the art

Facial landmarks have been widely used in FER methods in conjunction with other data modalities to enhance performance. Recently, many real-time facial landmark detection methods have been developed which have good performance in addition to their high efficiency [96, 219, 164, 11]. Figure 18 shows a sequence of 3 facial images of a person performing the expression “surprise” at three different time steps (i.e. the start (left), the apex of the expression (right) and an



Figure 18: Illustration of facial images in 3 different time steps (the first row) and their corresponding extracted landmark points (the second row) which are denoted as graph nodes in a spatio-temporal graph.

intermediate point in time (middle)) and the aligned landmark points for each image, which are extracted using the method in [47].

Facial landmark-based methods based on GCNs have been recently proposed in [141], [129] for facial expression and micro-expression recognition, respectively. In [141], a high-computational cost landmark extractor [47] was adopted to extract accurate 2D coordinates of 68 landmark points from each facial image in a video, or sequence of images. The extracted landmarks were modeled by a directed spatio-temporal graph. This graph is constructed using landmark points as nodes and triangle meshes among all landmarks, which are built by Delaunay method [38], as edges. The edges are weighted by the  $L_2$  distance between the nodes. The spatial connections between the graph nodes represent the muscle movements of the face and the temporal connections represent facial expression variations through time. Inspired by methods recently proposed for skeleton-based human action recognition, like the DGNN [177], the FER method also employs a multi-layer spatio-temporal GCN model to extract features from the spatio-temporal facial landmark graph and introduces the extracted features to a fully connected classification layer to predict the facial expression. The performance of this method has been evaluated on three widely used datasets, CK+ [134], MMI [145] and AFEW [43], and it has achieved comparable performance to the current state-of-the-art image/video-based FER methods.

### 4.1.3 Future work

Skeleton-based human action recognition and landmark-based facial expression recognition share similar ideas on using a sequence of graph structured data extracted from videos describing the human (face or body) activity and transforming the FER/HAR problem to a spatio-temporal graph classification problem. Accordingly, our GCN-based methods proposed for skeleton-based human action recognition can be extended for landmark-based facial expression recognition. Accordingly, in our future work we will explore the performance of existing state-of-the-art spatio-temporal GCN-based methods proposed for skeleton-based human action

recognition, including our proposed methods TA-GCN [78], PST-GCN [77] and ST-BLN [76] which are described in Sections 3.2, 3.3, 3.4 of this Deliverable, respectively, and try to extend them for dynamic landmark-based facial expression recognition to enhance both recognition performance and efficiency.

## 4.2 Probabilistic Class-Specific Classification

### 4.2.1 Introduction and objectives

Deep neural networks trained on large datasets can be used as generic feature extractors. For example, Convolutional Neural Networks trained on the ImageNet dataset [40] have been widely used as image-to-vector mappings coupled with machine learning methods receiving as input vector-based data representations. This is due to that neural networks trained on large datasets are able to learn data representations, especially at their earlier layers, that encode information in the input images that is useful for solving a multitude of image analysis tasks. The use of deep neural networks as generic feature extractors is commonly referred to as Transfer Learning, since information for an image analysis problem learnt by training the neural network on the large dataset is transferred to another image analysis problem through the neural network's weights.

For image analysis tasks where the objective is the discrimination of a class of interest from any other possibility, class-specific methodologies have shown to outperform binary classification methodologies. For example, in facial expression recognition based on facial images it is expected that the multiple images depicting the facial expressions under consideration will exist in a dataset designed to solve the problem, while the number of images describing all other facial expressions as well as other facial images with unspecified facial expression will be much higher. Moreover, to adequately well describe the differences of any other possibility, the number of sample images that do not belong to the class of interest needs to be high leading to costly annotation of the entire dataset. To handle this situation, the most widely adopted approach is to assign sample images belonging to a class different than the class of interest a negative label. However, the sample images belonging to the negative class and depict the same scene are still expected to be close to each other, thus forming subclasses. Existing class-specific methodologies do not consider such subclass structure of the negative class. Moreover, they are not able to provide a classification rule that can express the probability of correct assignment of an input image to the positive (or negative) class. We have proposed a probabilistic framework for class-specific learning that addresses these drawbacks of existing methods.

A summary of the state of the art and our work is provided hereafter. The corresponding paper is listed below, and can be found in Appendix 7.15:

- [85] A. Iosifidis, “*Probabilistic Class-Specific Discriminant Analysis*”, IEEE Access, vol. 8, pp. 183847-183855, 2020

### 4.2.2 Summary of state of the art

To perform class-specific learning, Class-Specific Discriminant Analysis (CSDA) and its variants [10, 90, 86, 195] have been proposed for determining an optimal subspace that discriminates samples belonging to the class of interest from samples belonging to the negative class, which combines all samples not belonging to the class of interest. While multi-class discriminant analysis models, like Linear Discriminant Analysis (LDA) and its variants [51, 87, 89,

224, 183, 225] can be used for determining class-specific subspaces, they are inherently limited by their formulation of the subspace learning problem, as they can define subspaces with limited number of dimensions. By defining multiple discriminant dimensions, CSDA can achieve better class discrimination and better performance. CSDA defines the discriminant criterion to be solved based on the the positive class data and the scatter of all negative data with respect to the mean of the positive class. Such a class discrimination criterion overlooks the structure of the negative class. Since in practice samples forming the negative class belong to many classes, different from the positive one, it is expected that they will form subclasses. After determining the data projection matrix  $\mathbf{W}$ , the training vectors  $\mathbf{x}_i, i = 1, \dots, N$  are mapped to the discriminant subspace by applying  $\mathbf{z}_i = \mathbf{W}^T \mathbf{x}_i$ . When a classification problem is considered, a classifier is trained using  $\mathbf{z}_i$ . For example, [90] trains a linear SVM on vectors  $\mathbf{d}_i = |\mathbf{z}_i - \boldsymbol{\mu}|$ , where  $\boldsymbol{\mu} = \mathbf{W}^T \mathbf{m}$  and the absolute value operator is applied element-wise on the resulting vector. However, this approach decouples the class discrimination step from the classification step. We proposed a generic probabilistic framework that can i) naturally incorporate the subclass information of the negative class in the optimization process, ii) define a classification rule in the discriminant subspace defined by the optimization of the class discriminant criterion, and iii) provide a probabilistic interpretation of the classification output.

### 4.2.3 Description of work performed so far

In the following, we describe the proposed Probabilistic Class-Specific Discriminant Analysis method. Let us denote by  $\mathbf{x} \in \mathbb{R}^D$  a random variable, realizations of which correspond to samples of the positive and negative classes in a class-specific problem.  $\mathbf{x}$  can be a representation of an input image expressed at a layer of a neural network, or it can be an input image representation defined otherwise. We model the positive class  $c_p$  as a multivariate Gaussian distribution having a mean vector  $\mathbf{m} \in \mathbb{R}^D$  and a covariance matrix  $\Phi_p \in \mathbb{R}^{D \times D}$ , i.e.:

$$P(\mathbf{x}|c_p) \sim N(\mathbf{x}|\mathbf{m}, \Phi_p). \quad (73)$$

We also model the negative class subclasses by multivariate Gaussian distributions:

$$P(\mathbf{y}) \sim N(\mathbf{y}|\mathbf{m}, \Phi_n), \quad (74)$$

where  $\mathbf{y} \in \mathbb{R}^D$  is a random variable concrete realizations of which correspond to the negative subclasses mean vectors and  $\Phi_n$  expresses the scatter of the negative subclasses with respect to the positive class mean vector. Then, the distribution of the samples from the negative subclasses with respect to the positive class is expressed by the following multi-modal distribution:

$$P(\mathbf{x}|c_n) = \int N(\mathbf{y}|\mathbf{m}, \Phi_n) N(\mathbf{x}|\mathbf{y}, \Phi_w) d\mathbf{y}, \quad (75)$$

where  $\Phi_w$  is the (common) covariance matrix of the negative subclasses. We can also define the following matrix  $\Phi_O = \Phi_n + \Phi_w$ , expressing the covariance of the entire dataset from the positive class mean.

The parameters of the above probabilistic model are estimated by maximizing the (log) probability of correct assignment of the training samples  $\mathbf{x}_i, i = 1, \dots, N$ :

$$\mathcal{L} = \ln P(\mathcal{S}_p|c_p) + \ln P(\mathcal{S}_n|c_n), \quad (76)$$

where  $\mathcal{S}_p = \{\mathbf{x}_1, \dots, \mathbf{x}_{N_p}\}$  is a set formed by i.i.d. positive samples, and  $\mathcal{S}_n = \{\mathcal{S}_1, \dots, \mathcal{S}_K\}$  is the set of i.i.d. negative samples, forming  $K$  subclasses  $\mathcal{S}_k, k = 1, \dots, K$ .

After the estimation of the model's parameters, a new sample  $\mathbf{x}^*$  can be classified by evaluating the posterior probabilities of the positive and negative classes given by:

$$P(c_p|\mathbf{x}^*) = \frac{p(\mathbf{x}^*|c_p)P(c_p)}{p(\mathbf{x}^*)} \quad \text{and} \quad P(c_n|\mathbf{x}^*) = \frac{p(\mathbf{x}^*|c_n)P(c_n)}{p(\mathbf{x}^*)}. \quad (77)$$

This leads to the following classification rule:

$$g(\mathbf{z}^*) = \ln P(c_p) - \ln P(c_n) + \frac{1}{2} \ln |\tilde{\Phi}_O| - \frac{1}{2} \ln |\tilde{\Phi}_p| - \frac{1}{2} \mathbf{z}^{*T} \tilde{\Phi}_p^{-1} \mathbf{z}^* + \frac{1}{2} \mathbf{z}^{*T} \tilde{\Phi}_O^{-1} \mathbf{z}^*, \quad (78)$$

where  $\mathbf{z}^* = \mathbf{W}^T \mathbf{x}^*$  is the projected vector to the class-specific discriminant subspace and  $\tilde{\Phi}_p$  and  $\tilde{\Phi}_O$  are the scatter matrix of the positive class and the (weighted) total scatter matrix of the data expressed in the class-specific discriminant subspace.

#### 4.2.4 Performance evaluation

The proposed method was evaluated on two settings, i.e. image retrieval as existing class-specific subspace learning methods are commonly evaluated on retrieval problems, as well as image classification problems. While standard class-specific approaches employ a classifier which is applied the data representations in the class-specific subspace, the proposed approach exploits the classification rule given in Eq. (78). We used the BU, Jaffe and Kanade datasets describing image-based facial expression recognition/verification problems for which we used the input images as features to the PCSDA model, the YALE and AR face recognition data sets for which we used the input images as features to the PCSDA model, and the 15-scenes and Caltech101 datasets describing scene recognition problems for which we used features extracted from deep neural networks as features to the PCSDA model, i.e. we adopted a transfer learning approach. The experimental protocols, along with performance evaluation experiments are presented in Appendix 7.15. Here we provide experimental results for classification problems described by the above-mentioned datasets in Table 38. Here we need to note that for performing classification using other subspace-based methods (PCA and LDA) one needs to perform a two-step process, i.e. data projection to the subspace followed determined by the method followed by the application of a classification method. We combined PCA with SVM and LDA with Nearest Class Centroid (NCC) classifiers. The choice of LDA+NCC is based on the fact that the LDA optimization problem exploits as a class discrimination criterion that of NCC. Despite the fact that PCSDA is a subspace learning method, its probabilistic formulation leads to the classification rule in Eq. (78), and thus there is no need to perform a classification method for PCSDA. As can be seen, the proposed PCSDA classifier performs on par with well-established classifiers, as well as with combination of subspace learning methods with classification schemes.

Table 38: Performance on classification problems.

	RR	SVM	PCA+SVM	LDA+NCC	PCSDA-1	PCSDA-K
Jaffe	0.6750 (0.1032)	0.6714 (0.1225)	0.2809 (0.0542)	0.7049 (0.1299)	0.6261 (0.1262)	0.5242 (0.1249)
Kanade	0.4001 (0.1520)	0.4122 (0.1271)	0.2597 (0.0464)	0.5194 (0.2677)	0.4546 (0.2427)	0.3861 (0.1894)
BU	0.4463 (0.0819)	0.4562 (0.1391)	0.2592 (0.0180)	0.5632 (0.1706)	0.5731 (0.1288)	0.5431 (0.1793)
YALE	0.9369 (0.0090)	0.9396 (0.0851)	0.2715 (0.0851)	0.9517 (0.0208)	0.9409 (0.0214)	0.9611 (0.0222)
AR	0.9664 (0.0352)	0.9683 (0.0343)	0.7346 (0.0394)	0.9760 (0.0452)	0.9674 (0.0481)	0.9516 (0.0629)
15-scenes	0.8997 (0.0854)	0.9005 (0.0517)	0.5365 (0.1507)	0.8993 (0.0539)	0.8573 (0.0654)	0.8130 (0.1275)
OptDigits	0.9875 (0.0900)	0.9881 (0.0082)	0.8985 (0.0706)	0.9899 (0.0077)	0.9569 (0.0141)	0.9826 (0.0074)
Caltech101	0.8130 (0.0971)	0.8140 (0.0807)	0.6042 (0.0394)	0.8075 (0.1448)	0.7605 (0.1829)	0.7663 (0.1955)

## 5 Deep speech and biosignals analysis and recognition

### 5.1 Speech Command Recognition based on Quadratic Self-organized Operational Network

#### 5.1.1 Introduction, Summary of the state-of-the-arts and Objective

Spoken commands recognition (SCR) is a subcategory of Automatic Speech recognition (ASR) in which the machines identify short spoken commands. It is a key technology which enables machines to understand human commands and act based on them. There are a huge number of potential applications for SCR including smart-phones with mobile assistants, robots following human spoken instructions, and smart home assistants. Such applications are also essential for OpenDR project, as a way to facilitate human-robot interaction.

Classical SCR approaches rely on extracting discriminative features from the raw speech. Frequency domain features like fast Fourier transform are among the most widely used features to convert the time domain speech into frequency space. After extraction of meaningful attributes, an acoustic model such as hidden Markov model (HMM) is classically used to represent the sequence of words of phonemes. Combination of neural networks and HMM-based approaches have also been in use since more than 20 years ago. Recently, deep learning-based approaches have dominated, in terms of performance accuracy, over conventional machine learning methods for SCR applications. Neural networks like convolutional neural networks (CNNs) and long short term memory (LSTM) networks have raised the accuracy of SCR history beyond what was achievable with classical methods. These deep models could be applied on the raw audio or on separately extracted features. However, these structures, especially those with high recognition rates, often rely on complex architectures and therefore require great amounts of memory and processing power to achieve real-time operation. Their energy consumption is also very high. These limitations significantly limit the SCR applications on robots or hand-held devices. In fact, lower computational power and memory limitations of the embedded devices are still serious barriers to fully exploit the benefits of highly accurate complicated deep networks. As robotics applications mainly target computationally constrained environments, fast and lightweight implementations are needed to work smoothly in real time.

To tackle the problem of computational complexity on embedded devices, two general approaches are commonly used. One can squeeze the state of the art deep models to reduce the computational costs without severely degrading the performance accuracy. Alternatively, more efficient deep networks and methods can be employed from the beginning in order to achieve better accuracy rates with lower computational costs. The second approach especially

becomes important when training examples are scarce or the input signals are of low quality or resolution. Using either of these ways, one can implement a reasonable classifier on energy constrained embedded devices.

We follow the second approach and propose a new network lightweight layer which enhances the speech command recognition capability. To this end, we use a novel combination of self organized operational neural networks (SelfONNs) [102] and quadratic forms kernels [242].

A summary of this work is provided hereafter. The corresponding preprint is listed below and can be found in Appendix 7.17:

- [181] M. Soltanian, J. Malik, J. Raitoharju, A. Iosifidis, S. Kiranyaz, M. Gabbouj. *Speech Command Recognition in Computationally Constrained Environments with a Quadratic Self-organized Operational Layer*. Under Review in IEEE ICASSP 2021, (arXiv:2011.11436).

### 5.1.2 Description of work performed so far

In this subsection we provide background on the self organized operational neural networks, as well as the quadratic kernels. Then we follow with the formulation of the proposed approach.

#### Self Organized Operational Neural Networks

Operational Neural Networks (ONNs) [101] are heterogeneous networks with a generalized neuron model that include a dictionary of nodal operators (instead of just an ordinary convolution) to boost the performance of the conventional CNNs. However, they need a greedy search to find the optimal operators which result in the best accuracy. The performance is also highly dependent on the selection of the initial operator set dictionary. In order to resolve these issues, the authors in [102] have recently proposed SelfONNs with generative neurons which are able to adapt the nodal operators of each neuron during the training phase. This removes the need for a fixed operator set and greedy search within that library to find the optimal set of operators. The authors show, via extensive experiments, the superiority of SelfONNs over ONNs in four applications: image synthesis, image denoising, face segmentation, and image transformation.

The main concept behind SelfONN is to use Taylor series expansion of the nodal function at each layer instead of the applying the ordinary convolution. An ordinary convolutional layer computes the convolution of the input with the layer weights as the following:

$$Y = X \circledast W + b, \quad (79)$$

where  $X$ ,  $W$ ,  $b$ , and  $Y$ , are input, weight, bias, and output tensors, respectively.

In ONNs, however, this is generalized to the following form:

$$Y_{\text{ONN}} = \Psi(X, W) + b, \quad (80)$$

where  $\Psi$  is an arbitrary nodal operator and may be a combination of different functions. If we set  $\Psi$  function equal to the regular convolution operator ( $\circledast$ ), the ONN layer transforms back to the ordinary convolutional layer.

In SelfONNs, instead of using a specific function  $\Psi$  as the nodal operator, a suitable function can be approximated with a truncated Taylor series expansion to resolve the above mentioned

shortcomings of the ONN. So, the SelfONN layer is defined as:

$$\begin{aligned}
 Y_{\text{SelfONN}} &= W_{T_0} + (X - A) \circledast W_{T_1} + (X - A)^2 \circledast W_{T_2} + \\
 &\quad \dots + (X - A)^K \circledast W_{T_K} + b_S \\
 &= \sum_{i=0}^K (X - A)^i \circledast W_{T_i} + b_S,
 \end{aligned} \tag{81}$$

where  $A$  is the point around which  $\Psi$  is expanded.  $W_{T_i} = \frac{1}{i!} \frac{\partial \Psi}{\partial X^i}$  is the  $i$ 'th order partial derivative of  $\Psi$  with respect to the input, and  $K$  controls the order of approximating polynomial.  $b_S$  is the bias term associated with the SelfONN layer. It is argued in [102] that the bias term ( $b_S$ ) and the DC term in Taylor expansion ( $W_{T_0}$ ) can be merged into a single term. In practice, the input is assumed to be centered around zero ( $A = 0$ ). As the goal is not to estimate any particular function  $\Psi$ , but to learn the most suitable nodal function,  $W_{T_i}$ s are learned via a training process (each  $W_{T_i}$  is a learnable weight which can be learned via back propagation).

### Quadratic-form Kernels

The core idea behind the network layer based on the quadratic form expansion [242] is that it generalizes the linear convolution by taking into account the cross correlation between the input elements within the receptive field of the layer kernel. In other words, in addition to ordinary convolution between the input and the weight, a second-order convolution between the input and an expanded weight tensor is computed which improves the final performance in terms of classification accuracy. The layer based on quadratic form expansion is therefore expressed as:

$$Y_{\text{Quadratic}} = X^H \circledast W_{Q_2} \circledast X + X \circledast W_{Q_1} + b_Q \tag{82}$$

in which  $W_{Q_1}$  and  $W_{Q_2}$  are weight tensors,  $b_Q$  is the bias term, and  $()^H$  is the Hermitian operator. Equation (82) is the same as (79) except for the added quadratic term  $X^H \circledast W_{Q_2} \circledast X$ . This quadratic term improves the SCR accuracy, as shown in the simulations, mainly because it models second-order dependencies of the input features.

### Proposed Quadratic Self Organized Operational Neural Network method

The concepts of SelfONN and quadratic convolution have not yet been applied to SCR problem. We are the first who use these methods in computationally constraint environments in order to boost the SCR accuracy. Additionally, we combine these two layer types into a unique layer which inherits advantages of both SelfONN (by a generalized convolution) and quadratic kernels (by modeling cross correlation of input feature points). To do so, we merge the Taylor series expansion in (3) with the quadratic-form convolution in (4) as the following:

$$\begin{aligned}
 Y_{QS} &= \sum_{i=0}^K (X^H - A^H)^i \circledast W_{Q_{2i}} \circledast (X - A)^i + \\
 &\quad \sum_{i=0}^K (X - A)^i \circledast W_{Q_{1i}} + b_Q,
 \end{aligned} \tag{83}$$

in which  $Y_{QS}$ ,  $W_{Q_{1i}}$ ,  $W_{Q_{2i}}$ , and  $b_Q$  are the output feature map, linear and quadratic weight tensors, and the bias term, respectively.

The flow diagram of the proposed layer is shown in Figure 19. The nodal operator based on the Taylor series expansion proposed in [102] is firstly applied to the input. The resulting feature

map is fed to the quadratic-form convolution. Regular max pooling and tangent hyperbolic activation are then applied to the output. Dropout is also used after activation function in the training phase to avoid over-fitting.

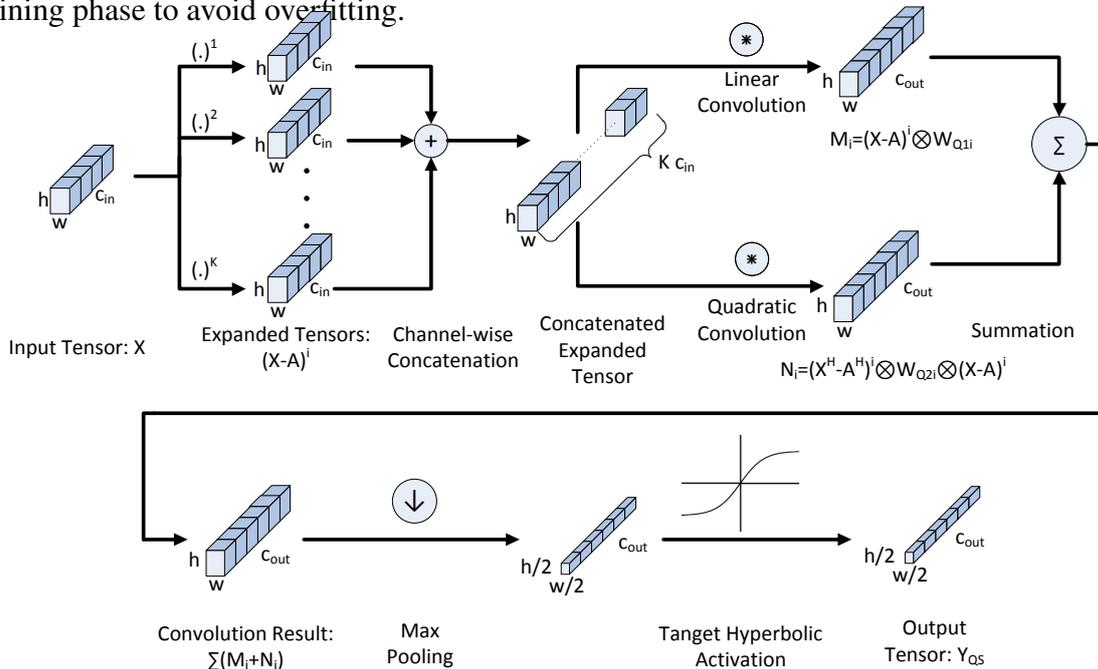


Figure 19: Flow diagram of the proposed quadratic-SelfONN based layer: The spectrogram, MFCC, or any feature maps are fed as the input. The input is then expanded as a power series, and the resulting feature maps are concatenated. Afterwards, linear and quadratic convolutions are applied and the results are summed together. Finally, max pooling and tangent hyperbolic activation function are applied.

### 5.1.3 Performance evaluation

#### Datasets

The proposed model is evaluated on Google Speech Commands (GSC) [216] and synthetic speech commands (SSC) [2] datasets.

The accuracy of the proposed method against the baseline is evaluated based on the number of correctly predicted labels in the test set divided by the total number of test samples, for both GSC and SSC datasets.

#### Data Pre-processing

The 1D raw input audio is converted to a 2D signal before being fed to the network. The, Mel-frequency cepstral coefficients (MFCCs) are extracted from each audio signal. Since the configuration is aimed to be used in computationally constrained environments, only 20 cepstral coefficients are kept at each signal window. The resulting single channel 2D MFCC  $\in \mathbb{R}^{20 \times 51}$  is finally normalized to lie between -1 and 1.

#### Network Structure

For comparison of the proposed layer with the baseline convolutional layer in terms of accuracy of final speech commands classification, we need to use the layers in a deep structure which

Table 39: Performance comparison of the proposed layer with ordinary convolutional and SelfONN layers over GSC and SSC datasets.

Dataset	Ordinary convolution	SelfONN	Proposed
GSC	85.2	87.6	89.8
SSC	95.5	97.1	97.9

is end-to-end trainable. As we primarily target computationally constrained environments, such as OpenDR specific robotics applications, we use a lightweight basic model for the evaluations. While the model could not dominate the complex state-of-the-art networks with unbounded resources, it is still quite enough for a proof of concept, i.e., showing the advantages of the proposed layer over the baseline convolutional layer under the resource limitations. Accordingly, we choose a structure similar to LeNet-1 [110]. The original LeNet-1 has two convolutional and one fully connected layers. The network employed here has the same structure, but the convolutional layers are replaced with the proposed quadratic SelfONN layer.

## Experimental Results

Table 39 shows the comparison of accuracy on the test set between convolutional, SelfONN, and proposed layers for GSC and SSC datasets. As shown, the network comprised of the proposed layer outperforms the network with convolutional or SelfONN layers in terms of test accuracy over both datasets. The accuracy gains for the proposed layer over the baseline convolutional layer are nearly 4.6% and 2.2% for GSC and SSC datasets, respectively. Similarly, the accuracy gains of the proposed layer over SelfONN layer are nearly 2.4% and 0.8% for GSC and SSC datasets, respectively.

Figure 20 evaluates the proposed layer based on the deployment complexity. To this end, the deployment time for a single 1 second speech command is computed by averaging the deployment time over the whole test set. The simulations and learning process are run on a machine with a Nvidia GeForce GTX 1080.

As can be seen, the deployment times of the proposed layer are higher than that of the convolution layer. However, the deployment time of the proposed layer is quite close to that of SelfONN.

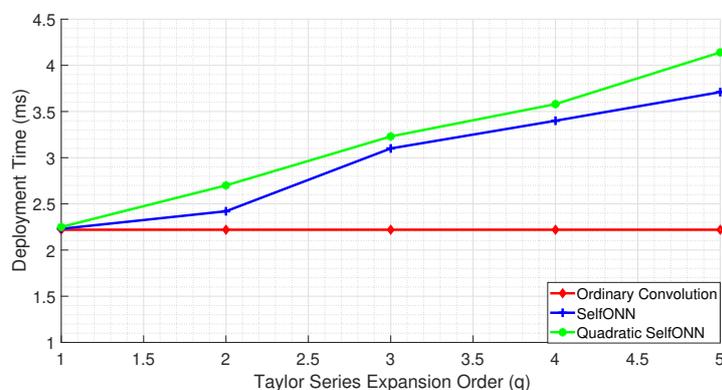


Figure 20: Comparison of the convolutional, SelfONN, and the proposed layers in terms of deployment time on a Nvidia GeForce GTX 1080 for SCR task over GSC dataset.

### **Comparison to the State-of-the-art**

According to leader board section of Kaggle challenge on GSC dataset [1], the first team has achieved an accuracy around 91%. However, they probably used much more complex networks or ensemble of classifiers. They also used the original wave signals with full resolution spectrograms or higher number of cepstral coefficients. Since, the purpose of our work is the proof of concept of the proposed layer in computationally limited environments, we just used a very light-weight network, with small resolution inputs and compared the accuracy with the baseline convolutional layers. Even with these limitations, the achieved accuracy is quite high. However, it is expected that employing the proposed layer in deeper networks will result in higher accuracy values. A similar discussion also holds for SSC dataset.

#### **5.1.4 Future Work**

Future work will involve using the proposed layer in more complex networks in combination with network squeezing methods like network pruning, distillation, and sparsification of the fully connected layers. Due to the great performance of the proposed method, employing it in few shot learning problem in SCR looks promising.

## **5.2 Heart Anomaly Detection using Attention-based Neural Bag-of-Features Learning**

### **5.2.1 Introduction, Summary of the state-of-the-arts and Objective**

Medical diagnosis, which plays a crucial role in ensuring human prosperity, is inherently an intricate process. The quality of the diagnosis is highly dependent on the expertise of the examiner. Since it takes several years and a great amount of resources to train human experts, medical diagnosis tools have been actively developed over the past decades to assist human examiners. Among the deadly diseases, cardiovascular diseases are the number one cause of death globally. Thus, a vast amount of healthcare products, including assistant robots in healthcare environments, have been developed to monitor patient's heart conditions.

For cardiovascular diseases, the ability to detect the anomaly of the heart in an early stage plays a crucial factor in the survival rate. Traditionally, early detection of heart anomaly is conducted by frequent visits to the cardiologist. This preventive approach, however, creates a huge amount of workloads to the cardiologists. With the development of machine learning techniques in recent years, several automatic cardiovascular diagnosis pipelines have been proposed to assist the cardiologists in the detection of heart anomaly. The majorities of heart anomaly detection pipeline operates on two types of input signals: Electrocardiogram (ECG) and Phonocardiogram (PCG).

ECG signal is the electrical voltages recorded over time through a set of electrodes that are placed on the skin. The electrodes detect the tiny electrical changes generated by the cardiac muscles during the cardiac cycles or heartbeats. By observing the ECG patterns, the cardiologists can detect cardiac abnormalities such as the rhythm disturbances or the lack of coronary artery blood flow. 12-lead ECG recordings are often used as a standard clinical analysis tool to diagnose heart related symptoms such as chest pains, electrical injuries, ventricular failures and so on. Although a 12-lead ECG device can provide more accurate signals about the activities of the heart, it is impractical to use over a long period of time. Thus, in majorities of portable healthcare devices, the ECG sensor often contains only a single lead, or up to three leads. As

ECG signals are electrical voltages measured over time, ECG recordings are often treated as univariate (single lead) or multivariate (multiple leads) time-series.

While ECG reflects the heart activities through electrical changes, PCG reflects the acoustic content of the heart, i.e., the sounds and murmurs, generated by during the cardiac cycles. The opening and closure of the heart valves is associated with accelerations and decelerations of the blood, leading to vibrations of the entire cardiac structure. These vibrations are audible at the chest wall, and listening for specific heart sounds can give an indication of the health of the heart. The PCG recording are often single-channel and considered as univariate time-series. Both ECG and PCG are a quick, safe, and painless way to check for heart rate, heart rhythm, and signs of potential heart disease.

Since ECG and PCG signals are signals that involve temporal information, a large amount of automatic diagnosis methods employ sequence modeling techniques. In the past, the detection pipelines contain multiple processing stages such as noise removal, heartbeat detection and segmentation, then anomaly classification [3]. Each of the processing stage is individually designed and incorporated with prior knowledge about the noise model, the characteristics of the heartbeat and so on. In these methods, different handcrafted features, which are designed to capture different spatial and temporal properties from the time-series, are used to train the final classifier. Recently, as end-to-end stochastic learning paradigm has become prominent, more and more end-to-end detection pipelines that utilizes deep neural networks have been proposed [166].

Convolutional Neural Networks (CNN) or Recurrent Neural Networks (RNN) or the combination of both are popular neural network architectures that have been employed in the heart anomaly detection systems. The common drawback in existing systems is that the employed deep neural networks are usually computationally intensive, which are unsuitable for real-time applications in hand-held devices or mobile robots. In addition, deep neural networks are considered by the health specialists as blackbox, which can only diagnose with a *yes* or *no* diagnosis without the *why*. Thus, in order to enable a collaborative environment between machine and human experts, it is desirable to develop detection algorithms that are capable of giving interpretable diagnosis. To this end, the contribution of our work includes the following:

- We propose a new type of attention mechanism, for sequence data, dubbed as 2DA, which acts as a layer in a computation graph and can be optimized jointly with other layers using stochastic gradient descent. The proposed attention mechanism is capable of identifying relevant sources of information to perform selective masking on the input sequence.
- The proposed attention mechanism is incorporated into the Neural Bag-of-Feature model to form an efficient anomaly detection model that generates interpretable and accurate diagnosis. Since the Neural Bag-of-Feature model is highly computationally efficient, the proposed detection pipeline is suitable for real-time applications.
- The resulting heart anomaly detection models will be optimized and incorporated into the OpenDR toolkit in the initial release. In addition, we will also provide pretrained models with different computational complexities to support different hardware requirements.

A summary of this work is provided hereafter. The corresponding preprints are listed below, and can be found in Appendix 7.14:

1. [201] D. T. Tran, N. Passalis, A. Tasos, M. Gabbouj, A. Iosifidis, “*Attention-based Neural Bag-of-Feature Learning for Sequence Data*”, arXiv:2005.12250, 2020.

## 5.2.2 Description of work performed so far

### 2D-Attention Mechanism (2DA)

A matrix  $\mathbf{S} \in \mathbb{R}^{M \times N}$  is a second-order tensor which has two modes, with  $M$  and  $N$  are the dimensions of the first and second mode, respectively. The matrix representation provides a natural way to represent a signal with two different sources of information. For example, a multivariate time-series is represented by a matrix with one mode representing the temporal dimension, and the other mode represents different sources that generate individual series.

The general idea of attention mechanism is to highlight important elements in the data while discarding irrelevant ones. For data represented as a matrix  $\mathbf{S}$ , rather than considering each element in  $\mathbf{S}$  individually, we would like to actively select certain columns or rows of  $\mathbf{S}$  while discarding the others. This is because columns or rows of  $\mathbf{S}$  usually form coherent sub-groups of the data. For example, discarding some temporal events or some individual series in a multivariate series corresponds to removing some rows or columns, depending on the orientation of the matrix.

To adaptively determine and focus on different columns or rows of a matrix, we propose 2D-Attention (2DA), with the functional form denoted by  $\mathcal{F}_{2DA}$ . This function takes a matrix  $\mathbf{S} \in \mathbb{R}^{M \times N}$  as the input, and returns  $\tilde{\mathbf{S}} \in \mathbb{R}^{M \times N}$  as the output. That is:

$$\tilde{\mathbf{S}} = \mathcal{F}_{2DA}(\mathbf{S}) \quad (84)$$

$\tilde{\mathbf{S}}$  can be considered as a filtered version of  $\mathbf{S}$ , where irrelevant columns of  $\mathbf{S}$  with respect to the learning problem are zeroed out. Here we should note that  $\mathcal{F}_{2DA}$  performs adaptive attention with respect to columns of  $\mathbf{S}$ . To focus on different rows of  $\mathbf{S}$ , we can simply apply  $\mathcal{F}_{2DA}$  to the transpose of  $\mathbf{S}$ .

The selection or rejection of the columns of  $\mathbf{S}$  is conducted via element-wise matrix multiplications as follows:

$$\tilde{\mathbf{S}} = \tau(\mathbf{S} \odot \mathbf{A}) + (1 - \tau)\mathbf{S} \quad (85)$$

where  $\mathbf{A} \in \mathbb{R}^{M \times N}$  denotes the attention mask with values in the range  $[0, 1]$ . Each column in  $\mathbf{A}$  encodes the importance of the corresponding column in  $\mathbf{S}$ . That is, the attention mask contains values that are close to 1 corresponding to those columns in  $\mathbf{S}$  that contain important information for the downstream learning task and vice versa. In Eq.(85), parameter  $\tau \in \mathbb{R}$ , which is jointly optimized with other parameters, is used to allow flexible control of the attention mechanism: when  $\mathbf{S}$  contains redundant or noisy information in its columns, the effect of attention mask  $\mathbf{A}$  is enabled by pushing  $\tau$  close to 1; on the other hand, when every column of  $\mathbf{S}$  is necessary, i.e., there is no need for attention, pushing  $\tau$  close to 0 will disable the effect of  $\mathbf{A}$ . The necessity of attention is thus automatically determined by optimizing  $\tau$  with respect to a given problem.

To calculate the attention mask  $\mathbf{A}$ , the proposed 2DA method learns to measure the relative importance between columns of  $\mathbf{S}$  via a specially designed weight matrix  $\mathbf{W} \in \mathbb{R}^{N \times N}$ : all elements of  $\mathbf{W}$  are learnable, i.e., they are updated during stochastic optimization, except the diagonal elements, which are fixed to  $1/N$ . The attention mask is calculated as follows:

$$\begin{aligned} \mathbf{A} &= \mathcal{G}(\mathbf{Z}) \\ \mathbf{Z} &= \mathbf{S}\mathbf{W} \end{aligned} \quad (86)$$

where  $\mathcal{G}(\mathbf{Z})$  denotes the soft-max function that is applied to every row of  $\mathbf{Z}$ . That is, every element of  $\mathbf{A}$  is non-negative, and each row of  $\mathbf{A}$  sums up to 1. Similar to other attention formulations [159, 12, 220], we use soft-max normalization to promote competitions between different columns of  $\mathbf{Z}$ .

As mentioned previously, the weight matrix  $\mathbf{W}$  is used to measure the relative importance between columns of  $\mathbf{S}$ , which is encoded in  $\mathbf{Z}$ , and thus  $\mathbf{A}$ . In order to see this, let us denote by  $\mathbf{s}_n \in \mathbb{R}^M$  and  $\mathbf{z}_n \in \mathbb{R}^M$  the  $n$ -th column of  $\mathbf{S}$  and  $\mathbf{Z}$ , respectively. Since  $\mathbf{Z} = \mathbf{S}\mathbf{W}$ , the  $n$ -th column of  $\mathbf{Z}$ , i.e.,  $\mathbf{z}_n$ , is calculated as the weighted combination of  $N$  columns of  $\mathbf{S}$ , with the weight of the  $n$ -th column always equal to  $1/N$  since the diagonal elements of  $\mathbf{W}$  are fixed to  $1/N$ . In this way, element  $z_{mn}$  (in  $\mathbf{Z}$ ) encodes the relative importance of  $s_{mn}$  (in  $\mathbf{S}$ ) with respect to other  $s_{mk}$ , for  $k \neq n$ .

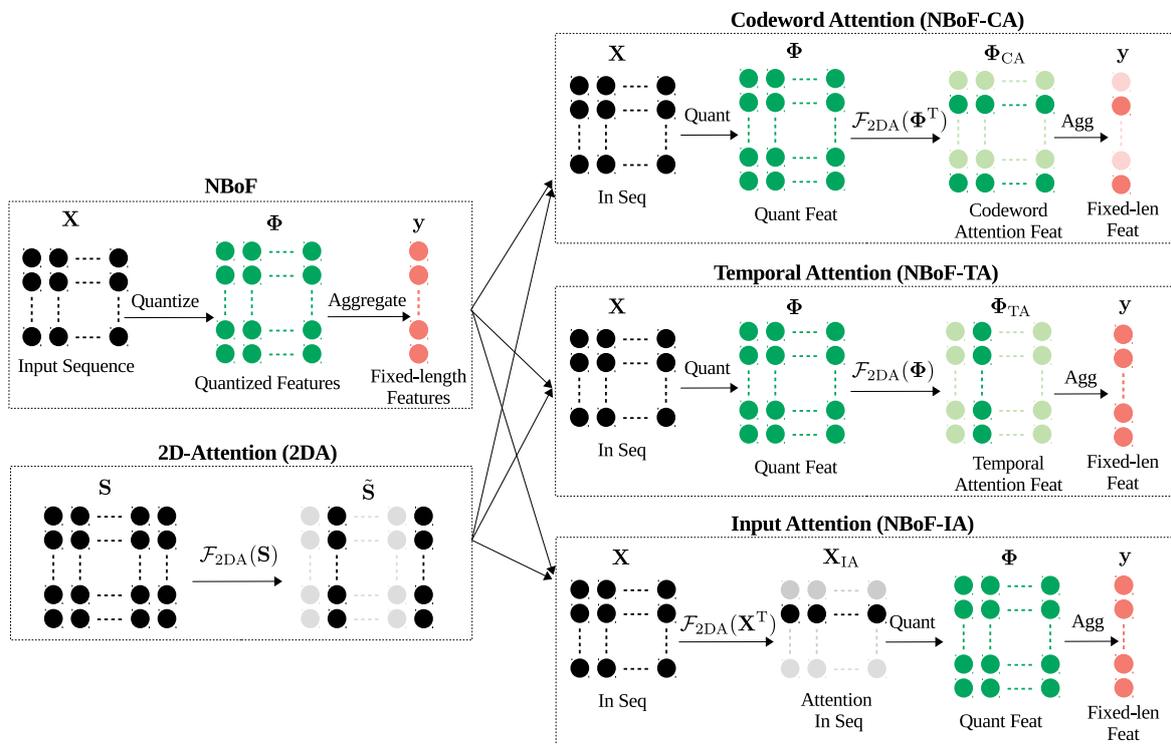


Figure 21: Illustration of the proposed attention formulation (2DA) and different attention-based NBoF models

### Attention-based Neural Bag-of-Features Learning

We incorporate the proposed 2DA attention mechanism into the Neural Bag-of-Features (NBoF) model in order to achieve different types of attention models:

- **Codeword Attention:** in the NBoF model, quantization results produced by each quantization neuron (codeword) are considered equally important for every input sequence. This property limits the feature quantization step to fully take advantage of the data-driven approach. In order to overcome this limitation, the proposed 2DA block can be applied to the quantized features to highlight or discard the outputs of certain quantization neurons. By doing so, the NBoF model is explicitly encouraged to learn a subset of specialized codewords for a given input pattern.

Particularly, given the quantized features denoted by  $\Phi \in \mathbb{R}^{K \times N}$ , we propose to apply attention to the rows of  $\Phi$  because the first mode of  $\Phi$  with dimension  $K$  denotes the number of quantization neurons or codewords. Since 2DA operates on the columns of the input matrix, the attention-based quantized features is calculated as follows:

$$\Phi_{CA} = \mathcal{F}_{2DA}(\Phi^T) \quad (87)$$

where  $\Phi^T$  denotes the transpose of  $\Phi$ .

- **Temporal Attention:** another limitation of the NBoF model lies in the aggregation step. In order to produce a fixed-length representation of the input sequence, the aggregation step in the NBoF model simply computes the mean of quantized features along the temporal mode. In this way, the NBoF model only allows equal contributions of all quantized features, disregarding the temporal information. Using our proposed 2DA formulation, it is straightforward to enable the NBoF model to attend to salient temporal information as follows:

$$\Phi_{TA} = \mathcal{F}_{2DA}(\Phi) \quad (88)$$

Since each column of  $\Phi$  contains quantized features of each time step, to obtain temporal attention-based features  $\Phi_{TA}$  we simply apply  $\mathcal{F}_{2DA}$  to  $\Phi$  as in Eq. (88).  $\Phi_{TA}$  is then averaged along the second dimension to produce the fixed-length representation of the input sequence. Although we still perform averaging in the aggregation step, the fixed-length representation is no longer the average of the quantized features, but a weighted average. This is because each time instance (column) in  $\Phi_{TA}$  has been scaled by different factors via the attention mechanism.

- **Input Attention:** noisy data is an inherent problem in many real-world applications. Noises might surface during the data acquisition process, such as ambient noise in audio signals or power line interference and motion artifacts in Electrocardiogram signals. In other scenarios, noises are inherent in the problem formulation since the relevance between the input sources and the targets might be unclear.

The proposed attention mechanism can also be used to filter out potential noisy series in a multivariate series as follows:

$$\mathbf{X}_{IA} = \mathcal{F}_{2DA}(\mathbf{X}^T) \quad (89)$$

where  $\mathbf{X} \in \mathbb{R}^{D \times N}$  denotes an input sequence of  $N$  steps of the NBoF model. Since we would like to apply attention over the individual series (rows of  $\mathbf{X}$ ),  $\mathcal{F}_{2DA}$  is applied to the transpose of  $\mathbf{X}$ .

The proposed attention-based NBoF models are illustrated in Figure 21.

### 5.2.3 Performance evaluation

The proposed attention-based NBoF models can be used for any sequence learning problems such as financial time-series, audio signals or biomedical signals. The experiment protocols, detailed description and analysis of our experiments can be found in the Appendix 7.14. In this

Table 40: Performance (mean of sensitivity and specificity) on PCG dataset. The higher, the better.

Models	Anomaly Detection	Quality Detection
GRU [30]	<b>90.08</b> $\pm$ 00.68	<b>72.74</b> $\pm$ 01.40
NBoF [155]	50.31 $\pm$ 00.42	49.69 $\pm$ 00.34
NBoF-CA (our)	88.09 $\pm$ 00.25	71.98 $\pm$ 03.00
NBoF-TA (our)	89.32 $\pm$ 01.02	72.57 $\pm$ 02.20
TNBoF [154]	54.12 $\pm$ 05.18	53.40 $\pm$ 04.21
TNBoF-CA (our)	88.68 $\pm$ 00.95	72.34 $\pm$ 01.32
TNBoF-TA (our)	88.81 $\pm$ 01.07	69.45 $\pm$ 00.89

section, we present the experiment results indicating the effectiveness of the proposed attention-based models in detecting abnormalities in the heart.

For this purpose, we evaluated the proposed attention models on two popular heart signals PCG and ECG, corresponding to two different datasets that were used in Physionet/Computing in Cardiology Challenge in 2016 and 2017, respectively:

- The PCG dataset used in our experiments come from the training set provided in the Physionet/Computing in Cardiology Challenge 2016 [33]. The objective of the challenge is to develop an automatic classification method for the anomaly (normal versus abnormal) and quality (good versus bad) detection given a PCG recording.
- The AF dataset focuses on the problem of atrial fibrillation detection from ECG recordings, which are provided as the development data (training set) in the Physionet/Computing in Cardiology Challenge 2017 [32]. The dataset contains 8528 single-lead ECG recordings lasting from 9 to 60 seconds. The objective of the challenge was to classify a given recording into one of the 4 classes: normal sinus rhythm, atrial fibrillation, alternative rhythm, and noise.

Table 41: Performance (averaged F1) on AF dataset

Models	F1
GRU [30]	76.42 $\pm$ 00.86
NBoF [155]	78.15 $\pm$ 00.82
NBoF-CA (our)	78.73 $\pm$ 00.71
NBoF-TA (our)	78.55 $\pm$ 00.90
TNBoF [154]	78.27 $\pm$ 01.02
TNBoF-CA (our)	78.71 $\pm$ 00.92
TNBoF-TA (our)	<b>79.52</b> $\pm$ 00.81

The detection results for the PCG dataset and the AF dataset are shown in Table 40 and 41, respectively. The suffices “-CA” and “-TA” refer to the codebook attention and temporal attention variants, respectively. The TNBoF model [154] is a variant of the original NBoF model that uses two codebooks for short and long-term temporal information extraction. As we can

see from Table 40, the state-of-the-arts GRU model performs slightly better than the temporal attention-based NBoF model (NBoF-TA) in both anomaly detection and signal quality detection in the PCG dataset. However, the attention-based models perform much better than the GRU model on the AF dataset in classifying atrial fibrillation. From both datasets, it is clear that by using attention mechanism, the NBoF and TNBoF model obtain significant performance gains.

## 6 Conclusions

AUTH worked towards objective O1 by developing highly-optimized and lightweight implementations for face and person detection (Section 2.1), face recognition (Section 2.2), as well as human pose estimation (Section 2.3). Furthermore, an adaptive inference approach fulfilling the strict speed and latency requirements that exist in many robotics applications was developed towards objective O1a (Section 2.5). A structured methodology for training and deploying regularized lightweight deep convolutional neural network models, capable of operating in real-time (Objective O1b) on embedded devices for high-resolution video input was also proposed (Section 2.4). Finally, several methods were developed towards objective O2b, including face recognition using an embedding-based approach (Section 2.6), as well as using synthesized facial views (Section 2.8), along with a realistic simulation environment for training active perception methods (Section 2.7) and a Webots-based interface for reducing development time when developing active perception-based methods (Section 2.10).

AU worked towards objective O1 by evaluating efficient lightweight implementations for video-based human activity recognition (Section 3.1). Furthermore, efficient deep learning solutions for skeleton-based human activity recognition were developed focusing on two aspects, i.e. automatic data selection and efficient neural network architectures. For automatic selection of the most informative body skeletons in a sequence through a learnable attention mechanism was developed (Section 3.2). For improving the inference speed through efficient network topologies, a method for automatic determination of a compact neural network architecture following a data-driven process leads to a reduced number of parameters needed to process the skeletons in the input sequence was developed (Section 3.3). Moreover, a network formed by bilinear spatio-temporal layers was developed (Section 3.4). A processing pipeline to further exploit the benefits of the proposed methods for landmark-based facial expression recognition was developed (Section 4.1). Moreover, a probabilistic framework for class-specific identification/verification and classification problems was developed and applied in facial expression recognition (Section 4.2).

TAU worked towards objective O1 by developing an end-to-end learning framework that optimizes both the signal acquisition parameters as well as the inference model based on compressed measurements of facial images (Section 2.9). In addition, a surrogate performance indicator was developed for this framework to quickly rank different sensor configurations when calibrating the signal acquisition hardware of a given robotic application. Beside the contributions in compressive learning for face recognition, TAU also contributed in deep biosignal analysis (Section 5.2) by developing different attention mechanisms for the Neural Bag-of-Feature models to highlight salient components from the heart signal to detect heart anomaly. Finally, within the speech recognition area, TAU also developed a novel type of lightweight neural layer by applying the quadratic kernel within the self organized operational neural network (Self-ONN) approach (5.1). The proposed operation has demonstrated the potential to surpass the performance of traditional convolution under computational constraints.

## References

- [1] TensorFlow Speech Recognition Challenge.
- [2] Synthetic Speech Commands Dataset, 2017.
- [3] J. H. Abawajy, A. V. Kelarev, and M. Chowdhury. Multistage approach for clustering and classification of ecg data. *Computer methods and programs in biomedicine*, 112(3):720–730, 2013.
- [4] Actin. Industrial-Grade Software for Advanced Robotic Solutions.
- [5] A. Adler, M. Elad, and M. Zibulevsky. Compressed learning: A deep neural network approach. *arXiv preprint arXiv:1610.09615*, 2016.
- [6] M. Aharon, M. Elad, A. Bruckstein, et al. K-svd: An algorithm for designing over-complete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311, 2006.
- [7] Y. Aloimonos. *Active perception*. Psychology Press, 2013.
- [8] P. Ammirato, P. Poirson, E. Park, J. Košecká, and A. C. Berg. A dataset for developing and benchmarking active vision. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1378–1385, 2017.
- [9] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.
- [10] S. Arashloo and J. Kittler. Class-specific kernel fusion of multiple descriptors for face verification using multiscale binarized statistical image features. *IEEE Transactions on Information Forensics and Security*, 9(12):2100–2109, 2014.
- [11] A. Asthana, S. Zafeiriou, S. Cheng, and M. Pantic. Incremental face alignment in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.
- [12] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
- [13] P. K. Baheti and M. A. Neifeld. Adaptive feature-specific imaging: a face recognition example. *Applied optics*, 47(10):B21–B31, 2008.
- [14] Y. Bai, S. S. Bhattacharyya, A. P. Happonen, and H. Huttunen. Elastic neural networks: A scalable framework for embedded computer vision. In *Proc. European Signal Processing Conference*, pages 1472–1476, 2018.
- [15] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos. Revisiting active perception. *Autonomous Robots*, 42(2):177–196, 2018.
- [16] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch. Emergent tool use from multi-agent autotutorials. *arXiv preprint arXiv:1909.07528*, 2019.

- [17] J. Behrmann, W. Grathwohl, R. T. Q. Chen, D. Duvenaud, and J.-H. Jacobsen. Invertible residual networks, 2019.
- [18] S. Bhattacharya, R. Sukthankar, R. Jin, and M. Shah. A probabilistic representation for efficient large scale visual recognition tasks. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 2593–2600, 2011.
- [19] V. Blanz, P. Grother, P. J. Phillips, and T. Vetter. Face recognition based on frontal views generated from non-frontal images. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 454–461. IEEE, 2005.
- [20] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016.
- [21] A. Bulat, J. Kossaiifi, G. Tzimiropoulos, and M. Pantic. Toward fast and accurate human pose estimation via soft-gated skip connections. *arXiv preprint arXiv:2002.11098*, 2020.
- [22] R. Calderbank and S. Jafarpour. Finding needles in compressed haystacks. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3441–3444. IEEE, 2012.
- [23] E. J. Candes, J. K. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(8):1207–1223, 2006.
- [24] E. J. Candès and M. B. Wakin. An introduction to compressive sampling [a sensing/sampling paradigm that goes against the common knowledge in data acquisition]. *IEEE signal processing magazine*, 25(2):21–30, 2008.
- [25] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. Vggface2: A dataset for recognising faces across pose and age, 2018.
- [26] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.
- [27] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733, 2017.
- [28] R. Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.
- [29] S. Chen, Y. Liu, X. Gao, and Z. Han. Mobilefacenets: Efficient cnns for accurate real-time face verification on mobile devices, 2018.
- [30] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- [31] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

- [32] G. D. Clifford, C. Liu, B. Moody, H. L. Li-wei, I. Silva, Q. Li, A. Johnson, and R. G. Mark. Af classification from a short single lead ecg recording: the physionet/computing in cardiology challenge 2017. In *2017 Computing in Cardiology (CinC)*, pages 1–4. IEEE, 2017.
- [33] G. D. Clifford, C. Liu, B. Moody, D. Springer, I. Silva, Q. Li, and R. G. Mark. Classification of normal/abnormal heart sound recordings: The physionet/computing in cardiology challenge 2016. In *2016 Computing in Cardiology Conference (CinC)*, pages 609–612. IEEE, 2016.
- [34] R. Cunha, M. Malaca, V. Sampaio, B. Guerreiro, P. Nousi, I. Mademlis, A. Tefas, and I. Pitas. Gimbal control for vision-based target tracking. In *Proc. European Conference on Signal Processing*, 2019.
- [35] M. A. Davenport, P. Boufounos, M. B. Wakin, R. G. Baraniuk, et al. Signal processing with compressive measurements. *J. Sel. Topics Signal Processing*, 4(2):445–460, 2010.
- [36] M. A. Davenport, M. F. Duarte, M. B. Wakin, J. N. Laska, D. Takhar, K. F. Kelly, and R. G. Baraniuk. The smashed filter for compressive classification and target recognition. In *Computational Imaging V*, volume 6498, page 64980H. International Society for Optics and Photonics, 2007.
- [37] L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.
- [38] B. Delaunay et al. Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7(793-800):1–2, 1934.
- [39] J. Delmerico, S. Isler, R. Sabzevari, and D. Scaramuzza. A comparison of volumetric information gain metrics for active 3d object reconstruction. *Autonomous Robots*, 42(2):197–208, 2018.
- [40] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
- [41] J. Deng, J. Guo, E. Ververas, I. Kotsia, and S. Zafeiriou. Retinaface: Single-shot multi-level face localisation in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5203–5212, 2020.
- [42] J. Deng, J. Guo, N. Xue, and S. Zafeiriou. Arcface: Additive angular margin loss for deep face recognition, 2019.
- [43] A. Dhall, R. Goecke, S. Ghosh, J. Joshi, J. Hoey, and T. Gedeon. From individual to group-level emotion recognition: Emotiw 5.0. In *ACM International Conference on Multimodal Interaction*, 2017.
- [44] R. Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute, August 2010.

- [45] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: A benchmark. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 304–311. IEEE, 2009.
- [46] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, T. Darrell, and K. Saenko. Long-term recurrent convolutional networks for visual recognition and description. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2625–2634, 2015.
- [47] X. Dong, Y. Yan, W. Ouyang, and Y. Yang. Style aggregated network for facial landmark detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
- [48] D. L. Donoho et al. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.
- [49] Y. Du, W. Wang, and L. Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *IEEE conference on computer vision and pattern recognition*, pages 1110–1118, 2015.
- [50] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6569–6578, 2019.
- [51] R. Duda, P. Hart, and D. Stork. *Pattern Classification, 2nd ed.* Wiley-Interscience, 2000.
- [52] G. Dulac-Arnold, D. J. Mankowitz, and T. Hester. Challenges of real-world reinforcement learning. *CoRR*, abs/1904.12901, 2019.
- [53] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.
- [54] C. Feichtenhofer. X3d: Expanding architectures for efficient video recognition. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [55] D. Ferigo, S. Traversaro, G. Metta, and D. Pucci. Gym-ignition: Reproducible robotic simulations for reinforcement learning, 2019.
- [56] C. Forster, M. Pizzoli, and D. Scaramuzza. Appearance-based active, monocular, dense reconstruction for micro aerial vehicles. 2014.
- [57] X. Gao, W. Hu, J. Tang, J. Liu, and Z. Guo. Optimized skeleton-based action recognition via sparsified graph regression. In *ACM International Conference on Multimedia*, pages 601–610, 2019.
- [58] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell. Compact bilinear pooling. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [59] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

- [60] A. Geitgey. Machine learning is fun! part 4: Modern face recognition with deep learning. <https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cff> 2016. [Online; accessed 16-October-2020].
- [61] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.
- [62] P. Gonçalves, P. Torres, C. Alves, F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli. The e-puck, a robot designed for education in engineering. *Proc. Conf. on Autonomous Robot Systems and Competitions*, 1, 01 2009.
- [63] P. Goyal and H. Kaiming. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*, 39:2999–3007, 2018.
- [64] L. Grasedyck, D. Kressner, and C. Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*, 36(1):53–78, 2013.
- [65] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.
- [66] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, 2016.
- [67] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. volume 9907, pages 87–102, 10 2016.
- [68] F. Han, B. Reily, W. Hoff, and H. Zhang. Space-time representation of people based on 3d skeletal data: A review. *Computer Vision and Image Understanding*, 158:85–105, 2017.
- [69] X. Han, H. Liu, F. Sun, and X. Zhang. Active object detection with multistep action prediction using deep q-network. *IEEE Transactions on Industrial Informatics*, 15(6):3723–3731, 2019.
- [70] B. Hassani and M. H. Mahoor. Facial expression recognition using enhanced deep 3d convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017.
- [71] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.
- [72] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.
- [73] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

- [74] Y. He, D. Xu, L. Wu, M. Jian, S. Xiang, and C. Pan. Lffd: A light and fast face detector for edge devices. *arXiv preprint arXiv:1904.10633*, 2019.
- [75] R. S. Heffner and H. E. Heffner. Evolution of sound localization in mammals. In *The evolutionary biology of hearing*, pages 691–715. 1992.
- [76] N. Heidari and A. Iosifidis. On the spatial attention in spatio-temporal graph convolutional networks for skeleton-based human action recognition. *arXiv preprint arXiv:2011.03833*, 2020.
- [77] N. Heidari and A. Iosifidis. Progressive spatio-temporal graph convolutional network for skeleton-based human action recognition. *arXiv preprint arXiv:2011.05668*, 2020.
- [78] N. Heidari and A. Iosifidis. Temporal Attention-Augmented Graph Convolutional Network for Efficient Skeleton-Based Human Action Recognition. In *International Conference on Pattern Recognition*, 2020.
- [79] E. Hoffer and N. Ailon. Deep metric learning using triplet network, 2018.
- [80] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
- [81] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7132–7141, 2018.
- [82] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger. Multi-scale dense networks for resource efficient image classification. In *Proc. International Conference on Learning Representations*, 2018.
- [83] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.
- [84] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and; 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.
- [85] A. Iosifidis. Probabilistic class-specific discriminant analysis. *IEEE Access*, 8:183847–183855, 2020.
- [86] A. Iosifidis, M. Gabbouj, and P. Pekki. Class-specific nonlinear projections using class-specific kernel spaces. *IEEE International Conference on Big Data Science and Engineering*, pages 1–8, 2015.
- [87] A. Iosifidis, A. Tefas, and I. Pitas. On the optimal class representation in Linear Discriminant Analysis. *IEEE Transactions on Neural Networks and Learning Systems*, 24(9):1491–1497, 2013.
- [88] A. Iosifidis, A. Tefas, and I. Pitas. Discriminant bag of words based representation for human action recognition. *Pattern Recognition Letters*, 49:185–192, 2014.

- [89] A. Iosifidis, A. Tefas, and I. Pitas. Kernel reference discriminant analysis. *Pattern Recognition Letters*, 49:85–91, 2014.
- [90] A. Iosifidis, A. Tefas, and I. Pitas. Class-specific reference discriminant analysis with application in human behavior analysis. *IEEE Transactions on Human-Machine Systems*, 45(3):315–326, 2015.
- [91] S. Isler, R. Sabzevari, J. Delmerico, and D. Scaramuzza. An information gain formulation for active volumetric 3d reconstruction. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3477–3484. IEEE, 2016.
- [92] K. Janocha and W. M. Czarnecki. On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659*, 2017.
- [93] Y.-G. Jiang, C.-W. Ngo, and J. Yang. Towards optimal bag-of-features for object categorization and semantic video retrieval. In *Proc. ACM International Conference on Image and Video Retrieval*, pages 494–501, 2007.
- [94] Joe Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4694–4702, 2015.
- [95] H. Jung, S. Lee, J. Yim, S. Park, and J. Kim. Joint fine-tuning in deep neural networks for facial expression recognition. In *IEEE International Conference on Computer Vision*, 2015.
- [96] Y. Kartynnik, A. Ablavatski, I. Grishchenko, and M. Grundmann. Real-time facial surface geometry from monocular video on mobile gpu. *arXiv preprint arXiv:1907.06724*, 2019.
- [97] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.
- [98] Q. Ke, M. Bennamoun, S. An, F. Sohel, and F. Boussaid. A new representation of skeleton sequences for 3d action recognition. In *IEEE conference on computer vision and pattern recognition*, pages 3288–3297, 2017.
- [99] J.-H. Kim, J. Jun, and B.-T. Zhang. Bilinear attention networks. In *Advances in Neural Information Processing Systems*, 2018.
- [100] T. S. Kim and A. Reiter. Interpretable 3d human action analysis with temporal convolutional networks. In *IEEE conference on computer vision and pattern recognition workshops*, pages 1623–1631. IEEE, 2017.
- [101] S. Kiranyaz, T. Ince, A. Iosifidis, and M. Gabbouj. Operational Neural Networks. *arXiv:1902.11106*, 2019.
- [102] S. Kiranyaz, J. Malik, H. B. Abdallah, T. Ince, A. Iosifidis, and M. Gabbouj. Self-Organized Operational Neural Networks with Generative Neurons. *arXiv preprint arXiv:2004.11778*, 2020.

- [103] M. Kirtas, K. Tsampazis, N. Passalis, and A. Tefas. Deepbots: A webots-based deep reinforcement learning framework for robotics. In I. Maglogiannis, L. Iliadis, and E. Pimenidis, editors, *Artificial Intelligence Applications and Innovations*, pages 64–75, Cham, 2020. Springer International Publishing.
- [104] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Proc. 2004 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, volume 3, pages 2149–2154, 2004.
- [105] D. Kollias and S. P. Zafeiriou. Exploiting multi-cnn features in cnn-rnn based dimensional emotion recognition on the omg in-the-wild dataset. *IEEE Transactions on Affective Computing*, pages 1–1, 2020.
- [106] M. Krestenitis, N. Passalis, A. Iosifidis, M. Gabbouj, and A. Tefas. Human action recognition using recurrent bag-of-features pooling. In *Proc. International Conference on Pattern Recognition*, 2020.
- [107] M. Krestenitis, N. Passalis, A. Iosifidis, M. Gabbouj, and A. Tefas. Recurrent bag-of-features for visual information analysis. *Pattern Recognition*, 2020.
- [108] O. Köpüklü, N. Kose, A. Gunduz, and G. Rigoll. Resource efficient 3d convolutional neural networks. In *IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 1910–1919, 2019.
- [109] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436, 2015.
- [110] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [111] B. Li, Y. Dai, X. Cheng, H. Chen, Y. Lin, and M. He. Skeleton based action recognition using translation-scale invariant image mapping and multi-scale deep cnn. In *IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 601–604. IEEE, 2017.
- [112] C. Li, Q. Zhong, D. Xie, and S. Pu. Skeleton-based action recognition with convolutional neural networks. In *IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 597–600. IEEE, 2017.
- [113] J. Li, W. Su, and Z. Wang. Simple pose: Rethinking and improving a bottom-up approach for multi-person pose estimation, 2019.
- [114] J. Li, Y. Wang, C. Wang, Y. Tai, J. Qian, J. Yang, C. Wang, J. Li, and F. Huang. Dsfed: dual shot face detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5060–5069, 2019.
- [115] J.-h. Li. Cyber security meets artificial intelligence: a survey. *Frontiers of Information Technology & Electronic Engineering*, 19(12):1462–1474, 2018.
- [116] L. Li, W. Zheng, Z. Zhang, Y. Huang, and L. Wang. Skeleton-based relational modeling for action recognition. *arXiv preprint arXiv:1805.02556*, 1(2):3, 2018.

- [117] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, and Q. Tian. Actional-structural graph convolutional networks for skeleton-based action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3595–3603, 2019.
- [118] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.
- [119] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.
- [120] H. Liu, J. Tu, and M. Liu. Two-stream 3d convolutional neural network for skeleton-based action recognition. *arXiv preprint arXiv:1705.08106*, 2017.
- [121] J. Liu, A. Shahroudy, D. Xu, and G. Wang. Spatio-temporal lstm with trust gates for 3d human action recognition. In *European conference on computer vision*, pages 816–833. Springer, 2016.
- [122] M. Liu, H. Liu, and C. Chen. Enhanced skeleton visualization for view invariant human action recognition. *Pattern Recognition*, 68:346–362, 2017.
- [123] S. Liu, D. Huang, et al. Receptive field block net for accurate and fast object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 385–400, 2018.
- [124] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.
- [125] W. Liu, S. Liao, W. Ren, W. Hu, and Y. Yu. High-level semantic feature detection: A new perspective for pedestrian detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5187–5196, 2019.
- [126] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. Sphereface: Deep hypersphere embedding for face recognition, 2018.
- [127] W. Liu, Y. Wen, Z. Yu, and M. Yang. Large-margin softmax loss for convolutional neural networks, 2017.
- [128] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.
- [129] L. Lo, H.-X. Xie, H.-H. Shuai, and W.-H. Cheng. Mer-gcn: Micro expression recognition based on relation modeling with graph convolutional network. *arXiv preprint arXiv:2004.08915*, 2020.
- [130] S. Lohit, K. Kulkarni, and P. Turaga. Direct inference on compressive measurements using convolutional neural networks. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 1913–1917. IEEE, 2016.

- [131] S. Lohit, K. Kulkarni, P. Turaga, J. Wang, and A. C. Sankaranarayanan. Reconstruction-free inference on compressive measurements. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–24, 2015.
- [132] N. G. Lopez, Y. L. E. Nuin, E. B. Moral, L. U. S. Juan, A. S. Rueda, V. M. Vilches, and R. Kojcev. gym-gazebo2, a toolkit for reinforcement learning using ros 2 and gazebo, 2019.
- [133] G. Loukas, T. Vuong, R. Heartfield, G. Sakellari, Y. Yoon, and D. Gan. Cloud-based cyber-physical intrusion detection for vehicles using deep learning. *IEEE Access*, 6:3491–3508, 2017.
- [134] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2010.
- [135] M. Mendoza, J. I. Vasquez-Gomez, H. Taud, L. E. Sucar, and C. Reta. Supervised learning of the next-best-view for 3d object reconstruction. *Pattern Recognition Letters*, 2020.
- [136] O. Michel. Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems*, 1(1):39–42, 2004.
- [137] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *arXiv preprint arXiv:2003.08934*, 2020.
- [138] A. Mollahosseini, D. Chan, and M. H. Mahoor. Going deeper in facial expression recognition using deep neural networks. In *IEEE Winter Conference on Applications of Computer Vision*, 2016.
- [139] S. Moschoglou, A. Papaioannou, C. Sagonas, J. Deng, I. Kotsia, and S. Zafeiriou. Agedb: the first manually collected, in-the-wild age database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop*, volume 2, page 5, 2017.
- [140] M. Nakada, H. Wang, and D. Terzopoulos. Acfr: Active face recognition using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 35–40, 2017.
- [141] Q. T. Ngoc, S. Lee, and B. C. Song. Facial landmark-based emotion recognition via directed graph neural network. *Electronics*, 9(5):764, 2020.
- [142] P. Nousi, D. Triantafyllidou, A. Tefas, and I. Pitas. Joint lightweight object tracking and detection for unmanned vehicles. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 160–164. IEEE, 2019.
- [143] D. Osokin. Real-time 2d multi-person pose estimation on cpu: Lightweight openpose. In *arXiv preprint arXiv:1811.12004*, 2018.
- [144] S. Pancoast and M. Akbacak. Bag-of-audio-words approach for multimedia event classification. In *Proc. Annual Conference of the International Speech Communication Association*, 2012.

- [145] M. Pantic, M. Valstar, R. Rademaker, and L. Maat. Web-based database for facial expression analysis. In *IEEE International Conference on Multimedia and Expo*, 2005.
- [146] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *BMVC*, 2015.
- [147] N. Passalis, J. Raitoharju, M. Gabbouj, and A. Tefas. Efficient adaptive inference leveraging bag-of-features-based early exits. In *Proc. IEEE International Workshop on Multimedia Signal Processing*, 2020.
- [148] N. Passalis, J. Raitoharju, A. Tefas, and M. Gabbouj. Efficient adaptive inference for deep convolutional neural networks using hierarchical early exits. *Pattern Recognition*, 2020.
- [149] N. Passalis and A. Tefas. Learning bag-of-features pooling for deep convolutional neural networks. In *Proc. IEEE International Conference on Computer Vision*, pages 5755–5763, 2017.
- [150] N. Passalis and A. Tefas. Neural bag-of-features learning. *Pattern Recognition*, 64:277–294, 2017.
- [151] N. Passalis and A. Tefas. Continuous drone control using deep reinforcement learning for frontal view person shooting. *Neural Computing and Applications*, pages 1–12, 2019.
- [152] N. Passalis and A. Tefas. Deep reinforcement learning for controlling frontal person close-up shooting. *Neurocomputing*, 335:37–47, 2019.
- [153] N. Passalis and A. Tefas. Leveraging active perception for improving embedding-based deep face recognition. In *Proc. IEEE International Workshop on Multimedia Signal Processing*, 2020.
- [154] N. Passalis, A. Tefas, J. Kannianen, M. Gabbouj, and A. Iosifidis. Temporal bag-of-features learning for predicting mid price movements using high frequency limit order book data. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2018.
- [155] N. Passalis, A. Tefas, J. Kannianen, M. Gabbouj, and A. Iosifidis. Temporal logistic neural bag-of-features for financial time series forecasting leveraging limit order book data. *arXiv preprint arXiv:1901.08280*, 2019.
- [156] N. Passalis, A. Tefas, and I. Pitas. Efficient camera control using 2d visual information for unmanned aerial vehicle-based cinematography. In *Proc. IEEE International Symposium on Circuits and Systems*, pages 1–5, 2018.
- [157] W. Peng, X. Hong, H. Chen, and G. Zhao. Learning graph convolutional network for skeleton-based human action recognition by neural searching. In *AAAI conference on artificial intelligence*, pages 2669–2676, 2020.
- [158] N. Pinto, Z. Stone, T. Zickler, and D. Cox. Scaling up biologically-inspired computer vision: A case study in unconstrained face recognition on facebook. In *CVPR 2011 WORKSHOPS*, pages 35–42. IEEE, 2011.

- [159] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. *arXiv preprint arXiv:1704.02971*, 2017.
- [160] S. K. Ramakrishnan and K. Grauman. Sidekick policy learning for active visual exploration. In *Proc. European Conference on Computer Vision*, pages 413–430, 2018.
- [161] H. Reboredo, F. Renna, R. Calderbank, and M. R. Rodrigues. Compressive classification. In *2013 IEEE International Symposium on Information Theory*, pages 674–678. IEEE, 2013.
- [162] H. Reboredo, F. Renna, R. Calderbank, and M. R. Rodrigues. Projections designs for compressive classification. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 1029–1032. IEEE, 2013.
- [163] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.
- [164] S. Ren, X. Cao, Y. Wei, and J. Sun. Face alignment at 3000 fps via regressing local binary features. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.
- [165] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.
- [166] A. H. Ribeiro, M. H. Ribeiro, G. M. Paixão, D. M. Oliveira, P. R. Gomes, J. A. Canazart, M. P. Ferreira, C. R. Andersson, P. W. Macfarlane, M. Wagner Jr, et al. Automatic diagnosis of the 12-lead ecg using a deep neural network. *Nature communications*, 11(1):1–9, 2020.
- [167] A. Richard and J. Gall. A bag-of-words equivalent recurrent neural network for action recognition. *Computer Vision and Image Understanding*, 156:79–91, Mar 2017.
- [168] C. C. V. P. R. C. D. J. S. Sengupta, J.C. Cheng. Frontal to profile face verification in the wild. In *IEEE Conference on Applications of Computer Vision*, 2016.
- [169] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.
- [170] C. Schenck and D. Fox. Visual closed-loop control for pouring liquids. In *Proc. IEEE International Conference on Robotics and Automation*, pages 2629–2636, 2017.
- [171] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.
- [172] D. W. Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.
- [173] S. Shah, D. Dey, C. Lovett, and A. Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017.

- [174] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang. Ntu rgb+ d: A large scale dataset for 3d human activity analysis. In *IEEE conference on computer vision and pattern recognition*, pages 1010–1019, 2016.
- [175] M. Shen and J. P. How. Active perception in adversarial scenarios using maximum entropy deep reinforcement learning. In *Proc. International Conference on Robotics and Automation*, pages 3384–3390, 2019.
- [176] L. Shi, Y. Zhang, J. Cheng, and H. Lu. Skeleton-based action recognition with directed graph neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7912–7921, 2019.
- [177] L. Shi, Y. Zhang, J. Cheng, and H. Lu. Skeleton-based action recognition with directed graph neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [178] L. Shi, Y. Zhang, J. Cheng, and H. Lu. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 12026–12035, 2019.
- [179] S. Singh, D. Singh, and V. Yadav. Face recognition using hog feature extraction and svm classifier. *International Journal*, 8(9), 2020.
- [180] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proc. the International Conference on Computer Vision*, page 1470, 2003.
- [181] M. Soltanian, J. Malik, J. Raitoharju, A. Iosifidis, S. Kiranyaz, and M. Gabbouj. Speech command recognition in computationally constrained environments with a quadratic self-organized operational layer. *arXiv:2011.11436*, 2020.
- [182] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu. An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In *AAAI conference on artificial intelligence*, 2017.
- [183] L. Souza, B. Gatto, J. Xue, and K. Fukui. Enhanced grassmann discriminant analysis with randomized time warping for motion recognition. *Pattern Recognition*, 97(1):107028, 2020.
- [184] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [185] D. Steckelmacher, H. Plisnier, D. M. Roijers, and A. Nowé. Sample-efficient model-free reinforcement learning with off-policy critics, 2019.
- [186] Y. Sun, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification, 2014.
- [187] M. Tan and Q. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. volume 97 of *Proceedings of Machine Learning Research (PMLR)*, pages 6105–6114, 09–15 Jun 2019.

- [188] M. Tan, R. Pang, and Q. V. Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10781–10790, 2020.
- [189] X. Tang, D. K. Du, Z. He, and J. Liu. Pyramidbox: A context-assisted single shot face detector. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 797–813, 2018.
- [190] Y. Tang. Deep learning using linear support vector machines. *arXiv preprint arXiv:1306.0239*, 2013.
- [191] Y. Tang, Y. Tian, J. Lu, P. Li, and J. Zhou. Deep progressive reinforcement learning for skeleton-based action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5323–5332, 2018.
- [192] S. Teerapittayanon, B. McDanel, and H.-T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *Proc. International Conference on Pattern Recognition*, pages 2464–2469, 2016.
- [193] K. Torkkola. Feature extraction by non-parametric mutual information maximization. *Journal of Machine Learning Research*, 3(Mar):1415–1438, 2003.
- [194] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, 2015.
- [195] D. Tran, M. Gabbouj, and A. Iosifidis. Multilinear class-specific discriminant analysis. *Pattern Recognition Letters*, 100:131–136, 2017.
- [196] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A closer look at spatiotemporal convolutions for action recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6450–6459, 2018.
- [197] D. T. Tran, M. Gabbouj, and A. Iosifidis. Multilinear compressive learning with prior knowledge. *arXiv preprint arXiv:2002.07203*, 2020.
- [198] D. T. Tran, M. Gabbouj, and A. Iosifidis. Performance indicator in multilinear compressive learning. In *2020 IEEE Symposium Series on Computational Intelligence*. IEEE, 2020.
- [199] D. T. Tran, A. Iosifidis, J. Kannianen, and M. Gabbouj. Temporal attention-augmented bilinear network for financial time-series data analysis. *IEEE Transactions on Neural Networks and Learning Systems*, 30(5):1407–1418, 2018.
- [200] D. T. Tran, S. Kiranyaz, M. Gabbouj, and A. Iosifidis. Heterogeneous multilayer generalized operational perceptron. *IEEE Transactions on Neural Networks and Learning Systems*, 31(3):710–724, 2019.
- [201] D. T. Tran, N. Passalis, A. Tefas, M. Gabbouj, and A. Iosifidis. Attention-based neural bag-of-features learning for sequence data. *arXiv preprint arXiv:2005.12250*, 2020.

- [202] D. T. Tran, M. Yamac, A. Degerli, M. Gabbouj, and A. Iosifidis. Multilinear compressive learning. *IEEE Transactions on Neural Networks and Learning Systems (2020) in press*, 2020.
- [203] I. Triguero, S. García, and F. Herrera. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information systems*, 42(2):245–284, 2015.
- [204] M. Tzelepi and A. Tefas. Class-specific discriminant regularization in real-time deep cnn models for binary classification problems. *Neural Processing Letters*, 51(2):1989–2005, 2020.
- [205] M. Tzelepi and A. Tefas. Improving the performance of lightweight cnns for binary classification using quadratic mutual information regularization. *Pattern Recognition*, page 107407, 2020.
- [206] M. Tzelepi and A. Tefas. Quadratic mutual information regularization in real-time deep cnn models. In *Proc. IEEE International Workshop on Machine Learning for Signal Processing*, 2020.
- [207] C. Tzelepis, V. Mezaris, and I. Patras. Linear maximum margin classifier for learning from uncertain data. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2948–2962, 2018.
- [208] A. Tzimas, N. Passalis, and A. Tefas. Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search. In *Proc. IEEE International Conference on Multimedia and Expo*, pages 1–6, 2020.
- [209] V. N. Vapnik and V. Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.
- [210] J. I. Vasquez-Gomez and D. E. T. Romero. Next-best-view regression using a 3d convolutional neural network. *MSc Thesis*, 2019.
- [211] E. Vazquez-Fernandez, A. Dacal-Nieto, F. Martin, and S. Torres-Guijarro. Entropy of gabor filtering for image quality assessment. In *International Conference Image Analysis and Recognition*, pages 52–61. Springer, 2010.
- [212] A. Veit and S. Belongie. Convolutional networks with adaptive inference graphs. In *Proc. European Conference on Computer Vision*, pages 3–18, 2018.
- [213] F. Wang, J. Cheng, W. Liu, and H. Liu. Additive margin softmax for face verification. *IEEE Signal Processing Letters*, 25(7):926–930, Jul 2018.
- [214] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu. Cosface: Large margin cosine loss for deep face recognition, 2018.
- [215] X. Wang, T. Xiao, Y. Jiang, S. Shao, J. Sun, and C. Shen. Repulsion loss: Detecting pedestrians in a crowd. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7774–7783, 2018.

- [216] P. Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.
- [217] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese. Gibson env: Real-world perception for embodied agents. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 9068–9079, 2018.
- [218] J. Xiang and G. Zhu. Joint face detection and facial expression recognition with mtcnn. In *2017 4th International Conference on Information Science and Control Engineering (ICISCE)*, pages 424–427. IEEE, 2017.
- [219] X. Xiong and F. De la Torre. Supervised descent method and its applications to face alignment. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.
- [220] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.
- [221] S. Yan, Y. Xiong, and D. Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI conference on artificial intelligence*, 2018.
- [222] S. Yang, P. Luo, C.-C. Loy, and X. Tang. Wider face: A face detection benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5525–5533, 2016.
- [223] D. Yashunin, T. Baydasov, and R. Vlasov. Maskface: multi-task face and landmark detector. *arXiv preprint arXiv:2005.09412*, 2020.
- [224] H. Ye, Y. Li, C. Chen, and Z. Zhang. Fast fisher discriminant analysis with randomized algorithms. *Pattern Recognition*, 72(12):82–92, 2017.
- [225] S. Zafeiriou, G. Tzimiropoulos, M. Petrou, and T. Stathaki. Regularized kernel discriminant analysis with a robust kernel for face recognition and verification. *IEEE Transactions on Neural Networks*, 23(3):526–534, 2012.
- [226] I. Zamora, N. G. Lopez, V. M. Vilches, and A. H. Cordero. Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo, 2016.
- [227] B. Zhang, J. Li, Y. Wang, Y. Tai, C. Wang, J. Li, F. Huang, Y. Xia, W. Pei, and R. Ji. Asfd: Automatic and scalable face detector. *arXiv preprint arXiv:2003.11228*, 2020.
- [228] F. Zhang, X. Fan, G. Ai, J. Song, Y. Qin, and J. Wu. Accurate face detection for high performance. *arXiv preprint arXiv:1905.01585*, 2019.
- [229] F. Zhang, X. Zhu, H. Dai, M. Ye, and C. Zhu. Distribution-aware coordinate representation for human pose estimation. *arXiv preprint arXiv:1910.06278*, 2019.
- [230] P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue, and N. Zheng. View adaptive recurrent neural networks for high performance human action recognition from skeleton data. In *IEEE International Conference on Computer Vision*, pages 2117–2126, 2017.

- [231] S. Zhang, R. Benenson, and B. Schiele. Citypersons: A diverse dataset for pedestrian detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3221, 2017.
- [232] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li. Single-shot refinement neural network for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4203–4212, 2018.
- [233] S. Zhang, R. Zhu, X. Wang, H. Shi, T. Fu, S. Wang, T. Mei, and S. Z. Li. Improved selective refinement network for face detection. *arXiv preprint arXiv:1901.06651*, 2019.
- [234] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li. Faceboxes: A cpu real-time face detector with high accuracy. In *2017 IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–9. IEEE, 2017.
- [235] T. Zhang, G. Kahn, S. Levine, and P. Abbeel. Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search. In *Proc. IEEE International Conference on Robotics and Automation*, pages 528–535, 2016.
- [236] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018.
- [237] H. Zhou, J. Liu, Z. Liu, Y. Liu, and X. Wang. Rotate-and-render: Unsupervised photorealistic face rotation from single-view images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5911–5920, 2020.
- [238] Y. Zhou, Y. Bai, S. S. Bhattacharyya, and H. Huttunen. Elastic neural networks for classification. In *Proc. IEEE International Conference on Artificial Intelligence Circuits and Systems*, pages 251–255, 2019.
- [239] C. Zimmermann, T. Welschhold, C. Dornhege, W. Burgard, and T. Brox. 3d human pose estimation in rgb-d images for robotic task learning. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1986–1992, 2018.
- [240] E. Zisselman, A. Adler, and M. Elad. Compressed learning for image classification: A deep neural network approach. *Processing, Analyzing and Learning of Images, Shapes, and Forms*, 19:1, 2018.
- [241] B. Zoph and Q. V. Le. Neural Architecture Search with Reinforcement Learning. In *International Conference on Learning Representations*, 2017.
- [242] G. Zoumpourlis, A. Doumanoglou, N. Vretos, and P. Daras. Non-linear convolution filters for CNN-based learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4761–4769, 2017.

## **7 Appendix**

### **7.1 Efficient adaptive inference for deep convolutional neural networks using hierarchical early exits**

The appended paper follows.

# Efficient Adaptive Inference for Deep Convolutional Neural Networks using Hierarchical Early Exits

Nikolaos Passalis<sup>a</sup>, Jenni Raitoharju<sup>b</sup>, Anastasios Tefas<sup>a</sup>, Moncef Gabbouj<sup>b</sup>

<sup>a</sup>*Department of Informatics, Aristotle University of Thessaloniki, Greece*

<sup>b</sup>*Faculty of Information Technology and Communication Sciences, Tampere University, Finland*

---

## Abstract

Early exits are capable of providing deep learning models with adaptive computational graphs that can readily adapt on-the-fly to the available resources. Despite their advantages, existing early exit methods suffer from many limitations which limit their performance, e.g., they ignore the information extracted from previous exit layers, they are unable to efficiently handle feature maps with large sizes, etc. To overcome these limitations we propose a Bag-of-Features (BoF)-based method that is capable of constructing efficient hierarchical early exit layers with minimal computational overhead, while also providing an adaptive inference method that allows for early stopping the inference process when the network is confident enough for its output, leading to significant performance benefits. To this end, the BoF model is extended and adapted to the needs of early exits by constructing additive shared histogram spaces that gradually refine the information extracted from the various layers of a network, in a hierarchical manner, while also employing a classification layer reuse strategy to further reduce the number of parameters needed per exit layer. Note that the proposed method is generic and can be readily combined with any neural network architecture. The effectiveness of the proposed method is demonstrated using five different image datasets, proving that early exits can be readily transformed into a practical tool, which can be effectively used in various real-world embedded applications.

*Keywords:* Adaptive Inference, Early Exits, Bag-of-Features, Deep Convolutional Neural Networks, Hierarchical Representations

---

## 1. Introduction

The advent of Deep Learning (DL) led to spectacular applications, including but not limited to autonomous cars [1], accurate medical diagnosis and disease prognosis [2, 3], intelligent buildings [4], and powerful methods for human-computer interaction [5, 6]. Despite its enormous success in the  
5  
aforementioned domains, DL suffers from a significant drawback: DL models often consist of millions of parameters and, as a result, they require using powerful and energy-consuming hardware both

---

\*Corresponding author: Nikolaos Passalis, *Email address:* `passalis@csd.auth.gr`

for training and deployment. This limitation significantly reduces their flexibility, increases the deployment costs, and slows down the penetration of DL in many domains, where these requirements cannot be satisfied. Many methods have been proposed in the literature for overcoming the aforementioned drawbacks, ranging from quantization and model compression approaches [7] to neural network pruning [8] and knowledge distillation methods [9, 10, 11].

Even though these methods can indeed lead to developing faster and more lightweight DL models, they also often lead to models that perform worse than the original ones (in terms of accuracy). Furthermore, the developed models are *static*, i.e., they are unable to adapt to the available computational resources on the fly. However, in many applications there is a need for *dynamic* networks that are able to adapt on demand to the current conditions. This can be better understood by the following example. Consider a real-time face recognition system, where the faces that are depicted in an acquired frame must be accurately recognized under strict real time constraints. Note that the time that will be spent for the recognition process is proportional to the number of faces that appear in a given frame. If an estimate for the maximum number of persons that will appear in a frame is known beforehand, then we can appropriately design a lightweight DL model that will always work within the given time constraints. However, what will happen if more persons appear in a given frame? Most of the existing methods will spend the predefined portion of time for recognizing the depicted faces and then either stop without recognizing the rest of the people or spend more time than the allowed for the recognition, reducing the quality of service. On the other hand, a dynamic model would be able to adapt to the available load by reducing the time needed for the recognition of each person, possibly by providing slightly less refined and accurate predictions, meeting the real time constraints of the system.

Therefore, in such applications we need models that will be able to dynamically adapt to the available load, e.g., by being able to provide faster (and possibly less accurate) predictions when the load is higher (e.g., when a lot of persons appear in a frame) and more refined and accurate (yet slower) predictions when the load is lower (e.g., when only a few persons appear in a frame). It is worth noting that apart from this case, the need for dynamically adapting a model to the current computational resources without retraining occurs in many other applications, e.g., deploying mobile applications on smartphones where it is usually infeasible to train separate models according to the computational resources available to the vast variety of different phone models [12].

Among the most promising approaches for overcoming these limitations are DL models with adaptive computational graphs, such as [13, 14, 15]. These models provide an easy and straightforward way to dynamically adapt the model on-the-fly to the available computational resources by altering the complexity of the model's graph, i.e., by choosing a different computational path according to

the available resources. One straightforward way to achieve this is by adding early exits at various layers of the network [12, 13, 15]. These early exits allow for estimating the final output of the network at various points of the inference process, without having to feed-forward through the whole network. Early exits provide, in theory, a solid way to adapt the inference to the available resources. Unfortunately, they do not always lead to acceptable performance [15], since they have to deal with enormous feature maps (especially for the earlier layers of a convolutional neural network). To this end, aggressive subsampling methods are usually employed, e.g., global average pooling [16]. As a result, these methods ignore both the spatial information and the distribution of the extracted feature vectors, reducing the performance of early exits (in terms of accuracy). Even though this problem is partly addressed in [12] by using a series of densely connected structures, this also requires a significant number of structural changes in the architecture of a network and cannot be easily used with existing neural networks. Furthermore, most of the existing early exit-based approaches completely ignore the information extracted by the previous exit layers, throwing away information that is readily available and can be potentially used to further increase the prediction accuracy for the subsequent exit layers.

In this work we propose using the Bag-of-Features (BoF) model [17], for compiling compact, yet rich and discriminative representations from the feature maps of each layer where an early exit is used. The BoF model was originally proposed to provide compact representations for complex objects that consist of multiple feature vectors, while keeping as much information as possible about the corresponding object. More specifically, BoF works as follows: a) A feature extractor is used to extract feature vectors from an object, e.g., multiple SIFT vectors can be extracted from an image [18]. b) Then, the extracted feature vectors are quantized into a number of bins, defined by a set of vectors called *codewords*. c) Finally, a compact histogram representation is extracted for each object simply by counting the number of feature vectors that were quantized into each bin. In this way, BoF provides an efficient way to compress large collections of feature vectors, such as those extracted from the earlier layers of a neural network, into compact histogram representations that can overcome the limitations of the existing methods used for providing early exits.

However, using the BoF model for providing efficient early exit implementations is not straightforward. First, note that the histogram space compiled at each exit layer is different, which can prohibit the efficient construction of hierarchical representations that take into account the representations extracted from the previous early exits, significantly limiting the performance of the model. To overcome this limitation we propose extracting additive hierarchical histogram representations, that gradually refine the estimations from the previous layers, by implicitly constructing a shared histogram space for all the exit layers. Thus, the codewords are learned in such way that the histogram representations extracted from the various levels of the network are compatible with each other. This also allows for

75 having a common classification layer, which is shared among all the early exits, further reducing the number of parameters needed for adding an early exit to the network. Furthermore, the proposed method also supports dynamic adaptive inference that allows for stopping the inference process at an early exit, if the network is confident enough for its decision. This allows for spending less time for classifying easy input samples, while spending more time processing the harder ones. As it is demon-  
80 strated through this paper, these modifications lead to enormous improvements over both traditional static DL models, as well as over existing early exit approaches.

The main contribution of this work is proposing a Bag-of-Features-based approach, fully adapted to the needs of early exits, overcoming most of the limitations of existing related approaches. More specifically, in this paper:

- 85 1. A BoF-based formulation [19], which is capable of a) maintaining more information regarding the distribution of the extracted feature vectors and b) keeping more spatial information in the extracted representation, than the existing early exit methods, is proposed. Note that the latter is especially important for earlier exits, where the receptive field of the convolutional layers is usually smaller.
- 90 2. An efficient hierarchical aggregation scheme, that works by implicitly constructing a common histogram representation space and then gradually refining the estimations of the network, is proposed. This allows for taking into account the information that was already extracted by the earlier layers. Note that most of the existing formulations ignore this information [15].
3. A classification layer reuse approach is employed to further reduce the number of parameters  
95 required for each early exit, minimizing the cost of adding additional exit layers.
4. An adaptive classification approach, that allows for selecting the most appropriate early exit according to the difficulty of each input sample, is proposed, allowing for reducing the load to the system, as well as reducing the energy consumption of DL models.

It is worth noting that the proposed method can be readily combined with any neural network ar-  
100 chitecture, since it requires no model-specific changes to the base network. Finally, the ability of the proposed method to transform early exits into a practical tool, that can be applied in many challenging real-world applications, is demonstrated using extensive experiments on five datasets.

The rest of this paper is structured as follows. First, the related work is briefly discussed in Section 2, while the proposed method is analytically derived in Section 3. Then, the proposed method  
105 is extensively evaluated in Section 4, while conclusions are drawn in Section 5.

## 2. Related Work

This work is related to DL models with adaptive computational graphs, which allow for dynamically adapting the model to the available computational resources/difficulty of the input samples. Using early exits is perhaps the most straightforward way to provide DL models with adaptive computational graphs [14], as demonstrated in [12, 13, 15, 16]. BranchyNet [13] proposed to include exit layers at various points of the computational graph of the model by including branches that are composed of  $3 \times 3$  convolution layers, followed by the appropriate classification layers. A similar approach, which skips the added complexity of convolutional layers by employing global average pooling layers to downsample the input to the each early exit was further examined in Elastic Networks [15, 16]. However, these methods often do not lead to acceptable performance, since they discard valuable spatial information, as well as useful information regarding the distribution of the extracted feature vectors, as experimentally demonstrated in Section 4. Even though this problem is partly addressed in [12] by using a series of densely connected structures, this also requires a significant number of structural changes in the architecture of a network and cannot be easily used with existing neural networks, rendering this approach significantly harder to implement compared to the rest of the proposed early exit approaches.

In this work we provide a powerful method that can overcome these limitations by proposing a hierarchical BoF-based approach that can also keep all the information extracted from the various exit layers of a neural network with minimal additional overhead. Also, it is worth noting that early exits have been used in past for a different purpose: early exits were used for reinforcing the gradients at various layers, reducing in this way the training problems that are related to vanishing gradient phenomena [20]. This work is also related to the traditional dictionary learning methods for the BoF model, for which a very rich literature exists [21, 22, 23]. However, these approaches have not been designed to work with early exits, i.e., dealing with features extracted from various levels of a network, constructing common representation spaces for these features, using additive hierarchical representations and reusing the same classification layers at various points of the same network in order to provide efficient early exit implementations. In [24], the use of BoF-based pooling was examined for the first time in the context of early exits. However, compared to the proposed method, this approach leads to significantly larger intermediate representations and requires more parameters, especially when applied in a hierarchical setting, as also demonstrated in the sensitivity analysis provided in Section 4.

To the best of our knowledge, this is the first work in which a hierarchical BoF-based formulation is employed for constructing additive histogram representation spaces that can be used for providing efficient hierarchical early exits. At the same time, the proposed method also provides an efficient

140 adaptive inference mechanism that allows for effectively selecting the most appropriate early exit, given that the network is confident enough for its classification decision.

### 3. Proposed Method

First, the necessary background and notation are introduced in Subsection 3.1. Then, the proposed Bag-of-Features-based method for providing efficient early exit implementations is analytically derived in Subsection 3.2. Furthermore, a powerful, yet efficient hierarchical early exit scheme, that builds upon the proposed BoF method, is described in Section 3.3. Finally, an adaptive inference method, that allows for early stopping the inference process when the network is confident enough, is proposed in Subsection 3.4.

#### 3.1. Background and Notation

150 Let  $f_{\mathbf{W}}(\mathbf{x}, i)$  denote the output of the  $i$ -th layer of a neural network composed of  $m$  layers, where the notation  $\mathbf{W}$  is used to denote the parameters of the network and  $\mathbf{x}$  denotes the input to the neural network. This work focuses on convolutional neural networks, where the input to the neural network is an image, i.e.,  $\mathbf{x} \in \mathbb{R}^{W \times H \times C}$ , where  $W$ ,  $H$  and  $C$  are the width, height and number of channels of the image respectively. Note that this is without loss of generality, since the proposed method can be also readily applied for networks operating on other signals, such as audio [25], time-series [26], etc. Also, note that the output of the  $i$ -th convolutional layer is a feature map that is also denoted by  $\mathbf{y}^{(i)} = f_{\mathbf{W}}(\mathbf{x}, i) \in \mathbb{R}^{W_i \times H_i \times C_i}$ , where  $W_i$  and  $H_i$  are the width and height of the feature map extracted from the  $i$ -th layer of the network, while  $C_i$  denotes the number of filters used in the corresponding convolutional layer. The final output of the neural network, denoted by  $\mathbf{y} = f_{\mathbf{W}}(\mathbf{x}, m) \in \mathbb{R}^{N_C}$ , is a vector containing the probabilities that the input image  $\mathbf{x}$  belongs to each of the  $N_C$  classes. Note that even though the aforementioned setup considers only classification problems, it is straightforward to extend it to handle other tasks as well, such as regression tasks.

Given a training set of  $N$  images  $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ , along with a target vector  $\mathbf{t}_i \in \mathbb{R}^{N_C}$ , which encodes the classification label for each image, then the network is trained using back-propagation to minimize a loss function  $\mathcal{L}$ :

$$\mathbf{W}' = \mathbf{W} - \eta \sum_{j=1}^N \frac{\partial \mathcal{L}(f_{\mathbf{W}}(\mathbf{x}_j, m), \mathbf{t}_j)}{\partial \mathbf{W}}, \quad (1)$$

where the notation  $\mathbf{W}'$  is used to denote the updated parameters of the neural network and  $\eta$  is the used learning rate. Usually the optimization is performed in batches, the classification target  $\mathbf{t}_i$  is a one-hot encoding of the class of each sample, while the cross-entropy loss is employed as the loss

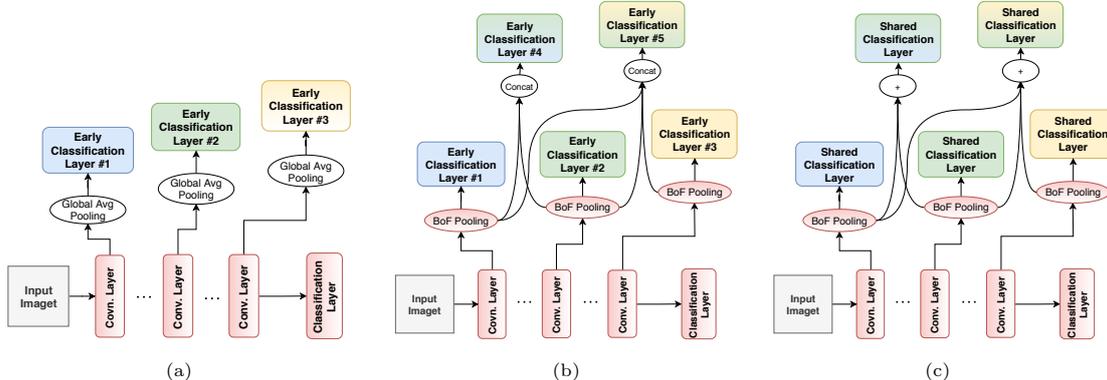


Figure 1: Comparing three different ways to use early exits. Fig. 1a demonstrates one typical way to use early exits to estimate the final output of the network at various points of its computational graph. Early exits can be improved by replacing global average pooling with the Bag-of-Features model to compile hierarchical early exits, as shown in Fig. 1b. Early exits can be further improved by sharing the same classification layer among different early exits and using additive histogram representations, as shown in Fig. 1c. This allows for reducing the number of parameters and forming implicit common representations spaces.

function:

$$\mathcal{L}(\mathbf{y}, \mathbf{t}) = - \sum_{i=1}^{N_C} [\mathbf{t}]_i \log([\mathbf{y}]_i), \quad (2)$$

where the notation  $[\mathbf{y}]_i$  is used to refer to the  $i$ -th element of vector  $\mathbf{y}$ .

### 3.2. Efficient Early Exits using Bag-of-Features Aggregation

Employing early exits provides a way to estimate the final output of the network at various points of its computational graph, without having to feed-forward the whole network [15]. That is, an additional estimator:

$$g_{\mathbf{W}_i}^{(i)}(\mathbf{y}^{(i)}) = g_{\mathbf{W}_i}^{(i)}(f_{\mathbf{W}_i}(\mathbf{x}, i)) \in \mathbb{R}^{N_C} \quad (3)$$

is employed at the  $i$ -th layer to estimate the final output of the network without feed-forwarding the network until the last layer. This process is illustrated in Fig. 1a. The notation  $\mathbf{W}_i$  is used to refer to the parameters of the  $i$ -th early exit. Each early exit usually employs a feature aggregation method, e.g., global average pooling, to extract a compact representation from each feature map, and a classification layer that estimates the final output of the network. Using an efficient feature aggregation approach is of crucial importance, since directly using the raw feature maps would lead to enormous classification layers (due to the large size of the feature maps, especially for the early layers of the network).

Existing methods usually apply global average pooling to extract a compact representation  $\mathbf{s}^{(i,avg)}$

out of the intermediate feature maps of the  $i$ -th layer of a network:

$$\mathbf{s}^{(i,avg)} = \frac{1}{W_i H_i} \sum_{k=1}^{W_i} \sum_{l=1}^{H_i} [\mathbf{y}^{(i)}]_{kl} \in \mathbb{R}^{C_i}, \quad (4)$$

where the notation  $[\mathbf{y}^{(i)}]_{kl}$  is used to refer to the feature vector extracted from the location  $(k, l)$  of the feature map of the  $i$ -th layer. The extracted averaged representation is then fed to a fully connected layer that is trained to estimate the final classification output of the network, as shown in Fig. 1a.

However, using plain global average pooling possibly discards a great amount of valuable information, as also discussed in Section 1 and experimentally demonstrated in Section 4. Therefore, to overcome this limitation, in this work a trainable BoF-based aggregation scheme is used to compile a compact representation that can be then fed to the used fully connected classification layer. First, a set of prototype vectors (also known as codewords)  $\mathbf{v}_{ij} \in \mathbb{R}^{C_i}$  are used to model the distribution of the feature vectors extracted from the  $i$ -th layer. The set of these vectors  $\mathcal{V}_i = \{\mathbf{v}_{i1}, \mathbf{v}_{i2}, \dots, \mathbf{v}_{iN_K}\}$ , where  $N_K$  is the number of codewords used, is called dictionary or codebook. Note that a different codebook  $\mathcal{V}_i$  must be used for each exit layer, since the distribution of the feature vectors extracted from each layer is different. Then, we can estimate the probability of observing each feature vector  $[\mathbf{y}^{(i)}]_{kl}$ , extracted from the  $i$ -layer of the network, for a given image  $\mathbf{x}$  using Kernel Density Estimation [27] as:

$$p([\mathbf{y}^{(i)}]_{kl} | \mathbf{x}) = \sum_{j=1}^{N_K} [\mathbf{s}^{(i)}]_j K([\mathbf{y}^{(i)}]_{kl}, \mathbf{v}_{ij}) \in [0, 1], \quad (5)$$

where  $K(\cdot)$  is a kernel function and  $\mathbf{s}^{(i)} \in \mathbb{R}^{N_K}$  are the parameters that control the density estimation. Employing a maximum likelihood estimator allows for estimating these parameters as:

$$\mathbf{s}^{(i)} = \arg \max_{\mathbf{s}} \sum_{k=1}^{W_i} \sum_{l=1}^{H_i} \log \left( \sum_{j=1}^{N_K} [\mathbf{s}]_j K([\mathbf{y}^{(i)}]_{kl}, \mathbf{v}_{ij}) \right) \in \mathbb{R}^{N_K}. \quad (6)$$

Indeed, as demonstrated in [27], these image specific parameters can be trivially calculated, giving rise to the well known soft-BoF formations [22, 28]. Therefore, the representation extracted from the  $i$ -th layer of the network is calculated as:

$$\mathbf{s}^{(i)} = \frac{1}{W_i H_i} \sum_{k=1}^{W_i} \sum_{l=1}^{H_i} \mathbf{u}_{ikl} \in \mathbb{R}^{N_K}, \quad (7)$$

where  $[\mathbf{u}_{ikl}]_j = \frac{K([\mathbf{y}^{(i)}]_{kl}, \mathbf{v}_{ij})}{\sum_{m=1}^{N_K} K([\mathbf{y}^{(i)}]_{kl}, \mathbf{v}_{im})} \in [0, 1]$ . The histogram vector  $\mathbf{s}^{(i)}$  essentially provides a compact summary that describes the semantic content of an image at various levels of granularity, maintaining more information regarding the actual *distribution* of the vectors  $[\mathbf{y}^{(i)}]_{kl}$  than the average representation ( $\mathbf{s}^{(i,avg)}$ ).

To implement the BoF model a normalized RBF layer, followed by a recurrent accumulation layer, is employed, as proposed in [19]. Furthermore, to simplify the implementation we also use a hyperbolic

(sigmoid) kernel in order to calculate the similarity between each the codeword and each extracted feature vectors [29]:

$$K([\mathbf{y}^{(i)}]_{kl}, \mathbf{v}) = \frac{1}{2} \left( \tanh(c_1 [\mathbf{y}^{(i)}]_{kl}^T \mathbf{v} + c_2) + 1 \right) \quad (8)$$

180 where  $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ , and  $c_1$  and  $c_2$  are the kernel parameters (typically set to  $c_1 = 1$  and  $c_2 = 0$ ). Note that the kernel is appropriately scaled to  $0 \dots 1$ , ensuring its compatibility with the quantization process.

The proposed BoF-based aggregation scheme is shown in Fig. 1b, where first we employ BoF-based aggregation to compile a compact representation  $\mathbf{s}^{(i)}$  for each image out of each layer, which 185 is subsequently fed to the final fully connected early classification layer (Early classification layers #1, #2 and #3). Furthermore, note that the quantization process can be applied at various spatial levels, as in various spatial pyramid aggregation schemes [30], giving rise to the Spatial BoF [19]. This process allows for retaining more spatial information, which can provide a significant benefit for the exits placed on earlier layers, where the effective receptive field of the convolutional layers is smaller.

Each early exit is trained to estimate the same targets using a representation extracted from the output of the  $i$ -th layer of the network  $\mathbf{y}^{(i)}$ , i.e.,

$$\mathbf{W}'_i = \mathbf{W}_i - \eta \sum_{j=1}^N \frac{\partial \mathcal{L} \left( g_{\mathbf{W}_i}^{(i)}(\mathbf{y}_j), \mathbf{t}_j \right)}{\partial \mathbf{W}}. \quad (9)$$

Note that early exits can be either trained after training the base network or by simultaneously training both the base network and the early exits. Even though the latter option can allow the network to adapt to the added early exits (and, as a result, lead to better accuracy for the early exits), it can potentially harm the classification performance for the main network, e.g., if too many early exits are used. To avoid this issue, in this work we first train the base network, we fix its parameters  $\mathbf{W}$ , and then train all the exit layers simultaneously. It is also worth noting that the network can be directly trained to predict a soft target vector, as produced by the neural network, instead of the binary target  $\mathbf{t}_j$ , following the neural network distillation principle [31]:

$$\mathbf{W}'_i = \mathbf{W}_i - \eta \sum_{j=1}^N \frac{\partial \mathcal{L} \left( g_{\mathbf{W}_i}^{(i)}(\mathbf{y}_j), f_{\mathbf{W}}(\mathbf{x}_j) \right)}{\partial \mathbf{W}}. \quad (10)$$

190 In this way, the network can be trained to directly estimate the actual output of the network, instead of being trained to predict the same targets as the output of the classification layer of the network.

### 3.3. Hierarchical Bag-of-Features Aggregation

The BoF-based aggregation method proposed in the previous Subsection is capable of extracting rich and discriminative representations from the feature map of each layer. However, all the information that exists in the representations extracted from the earlier exit layers is discarded at each exit

layer. Therefore, instead of merely relying on the representation extracted from the current layer, in this paper we propose compiling an incremental hierarchical representation that exploits the representations extracted from the previous exit layers, effectively overcoming the aforementioned limitation. To this end, the final representation  $\mathbf{s}^{(i,h)}$  extracted from the  $i$ -th layer is calculated as:

$$\mathbf{s}^{(i,h)} = \begin{cases} \mathbf{s}^{(i)} & \text{if } i = 1 \\ \mathbf{s}^{(i)} \frown \mathbf{s}^{(i-1,h)} & \text{if } i > 1 \end{cases}, \quad (11)$$

where the notation  $\mathbf{a} \frown \mathbf{b}$  is used to denote the concatenation of vectors  $\mathbf{a}$  and  $\mathbf{b}$ . This process is depicted in Fig. 1b (Classification Layer #4 and #5). It is worth noting that the representation extracted from the previous early exits can be cached and directly reused for the following early exits with no additional cost. As a result, this approach provides an easy and straightforward way to further refine the predictions of the early exits, increasing the classification accuracy of the network, as also demonstrated in Section 4.

However, this hierarchical approach can lead to increasingly larger exit layers as more exits are used, since the length of the representation  $\mathbf{s}^{(i,h)}$  gradually increases by  $N_K$  as the number of exit layers increases. To maintain the advantage of using hierarchical representations, without the added complexity of using increasingly larger exit layers we propose a simple, yet efficient, aggregation approach. Instead of simply concatenating the representations together, we propose forming an additive common histogram representation space for all the exit layers. That is, the updated histograms are calculated as:

$$\mathbf{s}^{(i,ch)} = \begin{cases} \mathbf{s}^{(i)} & \text{if } i = 1 \\ \mathbf{s}^{(i)} + \alpha \mathbf{s}^{(i-1,ch)} & \text{if } i > 1 \end{cases}, \quad (12)$$

where  $\alpha$  is a decaying factor for the previous histogram (typically set to  $\alpha = 1$  when only a few exit layers are used). This approach allows for gradually refining the histogram estimation, while keeping the potentially useful information extracted from the previous layers. Finally, this additive method allows for using a single classification layer that can be reused for all the exit layers, instead of learning separate classification layers for each early exit, which further promotes forming common representation spaces and reduces the number of parameters required for the early exits. This approach indeed leads to a significant reduction of the number of parameters required (since only one classification layer is required regardless of the number of exit layers used), with minimal impact on the classification accuracy, as also experimentally demonstrated in Section 4. The proposed classification layer sharing approach is shown in Fig. 1c. Note that each early exit is still equipped with a separate codebook used for appropriately constructing the corresponding histogram  $\mathbf{s}^{(i)}$ . Also, the hyper-parameter  $\alpha$  can be set to lower values to prevent significant distribution shifts that could negatively affect the shared

classification layers. Nonetheless, this behavior was not observed during the conducted experiments. Therefore,  $\alpha = 1$  was used for all the conducted experiments.

### 3.4. Adaptive Inference with Early Exits

There are several ways to feed-forward a network that is equipped with early exits. Perhaps the most straightforward approach is to use a specific early exit according to the available computational resources. Therefore, an earlier early exit is used when there are limited computational resources available, while latter early exits are used when more computational resources are available. For example, we can consider a real-time embedded system designed to perform face recognition from CCTV footage using a deep neural network. When more people appear in a frame then the time required to perform face recognition increases, since each face must be fed to the network. However, using early exits allows for dynamically reducing the number of calculations required for the recognition by appropriately selecting an early exit, allowing for meeting the real-time constraints by slightly reducing the quality of service without over-engineering the system, e.g., using an unnecessary powerful processor to handle every possible number of faces in real time.

The proposed method can be also used to reduce the load, and as a result the energy consumption, of a system by dynamically feed-forwarding through the network according to the difficulty of each training sample and the prediction uncertainty of each exit layer for the specific sample. To this end, we calculate the average class activations at each exit layer  $i$  as:

$$\mu_i = \frac{1}{N} \frac{1}{N_C} \sum_{k=1}^{N_C} \sum_{j=1}^N [g_{\mathbf{W}_i}^{(i)}(\mathbf{y}_j)]_k, \quad (13)$$

where  $g_{\mathbf{W}_i}^{(i)}$  is the classification output of the corresponding early exit. The average activations can be calculated either using the training set or, in order to acquire a more robust estimation, using a validation set. Then, the user specifies a confidence hyper-parameter  $\beta$  that controls when the inference process can stop early at the  $i$ -th exit layer, without having to feed-forward through the rest of the network. Therefore, the inference process can stop at the  $i$ -th layer when the maximum activation exceeds by  $\beta$  the average activation :

$$[g_{\mathbf{W}_i}^{(i)}(\mathbf{y}_j)]_k > \beta \mu_i, \quad (14)$$

where  $k = \arg \max g_{\mathbf{W}_i}^{(i)}(\mathbf{y}_j)$ . Note that setting different values for the hyper-parameter  $\beta$  allows for controlling the trade off between the classification accuracy of the network and the number of resources needed for classifying one input image. Larger values for the hyper-parameter  $\beta$  lead to more accurate classification decisions, while smaller values leads to faster (since the inference process can stop early), but less accurate classification decisions. It is worth noting that, as it is demonstrated in Section 4,

230 the inference process can be stopped early for many input samples with a minimal impact on the classification accuracy.

## 4. Experimental Evaluation

The proposed method is extensively evaluated in this Section. First, the used datasets, evaluation setup, and network architectures are described in Subsection 4.1, while the evaluation results are  
235 provided and discussed in Subsection 4.2.

### 4.1. Datasets and Evaluation Setup

Five datasets were used for evaluating the proposed method: a) the MNIST image classification dataset [32], b) the Fashion MNIST fashion product classification dataset [33], as well as the more challenging c) CIFAR-10 object recognition dataset [34], d) FER-2013 facial expression dataset [35],  
240 and e) PlantVillage dataset [36], which contains leaf images from healthy and infected plants.

Three different neural network architectures were also employed for the evaluation. The first one, called “CNN-1”, was used for the experiments conducted with the MNIST dataset. CNN-1 is composed of a  $3 \times 3$  convolution layer with 32 filters, followed by  $2 \times 2$  max pooling layer, a  $3 \times 3$  convolution layer with 64 filters, another  $2 \times 2$  max pooling layer, a fully connected layer with 1024  
245 neurons, dropout with rate  $p = 0.5$ , and a final fully connected classification layer. For the Fashion MNIST dataset we used a more powerful network that uses twice the number of filters in the first two convolutional layers. Therefore, this network, called “CNN-2”, is composed of a  $3 \times 3$  convolution layer with 64 filters, followed by a  $2 \times 2$  max pooling layer, a  $3 \times 3$  convolution layer with 128 filters, another  $2 \times 2$  max pooling layer, a fully connected layer with 1024 neurons layer, dropout with rate  
250  $p = 0.5$ , and a final fully connected classification layer. The ReLU activation function was used for all the layers. For the CIFAR-10 and FER-2013 datasets a more powerful network, the MobileNet v.2 [37], was employed, after appropriately tuning the filter size of the first convolutional layer for each dataset. Images from the PlantVillage dataset were resized to  $32 \times 32$  pixels, while random rotations (up to 30 degrees) and horizontal flips were used to augment the training dataset. We used classes  
255 that are related to the three following plants: “Pepper bell” (2 classes), “Potato” (3 classes), and “Tomato” (10 classes). Finally, the CNN-2 architecture was used for all the conducted experiments using this dataset. The categorical cross-entropy loss was used for training all the networks, combined with the Adam optimizer. The CNN-1 and CNN-2 models were trained for 50 epochs with learning rate 0.001, while the MobileNets and the networks used for the PlantVillage dataset were trained for  
260 50 epochs with a learning rate of 0.001, followed by additional 50 training epochs with a learning rate of 0.0001.

Table 1: Comparing the classification error and computational overhead for different variants of the proposed method. The classification error (%), computational complexity (MMAC) and required number of additional parameters is reported. The notation “AH-BoF (F, X)” refers to using the proposed method with  $X$  codewords, while “AH-BoF (S, X)” refers to employing additive histogram spaces and re-using the employed classification layer. When the “AH-SBoF” variant is used, then spatial segmentation into four regions is enabled.

Method	Early Exit - 1		Early Exit - 2		Hierarchical Exit		# Added Parameters.
	Error	MMAC	Error	MMAC	Error	MMAC	
<b>MNIST Dataset (CNN-1)</b>							
AH-BoF (F, 16)	15.19%	0.31 (8%)	5.37%	2.48 (61%)	3.79%	2.57 (63%)	2.25k
AH-BoF (S, 16)	16.79%	0.31 (8%)	6.05%	2.48 (61%)	4.38%	2.57 (63%)	1.75k
AH-SBoF (F, 8)	5.00%	0.26 (6%)	3.03%	2.47 (60%)	1.86%	2.51 (61%)	2.10k
AH-SBoF (S, 8)	6.11%	0.26 (6%)	3.92%	2.47 (60%)	2.57%	2.51 (61%)	1.12k
AH-SBoF (F, 32)	<b>2.85%</b>	0.40 (10%)	<b>2.63%</b>	2.51 (61%)	1.81%	2.69 (66%)	8.30k
AH-SBoF (S, 32)	3.48%	0.40 (10%)	2.67%	2.51 (61%)	<u>1.57%</u>	2.69 (66%)	4.43k
<b>Fashion MNIST Dataset (CNN-2)</b>							
AH-BoF (F, 16)	17.05%	0.61 (5%)	15.26%	9.42 (74%)	12.98%	9.60 (76%)	3.78k
AH-BoF (S, 16)	18.70%	0.61 (5%)	16.21%	9.42 (74%)	13.95%	9.60 (76%)	3.28k
AH-SBoF (F, 8)	15.55%	0.52 (4%)	14.51%	9.40 (74%)	11.54%	9.48 (75%)	2.87k
AH-SBoF (S, 8)	17.61%	0.52 (4%)	15.94%	9.40 (74%)	14.18%	9.48 (75%)	1.89k
AH-SBoF (F, 16)	<b>13.34%</b>	0.61 (5%)	<b>12.93%</b>	9.42 (74%)	<u>10.54%</u>	9.60 (76%)	5.70k
AH-SBoF (S, 16)	14.22%	0.61 (5%)	13.52%	9.42 (74%)	11.30%	9.60 (76%)	3.76k
<b>CIFAR-10 Dataset (MobileNet v.2)</b>							
AH-BoF (F, 16)	17.68%	44.41 (47%)	11.05%	64.25 (68%)	10.81%	64.32 (68%)	3.27k
AH-BoF (S, 16)	18.36%	44.41 (47%)	11.57%	64.25 (68%)	11.19%	64.32 (68%)	2.77k
AH-BoF (F, 32)	14.55%	44.48 (47%)	10.05%	64.35 (68%)	9.90%	64.48 (68%)	6.50k
AH-BoF (S, 32)	14.92%	44.48 (47%)	10.61%	64.35 (68%)	9.78%	64.48 (68%)	5.52k
AH-BoF (F, 128)	<b>11.76%</b>	44.88 (47%)	<b>9.14%</b>	64.95 (69%)	<b>8.82%</b>	65.48 (69%)	25.89k
AH-BoF (S, 128)	11.90%	44.88 (47%)	9.47%	64.95 (69%)	8.99%	65.48 (69%)	22.03k
<b>FER-2013 Dataset (MobileNet v.2)</b>							
AH-BoF (F, 16)	50.35%	98.60 (47%)	44.33%	143.24 (68%)	43.61%	143.39 (68%)	3.07k
AH-BoF (S, 16)	52.55%	98.60 (47%)	45.17%	143.24 (68%)	44.39%	143.39 (68%)	2.72k
AH-BoF (F, 32)	48.70%	98.75 (47%)	<b>43.66%</b>	143.46 (68%)	42.57%	143.76 (68%)	6.11k
AH-BoF (S, 32)	51.62%	98.75 (47%)	45.17%	143.46 (68%)	43.13%	143.76 (68%)	5.42k
AH-BoF (F, 128)	<b>48.39%</b>	99.64 (47%)	43.83%	144.80 (69%)	42.69%	146.00 (69%)	24.35k
AH-BoF (S, 128)	48.51%	99.64 (47%)	44.97%	144.80 (69%)	<u>42.13%</u>	146.00 (69%)	21.65k

The percentage of MMAC with the respect to the total MMAC required to feed-forward the network are reported in parenthesis. The best results for each early exit are reported in bold, while the best overall results for each dataset are underlined.

Table 2: Comparing the proposed “AH-(S)BoF” method to two competitive early-exit approaches (ElasticNet and BranchyNet). The same notation as in Table 1 is used for the proposed method.

Method	Dataset	Early Exit - 1		Early/Hierarchical Exit - 2		# Added Parameters.
		Error	MMAC	Error	MMAC	
ElasticNet	MNIST	49.99%	0.22 (5%)	7.07%	2.45 (60%)	1.95k
BranchyNet	MNIST	18.48%	0.36 (9%)	3.01%	2.48 (60%)	3.65k
AH-SBoF (S, 8)	MNIST	<b>6.11%</b>	0.26 (6%)	<b>2.57%</b>	2.51 (61%)	1.12k
ElasticNet	FashionMNIST	32.51%	0.43 (3%)	16.91%	9.37 (74%)	3.87k
BranchyNet	FashionMNIST	25.02%	0.71 (6%)	12.51%	9.41 (74%)	7.11k
AH-SBoF (S, 16)	FashionMNIST	<b>14.22%</b>	0.61 (5%)	<b>11.30%</b>	9.60 (76%)	3.76k
ElasticNet	CIFAR-10 Dataset	27.04%	44.34 (47%)	15.74%	64.15 (68%)	3.23k
BranchyNet	CIFAR-10 Dataset	16.65%	44.51 (47%)	11.82%	64.40 (68%)	11.89k
AH-BoF (S, 32)	CIFAR-10 Dataset	<b>14.92%</b>	44.48 (47%)	<b>9.78%</b>	64.48 (68%)	5.52k
ElasticNet	FER-2013 Dataset	59.88%	98.45 (47%)	51.40%	143.01 (68%)	2.26k
BranchyNet	FER-2013 Dataset	49.21%	98.68 (47%)	44.41%	143.36 (68%)	5.96k
AH-BoF (F, 32)	FER-2013 Dataset	<b>48.70%</b>	98.75 (47%)	<b>42.57%</b>	143.76 (68%)	6.11k
ElasticNet	PlantVillage	29.72%	4.04 (23%)	5.54%	14.09 (81%)	4.85k
BranchyNet	PlantVillage	21.48%	4.38 (25%)	6.59%	14.32 (82%)	12.06k
AH-BoF (S, 32)	PlantVillage	<b>12.37%</b>	4.25 (24%)	<b>5.26%</b>	14.40 (83%)	5.69k

The percentage of MMAC with the respect to the total MMAC required to feed-forward the network are reported in parenthesis.

For all the conducted experiments two early exits were employed, together with one hierarchical exit that combines the information extracted from these two exits. For both the CNN-1 and CNN-2 models, the first early exit layer was placed after the 1st convolutional layer, while the second early exit layer was placed after the 2nd convolutional layer. The early exits were placed at the 5th and 7th convolutional layers, respectively, for the MobileNet model. The proposed method is called Adaptive Hierarchical BoF, abbreviated as “AH-BoF” for the rest of the paper. When a spatial segmentation scheme into 4 regions is used, the proposed method is abbreviated as “AH-SBoF”. Note that when the Spatial BoF method is used, then the size of the extracted representation is increased by a factor of 4, since four different histograms are extracted (one for each spatial region). Two different variants of the proposed method are evaluated: a) concatenating the hierarchical representations (as described in (11)) and using separate classification layers for each early exit (denoted by “AH-(S)BoF (F, X)”, where X refers to the number of used codewords), and b) using additive histogram spaces (as described in (12)) and classification layer reuse (denoted by “AH-(S)BoF (S, X)”). The proposed method is also compared to using global average pooling aggregation, as proposed in [16] (denoted by “ElasticNet”), as well as to using an additional convolutional layer before performing the aggregation proposed for early exits (denoted by “BranchyNet”) [13]. Note that all the experiments were conducted using the PyTorch framework [38].

Table 3: Evaluating the proposed adaptive inference classification strategy for three different settings. The classification error and the average complexity for classifying each input sample (“Avg. MMAC”) are reported.

Method	Adaptive Inference (High Speed)		Adaptive Inference (Balanced)		Adaptive Inference (High Precision)	
	Error	Avg. MMAC	Error	Avg. MMAC	Error	Avg. MMAC
<b>MNIST Dataset (CNN-1)</b>						
Static Inference					0.68%	4.10
AH-BoF (F, 16)	4.16%	1.08	1.86%	2.07	0.73%	3.60
AH-BoF (S, 16)	4.45%	1.14	1.79%	2.23	0.72%	3.79
AH-SBoF (F, 8)	1.32%	0.69	0.94%	0.98	0.76%	1.69
AH-SBoF (S, 8)	1.61%	0.78	0.97%	1.19	0.70%	2.29
AH-SBoF (F, 32)	1.09%	0.67	0.74%	0.97	<b>0.68%</b>	<b>1.90</b>
AH-SBoF (S, 32)	1.06%	0.74	0.78%	1.08	0.68%	2.43
<b>Fashion MNIST Dataset (CNN-2)</b>						
Static Inference					7.82%	12.66
AH-BoF (F, 16)	10.10%	4.04	8.55%	5.86	7.85%	9.55
AH-BoF (S, 16)	10.18%	4.59	8.45%	6.84	7.84%	10.42
AH-SBoF (F, 8)	9.54%	3.62	8.38%	5.14	<b>7.81%</b>	<b>9.11</b>
AH-SBoF (S, 8)	10.45%	4.30	8.46%	6.34	7.84%	10.17
AH-SBoF (F, 16)	8.95%	3.52	8.17%	4.76	7.82%	9.92
AH-SBoF (S, 16)	9.13%	3.91	8.18%	5.50	7.82%	10.53
<b>CIFAR-10 (MobileNet v.2)</b>						
Static Inference					7.81%	94.61
AH-BoF (F, 16)	10.23%	53.20	8.81%	63.14	<b>7.76%</b>	<b>77.03</b>
AH-BoF (S, 16)	10.38%	53.32	8.72%	65.42	7.77%	80.75
AH-BoF (F, 32)	9.05%	52.73	8.25%	60.90	7.78%	70.77
AH-BoF (S, 32)	9.13%	52.84	8.37%	62.80	7.84%	74.84
AH-BoF (F, 128)	8.12%	52.09	8.03%	58.26	7.82%	64.33
AH-BoF (S, 128)	8.16%	52.36	7.98%	59.91	7.77%	67.21
<b>FER-2013 (MobileNet v.2)</b>						
Static Inference					38.84%	211.52
AH-BoF (F, 16)	43.58%	131.27	38.92%	184.97	39.03%	209.11
AH-BoF (S, 16)	43.86%	130.37	39.34%	193.07	39.20%	211.01
AH-BoF (F, 32)	41.73%	131.75	<b>38.59%</b>	<b>182.56</b>	38.62%	207.08
AH-BoF (S, 32)	44.36%	129.07	39.67%	189.62	39.73%	211.31
AH-BoF (F, 128)	41.32%	133.32	39.06%	176.74	39.26%	205.13
AH-BoF (S, 128)	41.74%	135.00	38.62%	182.47	38.70%	208.46
<b>PlantVillage (CNN-2)</b>						
Static Inference					1.24%	17.38
AH-BoF (F, 32)	3.28%	7.06	1.84%	8.96	1.27%	14.90
AH-BoF (S, 32)	3.60%	7.55	1.54%	10.40	1.24%	16.42
AH-BoF (F, 64)	2.52%	6.92	<b>1.55%</b>	<b>8.43</b>	1.28%	14.14
AH-BoF (S, 64)	2.75%	7.27	1.61%	9.41	1.25%	15.72

#### 4.2. Evaluation Results

280 First, we provide a sensitivity analysis, where we evaluate the effect of different design choices on the performance of the proposed method. The evaluation results are reported in Table 1. The total multiply-accumulate operations (Million MAC, MMAC) are also reported for each network up to the corresponding exit, while the column “# Added Parameters” refers to the number of added parameters to the network due to the early exits. The number in parenthesis for the MMAC refers to  
285 the percentage of MMAC for the current early exit with respect to the total number of MMAC required to feed-forward the whole network. Note that even with spending as little as 5% of the total MMAC (Fashion MNIST combined with “AH-SBoF(F, 16)”) leads to adequate results (classification accuracy  $< 14\%$ ). As it will be also demonstrated later, this allows for early stopping the inference process at the earlier exits, allowing for acquiring significant performance benefits. The two different variants of  
290 the proposed method, namely the AH-BoF with concatenated histogram spaces (“F” variant) and the AH-BoF with additive histogram spaces and classification layer reuse (“S” variant), are also compared in Table 1. The lightweight “S” variant is capable of significantly reducing the number of parameters in all the cases, with only a minimal impact on the classification accuracy for the first two early exits. On the other hand, the lightweight “S” variant is actually outperforming the other methods for the  
295 hierarchical exit that combines the information extracted from the other two early exits for two of the evaluated datasets (MNIST and FER-2013 datasets).

There are also several interesting conclusions that can be drawn regarding the parameters of the BoF model. First, note that by appropriately tuning the number of used codewords we can effectively control the trade-off between the added parameters and the classification accuracy of the network.  
300 Also, note that the added early exits have a minimal effect on the computational complexity, since regardless the complexity of the used BoF model, the number of MMAC remains almost the same (especially for the more complex MobileNet network). Furthermore, in almost any case, increasing the number of codewords seems to have a positive effect on the classification accuracy. At the same time, using spatial segmentation (“SBoF”) seems to be especially important when the early exits are  
305 used on convolutional layers with smaller receptive fields, e.g., CNN-1 and CNN-2.

The proposed method is also compared to the ElasticNet and BrancyNet approaches in Table 3. We selected the most competitive variant of the proposed method that is closer to the number of parameters used by the rest of the evaluated methods. Furthermore, we also tuned the number of filters in the BranchyNet model in order to avoid significantly exceeding the number of parameters  
310 used by the rest of the methods. Note that this is not always straightforward, since  $3 \times 3$  convolutions are used in BranchyNet instead of  $1 \times 1$ , as the proposed method does, leading to 9-fold increase in the required number of parameters. The proposed method always leads to improved accuracy over

the two evaluated methods, while requiring about the same number of parameters (or significantly less compared to some BranchyNet variants). Note that this is also true for many other variants of the proposed method, as reported in Table 1.

Next, we evaluated the effect of using the proposed adaptive inference strategy for early stopping the inference process when the network is confident enough. The evaluation results are reported in Table 3. Three different inference strategies were used, namely “High Speed” ( $\beta = 1$ ), “Balanced” ( $\beta = \frac{1+c\frac{1}{\mu_0}}{1+c}$ ) and “High Accuracy” ( $\beta = \frac{c}{\mu_0}$ ), where  $\mu_0$  (calculated as in (13)) is used to estimate the uncertainty of the neural network at the first early exit and  $c$  is a hyper-parameter that controls the uncertainty limit for stopping at an early exit ( $c = 0.99$  were used for all the conducted experiments). The “Static Inference” baseline refers to using the final classification layer of the network, without adding any early exit. Several interesting conclusions can be drawn from the results reported in Table 3. First, adding early exits and using the proposed method allows for significantly reducing the average inference time without harming the classification accuracy. For example, for the MNIST dataset the proposed method reduced the average MMAC by 4 times, while achieving the same classification performance. For the Fashion MNIST the error is slightly reduced to 7.81%, while the number of MMAC are reduced from 12.66 to 9.11. For the CIFAR-10 and FER-2013 datasets, using the proposed approach actually achieves lower classification error than the original network, hinting that earlier layers may contain useful information discarded by many neural network architectures, as also suggested by recent neural network architectures [39]. Furthermore, note that for the FER-2013 dataset the best results are obtained using an inference strategy that tends to stop at an earlier layer. The proposed adaptive inference method was capable of reducing the MMAC while achieving competitive classification performance for almost every evaluated case, demonstrating the practical usefulness of the proposed method for adaptively altering the computational graph of a DL model according to the difficulty of the input samples.

## 5. Conclusion

Deep learning models equipped with early exits are capable of providing adaptive computational graphs that allow for directly adapting a model to the currently available computational resources. However, existing methods for implementing early exits mainly employ naive aggregation methods, such as global average pooling, significantly restricting their performance. At the sample time, they usually ignore all the information that is extracted by the earlier exits, despite the fact that this information is often already extracted and available at no additional cost. In this paper, we proposed a Bag-of-Features (BoF)-based method that is capable of overcoming these limitations. To this end, the proposed method constructs efficient hierarchical early exit layers with minimal computational

overhead, since it employs additive shared histogram spaces, that gradually refine the information extracted from the various layers of a network, in a hierarchical manner. At the same time, it employs a classification layer reuse strategy that allows for further reducing the number of parameters needed per exit layer. The proposed method can be further combined with an adaptive inference strategy that allows for early stopping the inference process when the network is confident enough for its output, leading to further performance benefits. It is worth noting that the proposed method is generic and can be readily combined with any neural network architecture, leading to a practical tool that can be effectively used in various real-world embedded applications, as demonstrated through the conducted experiments on five different datasets. At the same time, the significant improvements obtained using the proposed approach pave the way for developing more sophisticated and advanced methods for designing and implementing early exits. Among them, adaptive methods for selecting the most appropriate early exit for each input sample can be developed, instead of relying on a fixed and pre-defined threshold, allowing for potentially further increasing the efficiency and accuracy of the proposed method.

## Acknowledgments

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR). This publication reflects the authors’ views only. The European Commission is not responsible for any use that may be made of the information it contains.

**Declarations of interest:** None

## References

- [1] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, *Nature* 521 (7553) (2015) 436.
- [2] J. Mehta, A. Majumdar, Rodeo: robust de-aliasing autoencoder for real-time medical image reconstruction, *Pattern Recognition* 63 (2017) 499–510.
- [3] B. Gecer, S. Aksoy, E. Mercan, L. G. Shapiro, D. L. Weaver, J. G. Elmore, Detection and classification of cancer in whole slide breast histopathology images using deep convolutional networks, *Pattern Recognition* 84 (2018) 345–356.
- [4] P. Li, Z. Chen, L. T. Yang, Q. Zhang, M. J. Deen, Deep convolutional computation model for feature learning on big data in internet of things, *IEEE Transactions on Industrial Informatics* 14 (2) (2018) 790–798.

- [5] M. Patacchiola, A. Cangelosi, Head pose estimation in the wild using convolutional neural networks and adaptive gradient methods, *Pattern Recognition* 71 (2017) 132–143.
- [6] Z. Jiao, X. Gao, Y. Wang, J. Li, H. Xu, Deep convolutional neural networks for mental load classification based on eeg data, *Pattern Recognition* 76 (2018) 582–595.
- [7] S. Han, H. Mao, W. J. Dally, Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, in: *Proceedings of the International Conference on Learning Representations*, 2016.
- [8] J.-H. Luo, J. Wu, W. Lin, Thinet: A filter level pruning method for deep neural network compression, in: *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5058–5066.
- [9] N. Passalis, A. Tefas, Learning deep representations with probabilistic knowledge transfer, in: *Proceedings of the European Conference on Computer Vision*, 2018, pp. 268–284.
- [10] W. Hao, Z. Zhang, Spatiotemporal distilled dense-connectivity network for video action recognition, *Pattern Recognition* 92 (2019) 13–24.
- [11] T.-B. Xu, P. Yang, X.-Y. Zhang, C.-L. Liu, Lightweightnet: Toward fast and lightweight convolutional neural networks via architecture distillation, *Pattern Recognition* 88 (2019) 272–284.
- [12] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, K. Q. Weinberger, Multi-scale dense networks for resource efficient image classification, in: *Proceedings of the International Conference on Learning Representations*, 2018.
- [13] S. Teerapittayanon, B. McDanel, H.-T. Kung, Branchynet: Fast inference via early exiting from deep neural networks, in: *Proceedings of the International Conference on Pattern Recognition*, 2016, pp. 2464–2469.
- [14] A. Veit, S. Belongie, Convolutional networks with adaptive inference graphs, in: *Proceedings of the European Conference on Computer Vision*, 2018, pp. 3–18.
- [15] Y. Bai, S. S. Bhattacharyya, A. P. Happonen, H. Huttunen, Elastic neural networks: A scalable framework for embedded computer vision, in: *Proceedings of the European Signal Processing Conference*, 2018, pp. 1472–1476.
- [16] Y. Zhou, Y. Bai, S. S. Bhattacharyya, H. Huttunen, Elastic neural networks for classification, in: *Proceedings of the IEEE International Conference on Artificial Intelligence Circuits and Systems*, 2019, pp. 251–255.

- [17] J. Sivic, A. Zisserman, Video google: A text retrieval approach to object matching in videos, in: Proceedings of the IEEE International Conference on Computer Vision, 2003, pp. 1470–1477.
- [18] D. G. Lowe, Object recognition from local scale-invariant features, in: Proceedings of the IEEE International Conference on Computer Vision, Vol. 2, 1999, pp. 1150–1157.
- [19] N. Passalis, A. Tefas, Learning bag-of-features pooling for deep convolutional neural networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 5766–5774.
- [20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Computer Vision and Pattern Recognition, 2015, pp. 1–9.
- [21] A. Iosifidis, A. Tefas, I. Pitas, Discriminant bag of words based representation for human action recognition, Pattern Recognition Letters 49 (2014) 185–192.
- [22] N. Passalis, A. Tefas, Neural bag-of-features learning, Pattern Recognition 64 (2017) 277 – 294.
- [23] N. Passalis, A. Tefas, Learning bag-of-embedded-words representations for textual information retrieval, Pattern Recognition 81 (2018) 254–267.
- [24] N. Passalis, J. Raitoharju, A. Tefas, M. Gabbouj, Adaptive inference using hierarchical convolutional bag-of-features for low-power embedded platforms, in: Proceedings of the IEEE International Conference on Image Processing, 2019, pp. 3048–3052.
- [25] S. Pancoast, M. Akbacak, Bag-of-audio-words approach for multimedia event classification, in: Proceedings of the Conference of the International Speech Communication Association, 2012.
- [26] N. Passalis, A. Tsantekidis, A. Tefas, J. Kannianen, M. Gabbouj, A. Iosifidis, Time-series classification using neural bag-of-features, in: Proceedings of the European Signal Processing Conference, 2017, pp. 301–305.
- [27] S. Bhattacharya, R. Sukthankar, R. Jin, M. Shah, A probabilistic representation for efficient large scale visual recognition tasks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2011, pp. 2593–2600.
- [28] J. C. Van Gemert, J.-M. Geusebroek, C. J. Veenman, A. W. Smeulders, Kernel codebooks for scene categorization, in: Proceedings of the European Conference on Computer Vision, 2008, pp. 696–709.

- 435 [29] N. Passalis, A. Tefas, J. Kannianen, M. Gabbouj, A. Iosifidis, Temporal logistic neural bag-of-features for financial time series forecasting leveraging limit order book data, arXiv preprint 1901.08280 (2019).
- [30] K. He, X. Zhang, S. Ren, J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, in: Proceedings of the of the European Conference on Computer Vision, 2014, 440 pp. 346–361.
- [31] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, in: Proceedings of the NIPS Deep Learning and Representation Learning Workshop, 2015.
- [32] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.
- 445 [33] H. Xiao, K. Rasul, R. Vollgraf, Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms, arXiv preprint 1708.07747 (2017).
- [34] A. Krizhevsky, Learning multiple layers of features from tiny images, Tech. rep., University of Toronto (2009).
- [35] O. Arriaga, M. Valdenegro-Toro, P. Plöger, Real-time convolutional neural networks for emotion and gender classification, arXiv preprint 1710.07557 (2017). 450
- [36] D. Hughes, M. Salathé, et al., An open access repository of images on plant health to enable the development of mobile disease diagnostics, arXiv preprint 1511.08060 (2015).
- [37] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520. 455
- [38] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, in: Proceedings of the Advances in Neural Information Processing Systems, 2019, pp. 8024–8035.
- 460 [39] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700–4708.

## **7.2 Efficient Adaptive Inference leveraging Bag-of-Features-based Early Exits**

The appended paper follows.

# Efficient Adaptive Inference Leveraging Bag-of-Features-based Early Exits

Nikolaos Passalis<sup>1</sup>, Jenni Raitoharju<sup>2</sup>, Moncef Gabbouj<sup>3</sup> and Anastasios Tefas<sup>1</sup>

<sup>1</sup>Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece

<sup>2</sup>Programme for Environmental Information, Finnish Environment Institute, Jyväskylä, Finland

<sup>3</sup>Faculty of Information Technology and Communication Sciences, Tampere University, Tampere, Finland

E-mails: passalis@csd.auth.gr, jenni.raitojarju@tuni.fi, moncef.gabbouj@tuni.fi, tefas@csd.auth.gr

**Abstract**—Early exits provide an effective way of implementing adaptive computational graphs over deep learning models. In this way it is possible to adapt them on-the-fly to the available computational resources or even to the difficulty of each input sample, reducing the energy and computational power requirements in many embedded and mobile applications. However, performing this kind of adaptive inference also comes with several challenges, since the difficulty of each sample must be estimated and the most appropriate early exit must be selected. It is worth noting that existing approaches often lead to highly unbalanced distributions over the selected early exits, reducing the efficiency of the adaptive inference process. At the same time, only a few resources can be devoted to the aforementioned process, in order to ensure that an adequate speedup will be obtained. The main contribution of this work is to provide an easy to use and tune adaptive inference approach for early exits that can overcome some of these limitations. In this way, the proposed method allows for a) obtaining a more balanced inference distribution among the early exits, b) relying on a single and interpretable hyperparameter for tuning its behavior (ranging from faster inference to higher accuracy), and c) improving the performance of the networks (increasing the accuracy and reducing the time needed for inference). Indeed, the effectiveness of the proposed method over existing approaches is demonstrated using four different image datasets.

**Index Terms**—Early Exits, Adaptive Inference, Bag-of-Features, Adaptive Computational Graphs

## I. INTRODUCTION

Deep Learning (DL) led to a number of impressive applications, ranging from autonomous cars [1] and robotics [2] to accurate medical diagnosis and disease prognosis [3]. However, despite its success in these areas, state-of-the-art DL models are computationally intensive, requiring an enormous amount of resources in order to be successfully deployed in most embedded and mobile applications. These limitations led to the development of various methods for training lightweight DL models, ranging from quantization [4] and pruning methods [5] to knowledge distillation approaches [6], [7]. These methods were indeed capable of providing faster and more lightweight models. However, this was typically achieved at the expense of the accuracy of the final models, since lightweight models tend to perform worse than the larger ones. At the same

time, the constructed models are unable to dynamically adapt to the difficulty of the input samples, leading to a constant cost for feed-forwarding through the network. However, it has been demonstrated that easier samples can be often classified correctly even with significantly smaller models [8], [9]. This hints at using models that can switch between using more layers/computational resources for harder examples, while keeping the computations to the minimum required for easier examples. In this way, it is possible to reduce the computational requirements and energy consumption, as well as improve inference speed, often without significantly affecting the accuracy of the model.

Indeed, models with *adaptive computational graphs*, such as [8], [9], [10], have been proposed to this end. These models provide an effective way to dynamically alter the number of computations to match the available computational resources by following a different path on the model’s graph. Perhaps the most straightforward way to provide such adaptive models is by using *early exits* at various layers of the network [8], [9], [10]. In this way it is possible to estimate the final output of a model, without feed-forwarding through the whole computational graph. However, even though early exits proved to be a valuable tool for adapting the models to the available computational resources and different inference scenarios, e.g., using the same model across different mobile devices, using them for adapting to the difficulty of each sample is usually not straightforward. There are two main reasons for this: a) early exits do not provide a way for directly estimating the difficulty of an input sample, while b) designing and employing a meaningful exit strategy that utilizes all the available exits in the optimal way is non-trivial. In fact, as we also experimentally demonstrate, naive methods tend to use only a small number of the exits that are available, often skipping intermediate exits, leading to sub-optimal results and reducing inference speed and accuracy.

To better understand these limitations, we need to consider the way existing methods perform adaptive inference. Most of them estimate the difficulty of a sample indirectly, by measuring the *confidence* of the network at a specific early exit. This can be done either by measuring the strongest activation or by measuring the entropy of the output probability distribution [8]. After that, they employ a *fixed* threshold at each early exit to decide whether the specific exit should be

This work was supported by the European Union’s Horizon 2020 Research and Innovation Program (OpenDR) under Grant 871449. This publication reflects the authors’ views only. The European Commission is not responsible for any use that may be made of the information it contains.

used or not. When the confidence level is above a certain threshold, the computation ends at the specific early exit. It is also worth noting that even though more sophisticated methods can be developed for estimating the difficulty of each sample and/or the confidence of the network, it is critical to keep the required computations to the minimum, since spending a significant amount of time deciding whether the computations should end at a specific exit, might end up consuming more time than feed-forwarding through the whole network.

At the same time, the aforementioned process typically involves manually fine-tuning these thresholds for each early exit in order to achieve specific inference goals, either regarding the speed or the accuracy of the network. However, these thresholds are typically application specific and there is no easy way to select and interpret them in order to tune the behavior of the network. Finally, it is worth noting that sub-optimally tuning these thresholds, e.g., by simply selecting the average confidence of a layer, often leads to a significantly skewed exit distribution, since the confidence of the network is typically not normally distributed. This behavior is indeed confirmed in Section III.

The main contribution of this work is to provide an easy to use and tune adaptive inference approach for early exits. To this end, the proposed method combines several methodological advances to overcome a number of limitations of existing methods. First, instead of using naive early exits that discard a significant amount of information regarding the input distribution [9], we employ a powerful Bag-of-Features (BoF)-based early exits strategy. This, allows us to provide compact and normalized histogram-based representations of the features extracted from each layer, where an early exit is placed, which, in turns, enables the efficient estimation of the difficulty of each sample. Then, a fast and robust way to estimate the confidence of the network at each early exit is employed and combined with an easy to use, yet effective methodology for tuning the behavior of the method according to different inference scenarios. In this way, the proposed method allows for a) obtaining a more balanced inference distribution among the early exits, b) relying on a single and interpretable hyper-parameter for tuning its behavior (ranging from faster inference to higher accuracy), and c) improving the performance of the networks (increase the accuracy and require less time for inference). To the best of our knowledge, this paper presents the first experimental study that demonstrates the importance of employing methods that lead to a balanced exit distribution, while also exploiting these observations to provide a dynamic inference scheme for BoF-based exits that can be easily adjusted to the needs of each application. The effectiveness of the proposed approach over existing approaches is demonstrated using four different image datasets.

The rest of the paper is structured as follows. The proposed method is introduced in Section II, while the experimental evaluation is provided in Section III. Finally, conclusions are drawn in Section IV.

## II. PROPOSED METHOD

The proposed method is presented in this Section. First, the notation and the employed early exit strategy are described. Then, the proposed adaptive inference approach is introduced and discussed. The notation  $f_{\mathbf{W}}(\mathbf{x}, i)$  is used to refer to the response of the  $i$ -th layer of a neural network that is composed of a total of  $m$  layers. The trainable parameters of the network are denoted by  $\mathbf{W}$ , while the input to the neural network is denoted by  $\mathbf{x}$ . In this paper, we focus on convolutional neural networks. Therefore, the input to the network is an image  $\mathbf{x} \in \mathbb{R}^{W \times H \times C}$ , where its width is denoted by  $W$ , its height by  $H$ , and the number of channels by  $C$ . It is worth noting that the proposed method can be also applied for networks operating on different kinds of signals. Furthermore, let  $\mathbf{y}^{(i)} = f_{\mathbf{W}}(\mathbf{x}, i) \in \mathbb{R}^{W_i \times H_i \times C_i}$  denote the output of the  $i$ -th convolutional layer. Similarly, the notation  $W_i$ ,  $H_i$  and  $C_i$  is used to refer to the width, height and number of channels of the corresponding feature map. Finally, the output of the network, which estimates the probability of each sample belonging to a different class (out of a total of  $N_C$  classes), is denoted by  $\mathbf{y} = f_{\mathbf{W}}(\mathbf{x}, m) \in \mathbb{R}^{N_C}$ .

Also, let  $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$  denote a training set of  $N$  images. Each image is also annotated by a target (ground truth annotation) vector  $\mathbf{t}_i \in \mathbb{R}^{N_C}$ . The network can be then trained using back-propagation to minimize a loss function  $\mathcal{L}$ :

$$\mathbf{W}' = \mathbf{W} - \eta \sum_{j=1}^N \frac{\partial \mathcal{L}(f_{\mathbf{W}}(\mathbf{x}_j, m), \mathbf{t}_j)}{\partial \mathbf{W}}. \quad (1)$$

The notation  $\mathbf{W}'$  is used to refer to the parameters of the network after an update, while  $\eta$  denotes the learning rate. For this paper, the cross-entropy loss function is used:

$$\mathcal{L}(\mathbf{y}, \mathbf{t}) = - \sum_{i=1}^{N_C} [\mathbf{t}]_i \log([\mathbf{y}]_i), \quad (2)$$

where the notation  $[\mathbf{y}]_i$  is used to refer to the  $i$ -th element of a vector  $\mathbf{y}$ .

In order to efficiently estimate the output of the network at various points of its computational graph, we employ an additional estimator  $g_{\mathbf{W}_i}^{(i)}(\cdot)$  on top of the feature maps extracted at the  $i$ -th layer as:

$$g_{\mathbf{W}_i}^{(i)}(\mathbf{y}^{(i)}) = g_{\mathbf{W}_i}^{(i)}(f_{\mathbf{W}_i}(\mathbf{x}, i)). \quad (3)$$

In this way, it is possible to estimate the final output of the network, without having to feed-forward through the whole architecture. We use the notation  $\mathbf{W}_i$  to refer to the parameters of the employed early exit. Typically, an early exit is composed of a feature aggregation approach, e.g., Global Average Pooling, followed by a fully connected classification layer. Note that the effectiveness and efficiency of the feature aggregation approach is crucial for the successful deployment of the resulting network. For example, the feature maps extracted from the first layers are typically very large and close to the size of the input image, requiring a way to efficiently compress

them into a compact representation that can be used to rapidly take a classification decision.

To overcome this limitation, in this work we employ a Bag-of-Features (BoF)-based aggregation approach [9]. Therefore, each feature vector extracted from each spatial location of feature map is first quantized using a set of  $N_K$  codewords, each one denoted by  $\mathbf{v}_{ij}$ , where  $i$  refers to the layer on which the BoF layer is used and  $j$  to the codewords. The codewords are used to represent the prototypical concepts captured by the corresponding layer [11], while a different set of codewords is used for different exits. The membership vector for each feature vector extracted from the  $i$ -th early exit is calculated as:

$$[\mathbf{u}_{ikl}]_j = \frac{K([\mathbf{y}^{(i)}]_{kl}, \mathbf{v}_{ij})}{\sum_{m=1}^{N_K} K([\mathbf{y}^{(i)}]_{kl}, \mathbf{v}_{im})} \in [0, 1], \quad (4)$$

where  $(k, l)$  is the location from which the feature vector was extracted and  $K(\cdot)$  is a kernel function that measures the similarity between a codeword and an input vector. Then, the membership vectors  $\mathbf{u}_{ikl}$  are aggregated, leading to the final constant length histogram representation of the  $i$ -th object as:

$$\mathbf{s}^{(i)} = \frac{1}{W_i H_i} \sum_{k=1}^{W_i} \sum_{l=1}^{H_i} \mathbf{u}_{ikl} \in \mathbb{R}^{N_K}, \quad (5)$$

Note that this histogram vector provides a compact semantic summary of the features extracted from the corresponding layer, allowing for efficiently performing classification tasks, regardless of the actual size of the input feature map.

Furthermore, BoF can be efficiently implemented in DL models by measuring the similarity between each codeword and each feature vector using a hyperbolic (sigmoid) kernel:

$$K([\mathbf{y}^{(i)}]_{kl}, \mathbf{v}_{ij}) = \frac{1}{2} \left( \tanh(c_1 [\mathbf{y}^{(i)}]_{kl}^T \mathbf{v}_{ij} + c_2) + 1 \right) \quad (6)$$

where  $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ , and  $c_1$  and  $c_2$  are the kernel parameters (typically set to  $c_1 = 1$  and  $c_2 = 0$ ). Then, BoF can be formulated as an inner product-based layer, followed by a recurrent accumulator [12]. The representation extracted from each early exit is then fed into a fully connected layer to obtain the final output of the  $i$ -th early exit. Each early exit is trained by minimizing the same loss as the main network:

$$\mathbf{W}'_i = \mathbf{W}_i - \eta \sum_{j=1}^N \frac{\partial \mathcal{L} \left( g_{\mathbf{W}_i}^{(i)}(\mathbf{y}_j), \mathbf{t}_j \right)}{\partial \mathbf{W}_i}. \quad (7)$$

Also, early exits can be either trained at the same time with the main network or separately, after first fixing the weights of the main network [9]. However, back-propagating gradients from the early exits to the main network can potentially negatively affect its performance in some cases by forcing the earlier layers to be more discriminative, harming in this way the granularity of the analysis performed by the DL model. Therefore, even though both approaches can be used, in this paper we follow the second approach by keeping the parameters of the main network frozen when training the early exits.

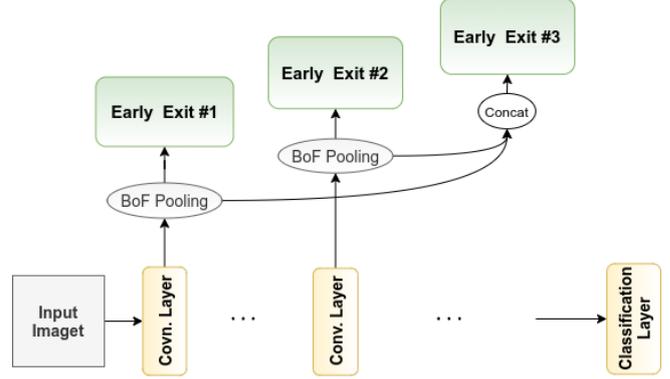


Fig. 1: Hierarchical Early Exits using Bag-of-Features. Note that any intermediate early exit can be selected to reduce the computational complexity of the inference process, given that the network is confident enough for the category of the current input sample.

The representations extracted from the previous layers can be also readily re-used by concatenating them with the current representation, building in this way a hierarchical inference structure, as shown in Fig. 1. Therefore, the representation  $\mathbf{s}^{(i,h)}$  extracted from the  $i$ -th layer is calculated as:

$$\mathbf{s}^{(i,h)} = \begin{cases} \mathbf{s}^{(i)} & \text{if } i = 1 \\ \mathbf{s}^{(i)} \frown \mathbf{s}^{(i-1,h)} & \text{if } i > 1 \end{cases}, \quad (8)$$

where  $\mathbf{a} \frown \mathbf{b}$  denotes the concatenation of vectors  $\mathbf{a}$  and  $\mathbf{b}$ . This approach allows for increasing the classification accuracy, with virtually zero additional cost, since the representations extracted from the previous layers can be readily cached and re-used.

In order to perform adaptive inference using a network equipped with early exits, an appropriate criterion must be used to decide whether inference should stop at a specific exit or continue until the next one. To this end, the difficulty of the input sample must be first estimated. In this work, we employ a simple, yet robust approach: the uncertainty of the network/difficulty  $r(\mathbf{x}, i)$  of an input sample  $\mathbf{x}$ , given the  $i$ -th exit, is estimated based on the confidence of the network on the class that corresponds to the neuron with the highest activation:

$$r(\mathbf{x}, i) = [g_{\mathbf{W}_i}^{(i)}(\mathbf{y}^{(i)})]_k, \quad (9)$$

where

$$k = \arg_{k'} \max [g_{\mathbf{W}_i}^{(i)}(\mathbf{y}^{(i)})]_{k'}. \quad (10)$$

Then, the most straightforward approach is to calculate a threshold for the  $i$ -th early exit based on the mean activation of the winning neurons during inference:

$$\mu_i = \frac{1}{N} \sum_{j=1}^N [g_{\mathbf{W}_i}^{(i)}(\mathbf{y}_j)]_k, \quad (11)$$

where  $g_{\mathbf{W}_i}^{(i)}$  is the output of the  $i$ -th early exit and again  $k = \arg \max [g_{\mathbf{W}_i}^{(i)}(\mathbf{y}_j)]_k$ . Then, the inference process can stop at the

$i$ -th exit whenever the confidence is higher than this threshold, i.e.,

$$r(\mathbf{x}, i) > \alpha\mu_i, \quad (12)$$

where  $\alpha$  is a hyper-parameter that allows for adapting the behavior of the inference process to the needs of each application. That is, using larger values for  $\alpha$  enables us to get more accurate result (at the expense of higher inference time, since the corresponding early exit will be chosen less frequently), while using smaller values allows for increasing the speed, but possibly leading to less accurate classification results.

Note that the value of  $\alpha$  is not bounded, since any positive number can be used. We have experimentally found out that this approach is also especially sensitive, since even small changes in the value of  $\alpha$  can lead to significant changes in the behavior of the inference process, as we also experimentally demonstrate in Section III. Furthermore, this behavior seems to be also related to the specific dataset/network used for training, with some datasets/networks requiring vastly different values compared to others. To overcome this limitation, we propose calculating the threshold based both on the mean activation of the current layer  $\mu_i$  (lower bound), as well as on the last layer of the network  $\mu_C$  (upper bound). Therefore, the threshold for selecting a specific early exit is calculated as:

$$r(\mathbf{x}, i) > (1 - \alpha)\mu_i + \alpha\mu_C, \quad (13)$$

for  $\alpha \in [0, 1]$ . In this way,  $\alpha$  is bounded between 0 and 1, while its value can be easily interpreted: as the value of  $\alpha$  increases from 0 to 1, less samples are using the current exit. Indeed, as demonstrated in Section III, the proposed way for calculating the threshold allows for acquiring significantly better performance, avoiding collapse phenomena where only a few of the early exits are actually used.

### III. EXPERIMENTAL EVALUATION

The experimental evaluation of the proposed approach is provided in this Section. Four datasets were used for evaluating the proposed method: a) the MNIST image classification dataset [13], b) the Fashion MNIST fashion product classification dataset [14], as well as the more challenging c) CIFAR-10 object recognition dataset [15], and d) FER-2013 facial expression dataset [16].

We used three different convolutional neural network architectures for the conducted experiments:

- **CNN-1**, which was used for the MNIST dataset,
- **CNN-2**, which was used for the Fashion MNIST dataset,
- **MobileNet v.2** [17], which was used for the CIFAR-10 and FER-2013 datasets.

CNN-1 employs a  $3 \times 3$  convolution layer with 32 filters, which is followed by  $2 \times 2$  max pooling layer, a  $3 \times 3$  convolution layer with 64 filters, another  $2 \times 2$  max pooling layer, a fully connected layer with 1024 neurons, dropout with rate  $p = 0.5$ , and a final fully connected classification layer. CNN-2 uses a  $3 \times 3$  convolution layer with 64 filters, which is followed by a  $2 \times 2$  max pooling layer, a  $3 \times 3$  convolution layer with 128 filters, another  $2 \times 2$  max pooling layer, a fully connected layer

TABLE I: Classification error and MFLOPs for the backbone networks (without using any early exit)

Dataset	Error	MFLOPs
MNIST	0.68	4.10
Fashion MNIST	7.82	12.66
CIFAR-10	7.81	94.61
FER-2013	38.84	211.52

with 1024 neurons layer, dropout with rate  $p = 0.5$ , and a final fully connected classification layer. All the layers (except from the final one) employ the ReLU activation function. For the MobileNet the filter size for the first convolutional layer was appropriately tuned, according to the image size of different datasets.

All networks were trained using the categorical cross-entropy loss. The Adam algorithm was used for the optimization using the default hyper-parameters [18]. The optimization ran for 50 epochs for the CNN-1 and CNN-2 models (a learning rate of 0.001 was used). On the other hand, MobileNet models were trained for 100 epochs (50 epochs with a learning rate of 0.001 and the remaining ones using a reduced learning rate of 0.0001). Two early exits and one hierarchical exit (combining the previous two ones) were used. For the CNN-1 and CNN-2 models, the first early exit was placed after the first convolutional layer, while the second one was placed after the second convolutional layer. The 5th and 7th convolutional layers were used for the MobileNet architecture. Finally, for the MNIST and FashionMNIST datasets, a spatial segmentation scheme into 4 regions was used for the BoF model employed for the early exits [11].

The proposed method is evaluated in Table II. Four different inference settings were used, ranging from faster, yet less accurate settings 1 and 2 to the slower, yet more accurate settings 3 and 4. These settings correspond to altering the value of the hyper-parameter  $\alpha$  used to calculate the inference threshold. For the baseline method, as described by (12), the value of  $\alpha$  was set to 1, 1.05, 1.07, and 1.095 for the four different settings. These values were obtained after carefully fine-tuning the upper limit in order to not collapse the inference process into using only some of the employed early exits. For the proposed method, the corresponding values were set to 0, 0.5, 0.7, and 0.95. Note that selecting these values is much easier for the proposed method, since  $\alpha$  always ranges between 0 and 1, while this value is not expected to be dataset-specific, compared to the baseline one. Also, two different BoF settings were used: BoF-1 and BoF-2. For the MNIST and FashionMNIST datasets, 8 codewords were used for each early exit, while for BoF-2 16 codewords were used (32 for the MNIST dataset). Also, 16/32 codewords were used for BoF-1/BoF-2 for the experiments involving the MobileNet.

Several interesting conclusions can be drawn based on the results reported in Table II. Note that for the first setting, both methods provide the same results, since setting  $\alpha = 1$  for the baseline is equivalent to setting  $\alpha = 0$  for the proposed method. First, note that all methods lead to a significant

TABLE II: Classification error (%) and MFLOPs for different adaptive inference settings

Dataset	Method	Setting 1		Setting 2		Setting 3		Setting 4		Class. Cost
		Error (%)	MFLOPs							
MNIST										
Baseline	BoF-1	1.32	0.69 ± 1.03	0.89	1.17 ± 1.65	0.79	1.54 ± 1.83	0.68	4.16 ± 0.00	4.0449
Proposed	BoF-1	1.32	0.69 ± 1.03	0.97	0.86 ± 1.20	0.82	1.02 ± 1.31	0.70	1.68 ± 1.55	2.3989
Baseline	BoF-2	1.09	0.68 ± 0.83	0.68	4.34 ± 0.00	0.68	4.34 ± 0.00	0.68	4.34 ± 0.00	10.5819
Proposed	BoF-2	1.09	0.68 ± 0.83	0.95	0.77 ± 0.94	0.86	0.85 ± 1.02	0.76	1.22 ± 1.25	4.2937
Fashion MNIST										
Baseline	BoF-1	9.54	3.62 ± 4.94	9.26	4.10 ± 5.21	9.12	4.31 ± 5.32	8.83	4.62 ± 5.45	10.4730
Proposed	BoF-1	9.54	3.62 ± 4.94	8.55	5.01 ± 5.52	8.26	5.92 ± 5.70	7.84	8.29 ± 5.49	4.8555
Baseline	BoF-2	8.95	3.51 ± 4.85	8.59	4.05 ± 5.16	8.42	4.28 ± 5.28	8.27	4.64 ± 5.46	8.7200
Proposed	BoF-2	8.95	3.51 ± 4.85	8.29	4.63 ± 5.38	8.10	5.34 ± 5.58	7.84	7.34 ± 5.70	6.6399
CIFAR-10										
Baseline	BoF-1	10.24	53.17 ± 16.49	9.52	55.59 ± 18.90	9.21	58.24 ± 21.52	9.05	61.45 ± 23.83	61.7983
Proposed	BoF-1	10.23	53.18 ± 16.50	9.06	56.93 ± 18.78	8.55	59.32 ± 19.76	7.87	65.71 ± 20.66	37.2700
Baseline	BoF-2	9.07	52.72 ± 16.32	8.52	58.93 ± 22.81	8.42	59.78 ± 23.20	8.21	61.09 ± 23.71	90.0544
Proposed	BoF-2	9.06	52.74 ± 16.34	8.24	55.59 ± 18.26	8.07	57.50 ± 19.19	7.78	62.06 ± 20.41	58.1173
FER-2013										
Baseline	BoF-1	43.58	131.27 ± 45.54	42.94	135.89 ± 47.59	42.57	137.69 ± 48.33	42.30	139.97 ± 49.09	152.8386
Proposed	BoF-1	43.58	131.27 ± 45.54	38.87	172.93 ± 47.77	38.67	183.43 ± 43.72	38.95	199.48 ± 31.20	39.0866
Baseline	BoF-2	41.74	131.74 ± 45.49	41.46	136.34 ± 47.42	41.18	138.33 ± 48.22	40.87	140.27 ± 48.80	299.9848
Proposed	BoF-2	41.74	131.74 ± 45.49	38.79	167.02 ± 48.87	38.84	178.49 ± 46.01	38.59	195.66 ± 35.08	60.0961

TABLE III: Distribution over the early exits for different inference settings

Dataset	Method	Setting 1		Setting 2		Setting 3		Setting 4		Unused/Least Used Exits
MNIST										
Baseline	BoF-1	0.84 - 0.10 - 0.02 - 0.04	0.77 - <b>0.00</b> - <b>0.00</b> - 0.23	0.67 - <b>0.00</b> - <b>0.00</b> - 0.33	<b>0.00</b> - <b>0.00</b> - <b>0.00</b> - 1.00	7				
Proposed	BoF-1	0.84 - 0.10 - 0.02 - 0.04	0.78 - 0.12 - 0.02 - 0.07	0.73 - 0.15 - 0.03 - 0.09	0.51 - 0.23 - 0.08 - 0.19	<b>0</b>				
Baseline	BoF-2	0.89 - 0.08 - <b>0.01</b> - 0.02	<b>0.00</b> - <b>0.00</b> - <b>0.00</b> - 1.00	<b>0.00</b> - <b>0.00</b> - <b>0.00</b> - 1.00	<b>0.00</b> - <b>0.00</b> - <b>0.00</b> - 1.00	10				
Proposed	BoF-2	0.89 - 0.08 - <b>0.01</b> - 0.02	0.85 - 0.10 - <b>0.01</b> - 0.03	0.83 - 0.12 - 0.02 - 0.03	0.68 - 0.23 - 0.03 - 0.06	<b>2</b>				
Fashion MNIST										
Baseline	BoF-1	0.71 - 0.11 - 0.02 - 0.16	0.67 - 0.11 - 0.02 - 0.20	0.66 - 0.11 - <b>0.01</b> - 0.21	0.63 - 0.11 - <b>0.01</b> - 0.24	2				
Proposed	BoF-1	0.71 - 0.11 - 0.02 - 0.16	0.59 - 0.12 - 0.03 - 0.26	0.52 - 0.12 - 0.03 - 0.33	0.32 - 0.11 - 0.05 - 0.52	<b>0</b>				
Baseline	BoF-2	0.73 - 0.11 - 0.02 - 0.14	0.69 - 0.11 - 0.02 - 0.19	0.67 - 0.11 - 0.02 - 0.21	0.64 - 0.11 - <b>0.01</b> - 0.24	1				
Proposed	BoF-2	0.73 - 0.11 - 0.02 - 0.14	0.63 - 0.12 - 0.03 - 0.22	0.57 - 0.12 - 0.03 - 0.27	0.41 - 0.11 - 0.05 - 0.43	<b>0</b>				
CIFAR-10										
Baseline	BoF-1	0.73 - 0.15 - <b>0.01</b> - 0.11	0.70 - 0.13 - <b>0.00</b> - 0.17	0.68 - 0.07 - <b>0.00</b> - 0.25	0.66 - <b>0.00</b> - <b>0.00</b> - 0.34	5				
Proposed	BoF-1	0.73 - 0.15 - <b>0.01</b> - 0.11	0.63 - 0.19 - <b>0.01</b> - 0.17	0.57 - 0.22 - <b>0.01</b> - 0.21	0.39 - 0.30 - 0.02 - 0.30	<b>3</b>				
Baseline	BoF-2	0.76 - 0.13 - <b>0.01</b> - 0.11	0.71 - <b>0.00</b> - <b>0.00</b> - 0.29	0.70 - <b>0.00</b> - <b>0.00</b> - 0.30	0.67 - <b>0.00</b> - <b>0.00</b> - 0.33	7				
Proposed	BoF-2	0.76 - 0.13 - <b>0.01</b> - 0.11	0.68 - 0.16 - <b>0.01</b> - 0.15	0.62 - 0.19 - <b>0.01</b> - 0.18	0.49 - 0.25 - <b>0.01</b> - 0.24	<b>4</b>				
FER-2013										
Baseline	BoF-1	0.60 - 0.17 - <b>0.01</b> - 0.22	0.56 - 0.17 - <b>0.01</b> - 0.26	0.54 - 0.17 - <b>0.01</b> - 0.27	0.52 - 0.17 - <b>0.01</b> - 0.29	4				
Proposed	BoF-1	0.60 - 0.17 - <b>0.01</b> - 0.22	0.22 - 0.18 - 0.02 - 0.58	0.15 - 0.15 - 0.02 - 0.68	0.05 - 0.08 - <b>0.01</b> - 0.85	<b>2</b>				
Baseline	BoF-2	0.59 - 0.18 - <b>0.01</b> - 0.22	0.55 - 0.18 - <b>0.01</b> - 0.25	0.53 - 0.18 - <b>0.01</b> - 0.27	0.52 - 0.18 - 0.02 - 0.29	3				
Proposed	BoF-2	0.59 - 0.18 - <b>0.01</b> - 0.22	0.26 - 0.21 - 0.02 - 0.51	0.18 - 0.17 - 0.02 - 0.63	0.07 - 0.11 - <b>0.01</b> - 0.80	<b>2</b>				

The percentage of the samples that use each of the 4 available exits is reported for each different setting. The first number corresponds to the first early exit, the second number to the second early exit, the third number to the hierarchical exit, while the last number to the final exit of the network.

reduction in the number of required FLOPs compared to the full networks, as reported in Table I. At the same time, the proposed method allows for a more fine-grained control over the behavior of the proposed method, in virtually every case, as demonstrated by the number of FLOPs for different settings. For example, note that for MNIST dataset (BoF-2), baseline collapses to using the final exit even for the second setting. At the same time, the proposed method does not lead to this behavior in any of the evaluated cases, even for the last setting. These results are also further validated in Table III, where the distribution of the used early exits are reported. The proposed method leads to a smoother distribution, avoiding phenomena where an early exit is rarely selected. Indeed, this can be easily verified by measuring the number of cases where an exit is selected for less than 1% of the time (last column of Table III).

Classification cost, which is calculated by dividing the mean number of FLOPs by the absolute classification error improvement over the first setting, is also reported in Table II. The mean classification cost calculated using the three remaining settings (2 to 4) is reported. Note that smaller values indicate better utilization of the resources, since the cost essentially measures the number of FLOPs required for each percentage of accuracy improvement. Again, the ability of the proposed method to significantly improve the inference results is verified, with some cases, e.g., FER-2013 dataset, leading to an almost 5-fold increase in efficiency.

Finally, note that, in many cases, the employed hierarchical exit (third exit) is only rarely selected, as shown in Table III. This is expected, since it generally provides small accuracy improvements over the second early exit, as also noted in [9]. This could possibly hint into designing architectures that only employ early exits on selected paths of the inference graph, allowing for avoiding unnecessary calculations that do not contribute to increasing the overall performance of the network. It is also worth noting that the proposed method allows for identifying such exits, since it leads to a smoother distribution over the used exits. Therefore, any exit that is below a certain threshold can be, in most of the cases, safely discarded, providing an efficient way for selecting the most appropriate points of the computational graph to add early exits.

#### IV. CONCLUSIONS

In this paper, we presented an easy to use and tune, yet effective adaptive inference approach for networks equipped with BoF-based early exits. The proposed method employs a fast confidence estimation approach, along with a bounded way to calculate the exit thresholds for the various exits. As it was experimentally demonstrated, this allows for obtaining a more balanced inference distribution among the early exits increasing the accuracy and requiring less time for inference. At the same time, the proposed method paves the way for more advanced adaptive inference approaches for models equipped with early exits. For example, the proposed method provides an implicit way to estimate the layers on which early exits should be placed, avoiding the need for more expensive

neural architecture search methods, more advanced approaches for estimating the confidence for each sample, while neural network distillation methods [19] can be employed to further improve the performance of the proposed method.

#### REFERENCES

- [1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436, 2015.
- [2] Niko Sünderhauf, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, et al., "The limits and potentials of deep learning for robotics," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 405–420, 2018.
- [3] Baris Gecer, Selim Aksoy, Ezgi Mercan, Linda G Shapiro, Donald L Weaver, and Joann G Elmore, "Detection and classification of cancer in whole slide breast histopathology images using deep convolutional networks," *Pattern Recognition*, vol. 84, pp. 345–356, 2018.
- [4] Song Han, Huizi Mao, and William J Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proceedings of the International Conference on Learning Representations*, 2016.
- [5] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proc. IEEE International Conference on Computer Vision*, 2017, pp. 5058–5066.
- [6] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.
- [7] Nikolaos Passalis and Anastasios Tefas, "Learning deep representations with probabilistic knowledge transfer," in *Proc. European Conference on Computer Vision*, 2018, pp. 268–284.
- [8] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *Proc. International Conference on Pattern Recognition*, 2016, pp. 2464–2469.
- [9] Nikolaos Passalis, Jenni Raitoharju, Anastasios Tefas, and Moncef Gabbouj, "Adaptive inference using hierarchical convolutional bag-of-features for low-power embedded platforms," in *Proc. IEEE International Conference on Image Processing*, 2019, pp. 3048–3052.
- [10] Yue Bai, Shuvra S Bhattacharyya, Antti P Happonen, and Heikki Huttunen, "Elastic neural networks: A scalable framework for embedded computer vision," in *Proceedings of the European Signal Processing Conference*, 2018, pp. 1472–1476.
- [11] Nikolaos Passalis and Anastasios Tefas, "Learning bag-of-features pooling for deep convolutional neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5766–5774.
- [12] Nikolaos Passalis, Anastasios Tefas, Juho Kanninen, Moncef Gabbouj, and Alexandros Iosifidis, "Temporal logistic neural bag-of-features for financial time series forecasting leveraging limit order book data," *arXiv preprint 1901.08280 (2019)*.
- [13] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [14] Han Xiao, Kashif Rasul, and Roland Vollgraf, "Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint 1708.07747 (2017)*.
- [15] Alex Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., University of Toronto, 2009.
- [16] Octavio Arriaga, Matias Valdenegro-Toro, and Paul Plöger, "Real-time convolutional neural networks for emotion and gender classification," *arXiv preprint 1710.07557 (2017)*.
- [17] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.
- [18] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
- [19] Nikolaos Passalis, Maria Tzelepi, and Anastasios Tefas, "Heterogeneous knowledge distillation using information flow modeling," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2339–2348.

### **7.3 Leveraging Active Perception for Improving Embedding-based Deep Face Recognition**

The appended paper follows.

# Leveraging Active Perception for Improving Embedding-based Deep Face Recognition

Nikolaos Passalis and Anastasios Tefas  
Artificial Intelligence Information Analysis Laboratory  
Department of Informatics, Faculty of Sciences  
Aristotle University of Thessaloniki, Thessaloniki, Greece  
E-mail: {passalis, tefas}@csd.auth.gr

**Abstract**—Even though recent advances in deep learning (DL) led to tremendous improvements for various computer and robotic vision tasks, existing DL approaches suffer from a significant limitation: they typically ignore that robots and cyber-physical systems are capable of interacting with the environment in order to better sense their surroundings. In this work we argue that perceiving the world through physical interaction, i.e., employing active perception, allows for both increasing the accuracy of DL models, as well as for deploying smaller and faster models. To this end, we propose an active perception-based face recognition approach, which is capable of simultaneously extracting discriminative embeddings, as well as predicting in which direction the robot must move in order to get a more discriminative view. To the best of our knowledge, we provide the first embedding-based active perception method for deep face recognition. As we experimentally demonstrate, the proposed method can indeed lead to significant improvements, increasing the face recognition accuracy up to 9%, as well as allowing for using overall smaller and faster models, reducing the number of parameters by over one order of magnitude.

## I. INTRODUCTION

Deep Learning (DL) has led to tremendous improvements in recent years for various challenging computer vision tasks [1], including, but not limited to, object detection and recognition [2], scene segmentation [3], face recognition [4], and others. The advanced perception capabilities enabled by DL also provided powerful tools for various robotics tasks, leading to the development of spectacular applications, such as autonomous cars [5], drones [6], [7], and robots that can seamlessly interact with humans, e.g., collaborative manufacturing [8].

However, despite these recent achievements of DL in these areas, most of the existing methods suffer from a significant drawback: they follow a static inference paradigm, as inherited by the traditional computer vision pipeline. More specifically, DL models perform inference on a fixed and static input, ignoring that robots, as well as many cyber-physical systems [9], [10], have the ability of *interacting* with the environment in order to better sense their surroundings. For example, consider the task of face recognition, where a robot has acquired a sub-optimal profile view of a subject. An existing

static perception-based DL model might fail to recognize the subject from this view, especially if it has never been trained on profile face images. However, it is usually possible that the robot can acquire a better and more discriminative view by more appropriately repositioning itself with respect to the human subject. Therefore, in this case, the exact same DL model, will probably be able to recognize the subject, after the robot repositions itself in a more appropriate angle with respect to the subject. This approach, which is called *active perception* [11], [12], [13], allows for manipulating the robot/sensor in order to acquire a better and more clean view/signal, leading to improved situational awareness. It is worth noting that this process is very similar to the way humans and various animals interact and understand their environment. For example, humans tend to look from different angles when trying to process complex visual stimuli, while many mammals have specialized muscles that rotate their ears toward the source of an audio signal [14].

A number of recent, yet quite primitive approaches, demonstrated that active perception can indeed increase the perception capabilities of various models. For example, in [15] it is demonstrated that developing a deep learning system that also predicts the next best move for a robot, using reinforcement learning, can significantly improve the performance of object detection, where the viewing angle, occlusions and the scale of each object can have a significant effect on the object recognition accuracy. Similar observations were also reported by more recent works [16], [17], [18]. At the same time, it is worth noting that active perception approaches often allow for developing faster and more lightweight DL models, since models are trained in order to solve a simpler problem. For example, in the case of object detection [15], a simpler model can be trained just for recognizing the objects from a limited number of angles, since a robot can usually acquire a more appropriate view that allows for accurately recognizing the corresponding object. To the best of our knowledge, despite these encouraging results in these areas, there have not been any thoroughly study on developing deep learning-based active perception models for human-centric robot perception, such face recognition.

Motivated by the aforementioned observations, we examine two main hypotheses in this work. First, we argue that the recognition accuracy of DL models can be improved by

This work was supported by the European Union’s Horizon 2020 Research and Innovation Program (OpenDR) under Grant 871449. This publication reflects the authors’ views only. The European Commission is not responsible for any use that may be made of the information it contains.

acquiring a more appropriate view, after manipulating the position of a robot inside the world. For example, moving a robot closer to a human is expected to improve the confidence of the robot when recognizing a human, as well as reduce the recognition errors. At the same time, we hypothesize that a significant part of the complexity in modern DL models arise from their ability to perform view invariant inference. Therefore, we argue that significantly smaller models can be used when active perception approaches are employed, without reducing recognition accuracy.

The main contribution of this work is proposing a DL-based active perception method for embedding-based face recognition, as well as examining the behavior of such approach on a real multi-view face image dataset, shedding light on the aforementioned research questions. The proposed method is capable of simultaneously learning discriminative embeddings, that can disentangle the representations extracted from facial images that belong to different persons, as well as learning which should be the next control action by a robot carrying a camera in order to improve the face recognition confidence, as shown in Fig. 1. The proposed method does not rely on prior knowledge, such as that frontal views might lead to better recognition accuracy, and it is capable of autonomously learning how to acquire the best view in order to facilitate the task at hand. Therefore, the proposed formulation is generic and task agnostic and can be applied to virtually any DL-based recognition model, given that the appropriate simulation environment have been developed and/or the appropriate dataset have been collected.

Furthermore, the proposed method is computationally efficient, since it utilizes the same main network backbone both for extracting a discriminative embedding, as well as for predicting the next action that must be performed in order to increase the recognition confidence, as shown in Fig. 1. To this end, two different branches, an embedding branch and a control action branch, are employed, as further described in Section II. Additionally, instead of using a computationally intensive reinforcement learning-based approach for the optimization, similar to other active perception methods [15], [17], the proposed method employs a purely supervised learning approach, which can significantly accelerate the convergence of the method. It is worth noting that the proposed method is task agnostic and can be trivially implemented in the typical batch-based setting used for training DL models. Therefore, it can be directly used for most classification/regression tasks, extending beyond the face analysis applications presented in this paper, paving the way for providing generic active perception-enable DL models. Finally, to the best of our knowledge, this is the first deep learning-based active perception approach that allows for efficiently optimizing one unified architecture towards both learning discriminative embeddings, as well as performing control.

The rest of the paper is structured as follows. First, the proposed method is introduced and analytically derived in Section II, while an extensive experimental evaluation is provided in Section III. Finally, conclusions are drawn and

further research directions are discussed in Section IV.

## II. PROPOSED METHOD

The proposed method is presented in this Section. First, the necessary notation and a brief introduction to representation learning for face recognition is provided. Next, the proposed approach is presented and discussed in detail.

### A. Notation and Representation Learning

Let  $\mathbf{x}_i \in \mathbb{R}^{W \times H \times C}$  denote a (cropped) face image, where  $W$ ,  $H$  and  $C$  are the width, height and number of channels of the corresponding image. Also, let  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_i, \dots, \mathbf{x}_N\}$  be a collection of  $N$  training images, while the binary variable  $d_{ij} \in \{0, 1\}$  is introduced to denote whether the  $i$ -th face image belongs to the same person as the one depicted in the  $j$ -th face image. Most recent deep face recognition methods, e.g., [19], aim at learning an appropriate model  $\mathbf{y} = f_{\theta_r}(\mathbf{x})$  that will extract a discriminative identify-oriented representation from each face image by solving the following optimization problem:

$$\theta_r = \arg \min_{\theta} \sum_{i=1}^N \sum_{j=1, j \neq i}^N \mathcal{L}(f_{\theta}(\mathbf{x}_i), f_{\theta}(\mathbf{x}_j), d_{ij}) \quad (1)$$

Different loss functions  $\mathcal{L}(\cdot)$  have been proposed to this end. In this work, we employ the *contrastive* loss [20], [21], which is minimized when embeddings that belong to the same identity are as close as possible, while the representations of face images that do not belong to the same person maintain at least a distance of  $\sqrt{m}$ :

$$\mathcal{L}_c(\mathbf{y}_i, \mathbf{y}_j, d_{ij}) = d_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|_2^2 + (1 - d_{ij}) \max(0, m - \|\mathbf{y}_i - \mathbf{y}_j\|_2^2), \quad (2)$$

where  $\|\cdot\|_2$  refers to the  $l^2$  norm of a vector. After training the model  $\mathbf{y} = f_{\theta_r}(\mathbf{x})$ , the identity of a person depicted in a previously unseen image  $\mathbf{x}$  can be obtained simply by performing nearest neighbor search on a database that contains images  $\mathbf{x}_i$  of known identities, i.e.,  $\mathcal{X}_d = \{(\mathbf{x}_i, l_i)\}$ , where  $l_i$  is the identity of the person depicted in the  $i$ -th image. Therefore, during inference the identity  $l$  of a person appearing in a novel image  $\mathbf{x}$  is obtained as:

$$l = l_i, \text{ where } i = \arg \min_i \|f(\mathbf{x}_i) - f(\mathbf{x})\|_2 \quad (\forall (\mathbf{x}_i, l_i) \in \mathcal{X}_d). \quad (3)$$

### B. Active Perception for Face Recognition

Even though this static approach presented in the previous subsection allows for achieving quite impressive face recognition results, as well as for easily training the model using a collection of static images, it comes with an important drawback: it ignores the ability of robotic systems to interact with the environment in order to get a more discriminative view for the task at hand. For example, a drone carrying a camera can fly to the appropriate direction in order to acquire a more clean frontal view of a person, allowing for analyzing the input with greater confidence. To this end, we introduce

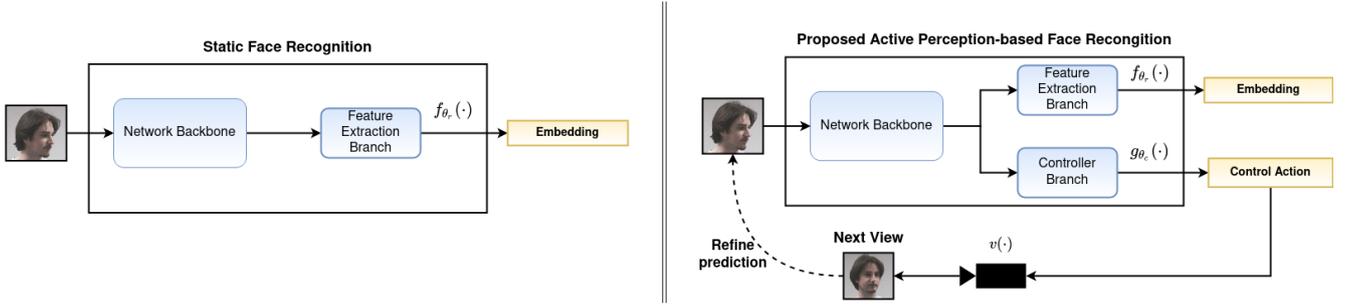


Fig. 1: Comparing the proposed active perception-based approach to static face recognition. We simultaneously train a DL model to predict both a discriminative feature vector, which is used for face recognition, as well as a high-level description for the next best action, which can be used for acquiring a better and more discriminative view of the input. Note that the model is *not* trained for facial pose estimation, yet it implicitly learns the control actions that will lead to the view that will provide the best recognition results.

a trainable *controller*  $\mathbf{a}_t = g_{\theta_c}(\mathbf{x}^{(t)})$ , where  $\theta_c$  is a set of trainable parameters for the controller model, that receives an observation (image)  $\mathbf{x}^{(t)}$  from the environment at time  $t$  and provides an appropriate control command  $\mathbf{a}_t$  to the robot. Then, the updated observation is obtained by *executing* the corresponding action  $\mathbf{a}_t$  as:

$$\mathbf{x}^{(t+1)} = v(\mathbf{a}_t, t), \quad (4)$$

where  $v(\cdot)$  is either a model of the environment that returns the result of a simulated action  $\mathbf{a}_t$  at time  $t$ , or the real environment, in the case of deploying the model into a real system, where we execute the corresponding action and get the updated observation. In this way, the controller  $g_{\theta_c}(\cdot)$  provides a way to actively interact with the environment in order to get updated sensory stimuli, that will, in turn, lead to more accurate predictions for the embedding extractor  $f_{\theta_r}(\cdot)$ . For the rest of the paper, we will refer to the environment  $v(\mathbf{a}_t, t)$  as  $v(\mathbf{a}_t)$  to avoid cluttering the used notation.

Both the feature extractor model  $f_{\theta_r}(\cdot)$ , as well as the controller model  $g_{\theta_c}(\cdot)$  must be appropriately trained for the task at hand. That is, the feature extractor model must be trained to extract discriminative embeddings, while the controller model must be trained in order to provide the appropriate control commands that will allow for getting a view that will maximize the face recognition accuracy. To this end, the optimization problem provided in (1) is updated following the proposed active perception setting:

$$\theta_r, \theta_c = \arg \min_{\theta_1, \theta_2} \sum_{i=1}^N \sum_{j=1, j \neq i}^N \mathcal{L}(f_{\theta_1}(v(g_{\theta_2}(\mathbf{x}_i))), f_{\theta_1}(\mathbf{x}_j), d_{ij}). \quad (5)$$

Note that the active perception controller is allowed to manipulate only the first input to the loss function, since the other one corresponds to the fixed version that is stored in the database. Ideally, the controller should be aware of the remaining images contained in the database, since this

information can be exploited in order to acquire a view that matches the view of the person that is stored in the database. However, in this paper we assume that all images in the database are acquired under similar conditions, allowing for ignoring possible view variations of the database images.

It is also worth noting that (5) provides (at least) two different ways to minimize the employed loss: a) either by learning a powerful view-invariant feature extractor  $f_{\theta_r}(\cdot)$ , or b) by jointly learning a feature extractor along with an appropriate controller that is capable of acquiring images that makes the recognition problem easier. For the deployment, and after training both models simultaneously, the robot can either output the most probable prediction according to (3), or first appropriately control its camera (e.g., by moving itself or controlling a gimbal) in order to acquire an updated view. It is also possible that the controller will not provide any suggested action, as further explained below. In this case, we assume that the robot has already obtained an optimal view and no action should be performed.

Without loss of generality, in this paper we assume that control will be performed in discrete steps on one (horizontal) axis, which lies along a sphere centered on the subject's face. The proposed approach can be directly extended to handle multiple axes as well. Therefore, the controller  $g_{\theta_c}(\cdot) \in \mathbb{R}^3$  support three possible actions:

- 1) "stay" ( $\mathbf{a}_t = (1, 0, 0)$ ),
- 2) "left" ( $\mathbf{a}_t = (0, 1, 0)$ ),
- 3) "right" ( $\mathbf{a}_t = (0, 0, 1)$ ),

with each of the three output neurons corresponding to the confidence of the controller for performing each of the three possible actions. Note that the controller  $g_{\theta_c}(\cdot)$  only provides high-level control commands, which must be appropriately translated into actual control commands by using an appropriate controller, e.g., an PID controller [6], [22].

Despite the updated formulation provided in (5), it is still not straightforward to directly optimize  $g_{\theta_c}(\cdot)$ , since the model  $v(\cdot)$  is usually not fully known and it is not differentiable.

Instead of using reinforcement learning (RL), which is typically used to solve such control problems [15], [17], in this work we opt for a simpler, yet more efficient approach, which arises from the following assumption/observation: the recognition confidence is expected to monotonically increase or decrease when moving towards the same direction (at least for small continuous intervals). We call this *smooth control manifold* assumption. Even though it is possible that this assumption does not hold in practice, this approach allows for significantly simplifying the optimization process, as well as increasing the face recognition accuracy, as we further demonstrate in Section III. Therefore, for each face image sampled during the training process, we propose executing all the three possible actions simultaneously (using the appropriate simulation environment/dataset) and observing the effect on the recognition confidence. Let  $\mathbf{x}_{i0}$ ,  $\mathbf{x}_{i1}$ , and  $\mathbf{x}_{i2}$  denote the updated facial image obtained after moving the robot to the left, right, or performing no action. Then, the training target for the controller can be trivially acquired by choosing the action that minimizes the distance between the representation of the correct face and the current face. Therefore, the controller target  $d_i^{(a)}$  for an image  $\mathbf{x}_i$  and a positive example  $\mathbf{x}_p$  is acquired as:

$$d_i^{(a)} = \arg \min_{k \in \{0,1,2\}} \|\mathbf{x}_{ik} - f(\mathbf{x}_p)\|_2. \quad (6)$$

For negative examples there are two options: a) either not training the controller with them, or b) training the controller in order to again minimize the distance between the embedding vectors (despite belonging to different persons). The motivation for the latter option is that the controller should always perform control in order to find the view that provides the best matching between face embeddings. In this work, the first option was selected, since it was experimentally shown to lead to slightly better recognition results.

Therefore, the loss to be minimized when optimizing the controller is defined as:

$$\mathcal{L}_g = \sum_{i=1}^N \sum_{j=1, j \neq i}^N d_{ij} \mathcal{L}_x(g_{\theta_c}(\mathbf{x}_i), d_i^{(a)}), \quad (7)$$

where  $\mathcal{L}_x$  denotes the categorical cross-entropy loss. The feature extractor can be still trained as before, i.e., by minimizing the loss:

$$\mathcal{L}_f = \sum_{i=1}^N \sum_{j=1, j \neq i}^N \mathcal{L}(f_{\theta_r}(\mathbf{x}_i), f_{\theta}(\mathbf{x}_j), d_{ij}), \quad (8)$$

as provided in (1). Therefore, the final loss is obtained as:

$$\mathcal{L} = \mathcal{L}_g + \mathcal{L}_f \quad (9)$$

Gradient descent is employed for optimizing both models as:

$$\Delta\theta_r = -\eta_r \frac{\partial \mathcal{L}}{\partial \theta_r}, \quad \text{and} \quad \Delta\theta_c = -\eta_c \frac{\partial \mathcal{L}}{\partial \theta_c}, \quad (10)$$

where  $\eta_r$  and  $\eta_c$  are the learning rates for the feature extraction and controller models respectively. For all the experiments conducted in this paper we set  $\eta_r = \eta_c = 10^{-3}$ , while a

common backbone network with convolutional layers is shared between  $f_{\theta_r}(\cdot)$  and  $g_{\theta_c}(\cdot)$ , with two different branches used for implementing these two function, as further described in Section III. The structure of the employed network architecture is depicted in Fig. 1.

### III. EXPERIMENTAL EVALUATION

The experimental evaluation is provided in this Section. First, the employed datasets, experimental setup and neural network architectures are presented. Then, the proposed method is extensively evaluated and discussed.

#### A. Dataset and Experimental Setup

The proposed method was evaluated using the Head Pose Image Dataset (HPID) [23], which contains facial images of several persons at various pans and tilts, ranging from  $-90^\circ$  to  $90^\circ$ . The HPID dataset was selected for evaluating the proposed method, since it contains real images at various angles (instead of simulated faces), as also depicted in Fig. 2, and it provides the full range of pans (from  $-90^\circ$  to  $90^\circ$  with  $15^\circ$  steps). The small number of identities and face images per pose contained in the dataset renders this evaluation setup especially challenging, since it corresponds to a realistic few-shot learning scenario, which is often encountered in various robotics applications [24].

The 75% of the persons contained in the dataset was used to train the models, while the remaining 25% persons were used for evaluating the trained models. All experiments were conducted 5 times and the mean and standard deviation of the recognition accuracy is reported. Two evaluation setups were used to examine the performance of the proposed method under two different settings: a) ‘‘Set 1’’, where the recognition database contains face images with pans between  $-15^\circ$  to  $15^\circ$ , and a) ‘‘Set 2’’, where face images with pans between  $30^\circ$  to  $60^\circ$  were used. Images with tilt between  $-30^\circ$  and  $30^\circ$  were used for all the conducted experiments.

The backbone network used for the conducted experiments consists of four  $3 \times 3$  convolutional layers with 8, 16, 32 and 64 filters, respectively. The ReLU activation function was used for all the convolutions layer [25], while  $2 \times 2$  max pooling was employed after each layer. A fully connected layer with 256 neurons follows the last convolutional layer of the backbone. The feature extraction branch consists of a fully connected layer with 64 neurons, while the controller branch is composed of a fully connected layer with 32 hidden neurons and a final layer with 3 neurons (one for each action). Images of  $88 \times 88$  pixels were fed to the network, while the Adam optimizer was employed for the optimization [26]. The proposed method was pre-trained on the training dataset for 100 epochs (by training only the feature extraction branch), followed by 50 training epochs, where both branches were simultaneously optimized. Finally, note that the cross entropy loss was weighted with class weights, which were equal to 1 for the left and right actions and to 0.01 for the stay action, since the model tended to more frequently select this action.

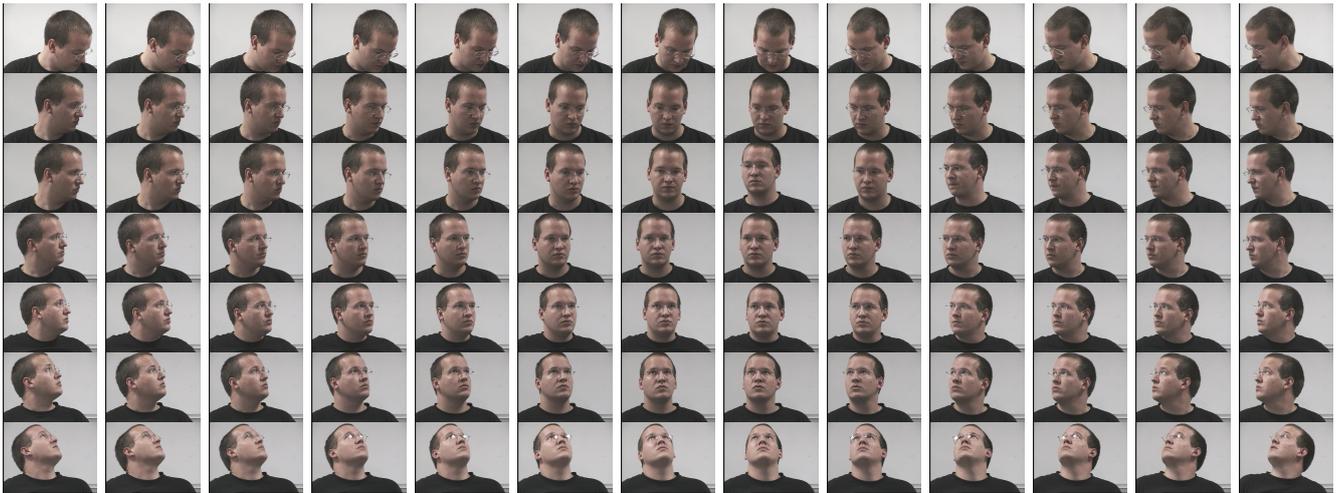


Fig. 2: Sample images contained in the HPID dataset. Note that a complete set of poses are included in the dataset, allowing for evaluating the proposed method using real data instead of using face images generated from simulators.

TABLE I: Experimental Evaluation using the HPID dataset

Method	Accuracy (Set 1)	Accuracy (Set 2)
Static Perception	54.1 ± 3.4%	49.9 ± 4.1%
Static Perception (finet.)	52.7 ± 4.2%	51.4 ± 4.6%
Proposed (1 step)	61.6 ± 5.1%	58.8 ± 7.0%
Proposed (3 steps)	<b>62.2 ± 5.9%</b>	<b>58.9 ± 6.5%</b>

TABLE II: Evaluating the effect of model size on face recognition accuracy

Method	Network	Accuracy	# Param.
Static Perception	0.25×	38.8 ± 6.6%	12k
Static Perception	0.5×	49.0 ± 5.5%	47k
Static Perception	1×	54.1 ± 3.4%	189k
Proposed	0.25×	<b>57.5 ± 5.8%</b>	14k
Proposed	0.5×	<b>60.2 ± 6.9%</b>	52k
Proposed	1×	<b>62.2 ± 5.9%</b>	197k

### B. Experimental Evaluation

The evaluation results are presented in Table I. The proposed method is compared to a static perception approach (“Static Perception”), where the exact same DL model is used, but without employing the control branch. This baseline was trained for 100 epochs. To ensure a fair comparison with the proposed method, we also report evaluation results for the same model, further finetuned for 50 additional training epochs. The proposed method manages to increase the recognition accuracy by more than 7%, just after one control step, which can lead in a view change of at most 15°. This demonstrates the effectiveness of the proposed active perception approach and indicates the importance of exploiting the ability of a robotic system to interact with the environment in order to acquire a better and more discriminative view for the task at hand, confirming the first hypothesis posed

in Section I. It is worth noting that significant improvements are obtained regardless the type of images contained in the database (Setup 1 and Setup 2). When allowed to perform additional control steps (3 steps instead of just 1), the proposed method again further increases the recognition accuracy for both setups. These results suggest that using more fine-grained control approaches, that can further adjust the control steps according the current state, can potentially lead to even greater accuracy improvements.

To evaluate the second hypothesis, i.e., that using active perception allows for using smaller and faster models with only a small impact on the final recognition accuracy, an additional set of experiments was conducted. The experimental results are reported in Table II. Two different networks were employed to this end. The same architecture as in the previous experiment is used, but the number of neurons per layer is reduced by 0.5× and 0.25× (respectively) compared to the original architecture. Indeed, active perception can exceed the performance of traditional static perception models, using one order of magnitude less parameters, leading to faster and more accurate models.

## IV. CONCLUSIONS

In this work, we presented a DL method for face recognition that utilizes active perception. The proposed method employs a hybrid architecture with two output branches, allowing for simultaneously learning discriminative embeddings, as well as providing the appropriate high-level control commands. As it was experimentally demonstrated, the proposed method indeed allows for improving the face recognition accuracy, while, at the same time, allows for using simpler and more efficient models to perform face recognition, leveraging the ability of the system to acquire a view that is more suitable for the task at hand. Finally, it is worth noting that the proposed method

is generic and task-agnostic and it provides a straightforward way of enabling active-perception capabilities for many of the existing static perception DL models.

The proposed method paves the way for developing generic active perception approaches for a wide variety of tasks, since it provides an efficient task-agnostic way for training DL methods that can actively interact with the environment to obtain a more discriminative view. The most important obstacle for developing such approaches is the lack of appropriate datasets, needed for obtaining realistic views of the environments. Apart from collecting such datasets, which is an expensive and tedious task, several interesting alternatives exist. For example, Generative Adversarial Networks can be used to perform image-to-image translation to obtain different views for existing datasets [27], [28], 3D models can be created from real images and the various views can be directly rendered [29], both real and simulated data can be combined using highly realistic simulations [30], [31], while knowledge distillation methods could be used to further reduce the gap between real and simulated data [32], [33].

## REFERENCES

- [1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [2] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.
- [3] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.
- [4] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao, "A discriminative feature learning approach for deep face recognition," in *Proc. European Conference on Computer Vision*, 2016, pp. 499–515.
- [5] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Praseoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al., "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.
- [6] Nikolaos Passalis, Anastasios Tefas, and Ioannis Pitas, "Efficient camera control using 2d visual information for unmanned aerial vehicle-based cinematography," in *Proc. IEEE International Symposium on Circuits and Systems*, 2018, pp. 1–5.
- [7] Nikolaos Passalis and Anastasios Tefas, "Deep reinforcement learning for controlling frontal person close-up shooting," *Neurocomputing*, vol. 335, pp. 37–47, 2019.
- [8] Quan Liu, Zhihao Liu, Wenjun Xu, Quan Tang, Zude Zhou, and Duc Truong Pham, "Human-robot collaboration in disassembly for sustainable manufacturing," *International Journal of Production Research*, vol. 57, no. 12, pp. 4027–4044, 2019.
- [9] Jian-hua Li, "Cyber security meets artificial intelligence: a survey," *Frontiers of Information Technology & Electronic Engineering*, vol. 19, no. 12, pp. 1462–1474, 2018.
- [10] George Loukas, Tuan Vuong, Ryan Heartfield, Georgia Sakellari, Yonpil Yoon, and Diane Gan, "Cloud-based cyber-physical intrusion detection for vehicles using deep learning," *IEEE Access*, vol. 6, pp. 3491–3508, 2017.
- [11] Yiannis Aloimonos, *Active perception*, Psychology Press, 2013.
- [12] Ruzena Bajcsy, Yiannis Aloimonos, and John K Tsotsos, "Revisiting active perception," *Autonomous Robots*, vol. 42, no. 2, pp. 177–196, 2018.
- [13] Macheng Shen and Jonathan P How, "Active perception in adversarial scenarios using maximum entropy deep reinforcement learning," in *Proc. International Conference on Robotics and Automation*. IEEE, 2019, pp. 3384–3390.
- [14] Rickye S Heffner and Henry E Heffner, "Evolution of sound localization in mammals," in *The evolutionary biology of hearing*, pp. 691–715, 1992.
- [15] Phil Ammirato, Patrick Poirson, Eunbyung Park, Jana Kořecká, and Alexander C Berg, "A dataset for developing and benchmarking active vision," in *Proc. IEEE International Conference on Robotics and Automation*, 2017, pp. 1378–1385.
- [16] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese, "Gibson env: Real-world perception for embodied agents," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9068–9079.
- [17] Xiaoning Han, Huaping Liu, Fuchun Sun, and Xinyu Zhang, "Active object detection with multistep action prediction using deep q-network," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3723–3731, 2019.
- [18] Santhosh K Ramakrishnan and Kristen Grauman, "Sidekick policy learning for active visual exploration," in *Proc. European Conference on Computer Vision*, 2018, pp. 413–430.
- [19] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 212–220.
- [20] Raia Hadsell, Sumit Chopra, and Yann LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, vol. 2, pp. 1735–1742.
- [21] Zheng Lian, Ya Li, Jianhua Tao, and Jian Huang, "Speech emotion recognition via contrastive loss under siamese networks," in *Proc. Joint Workshop of the 4th Workshop on Affective Social Multimedia Computing and First Multi-Modal Affective Computing of Large-Scale Multimedia Data*, 2018, pp. 21–26.
- [22] Connor Schenck and Dieter Fox, "Visual closed-loop control for pouring liquids," in *Proc. IEEE International Conference on Robotics and Automation*. IEEE, 2017, pp. 2629–2636.
- [23] Nicolas Gourier, Daniela Hall, and James L Crowley, "Estimating face orientation from robust detection of salient facial features," in *Proc. ICPR International Workshop on Visual Observation of Deictic Gestures*, 2004.
- [24] Jake Snell, Kevin Swersky, and Richard Zemel, "Prototypical networks for few-shot learning," in *Proc. Advances in Neural Information Processing Systems*, 2017, pp. 4077–4087.
- [25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.
- [26] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
- [27] Rameen Abdal, Yipeng Qin, and Peter Wonka, "Image2stylegan: How to embed images into the stylegan latent space?," in *Proc. IEEE International Conference on Computer Vision*, 2019, pp. 4432–4441.
- [28] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha, "Stargan v2: Diverse image synthesis for multiple domains," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2020.
- [29] Alexandros Lattas, Stylianos Moschoglou, Baris Gecer, Stylianos Ploumpis, Vasileios Triantafyllou, Abhijeet Ghosh, and Stefanos Zafeiriou, "Avatarme: Realistically renderable 3d facial reconstruction in-the-wild," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2020.
- [30] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and service robotics*. Springer, 2018, pp. 621–635.
- [31] Olivier Michel, "Cyberbotics Ltd. webots™: professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, pp. 5, 2004.
- [32] Nikolaos Passalis and Anastasios Tefas, "Learning deep representations with probabilistic knowledge transfer," in *Proc. European Conference on Computer Vision (ECCV)*, 2018, pp. 268–284.
- [33] Nikolaos Passalis, Maria Tzelepi, and Anastasios Tefas, "Heterogeneous knowledge distillation using information flow modeling," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2339–2348.

## **7.4 Leveraging Deep Reinforcement Learning for Active Shooting Under Open-world Setting**

The appended paper follows.

# LEVERAGING DEEP REINFORCEMENT LEARNING FOR ACTIVE SHOOTING UNDER OPEN-WORLD SETTING

A. Tzimas, N. Passalis, and A. Tefas

Department of Informatics, Aristotle University of Thessaloniki, Greece  
E-mail: {tzimasak, passalis, tefas}@csd.auth.gr

## ABSTRACT

Recent advances in Deep Reinforcement Learning (DRL) led to the development of powerful agents that can learn how to perform complicated tasks in an end-to-end fashion operating directly on raw unstructured data, e.g., images. However, the real world performance of such methods critically relies on the quality of the simulation environments used for training them. The main contribution of this paper is the development of a realistic simulation environment, by employing a state-of-the-art graphics engine, for training DRL agents that are able to control a drone for performing active shooting. In contrast with previous approaches, that solely relied on simplistic constrained datasets, the environment employed in this work supports a challenging open-world setting, providing a solid step towards developing effective RL methods for various drone control tasks. An appropriate reward shaping approach is also introduced in this work, ensuring that the agent will behave as expected, avoiding erratic movements, as demonstrated through the conducted experiments.

*Index Terms*— Active Shooting, Deep Reinforcement Learning, Open-World Setting

## 1. INTRODUCTION

Deep Learning (DL) provided powerful information analysis tools that allowed for developing a wide range of intelligent control methods [1, 2, 3, 4]. These applications range from developing autonomous cars [2, 3], to drones that can autonomously perform various challenging tasks, such following trails in forests [4], assisting rescue operations [5], and covering various public events [6]. Early approaches, e.g., [4], relied on DL for performing the visual information analysis, the output of which was used by traditional control approaches, e.g., proportional-integral-derivative (PID) controllers [7, 8], to perform the task at hand. These approaches managed to solve a wide range of difficult problems that required performing information analysis from highly unstructured data, e.g., detecting and/or tracking the objects of interest in an image [9, 10], by employing powerful DL-based analysis methods.



**Fig. 1:** Open-world simulation environment developed in this paper

However, these approaches suffer from a significant drawback: information analysis is performed independently from control. That is, we typically first design a DL method for extracting some handcrafted high-level features, e.g., bounding boxes, that are then used by the control algorithm to perform the task at hand. This pipeline allows for easily developing control algorithms that leverage the power of deep learning for performing complex tasks. However, it is worth noting that such controllers are usually not optimal, since optimal control often requires anticipating scenarios that depend on higher level information that is not extracted from the data. For example, a drone covering an athletic event should slow down when following a runner that approaches the finish line, even if the runner accelerates towards reaching the end of the race. So, even though the bounding box of an athlete is usually enough for accurately following her/him through the race, it discards useful information that could be used to improve the control quality. These limitations can be addressed either by manually creating handcrafted features that can be used to exhaustively handle such scenarios, or by using deep reinforcement learning methods (DRL) that are capable of learning how to perform optimal control directly using the raw data and without requiring to design handcrafted intermediate features [11]. In this way, DRL allows for merging the information analysis and control processes into one unified model that can be trained in an end-to-end fashion towards achieving the desired task.

Indeed, DRL is capable of providing agents that operate directly on the raw input, e.g., pixels in the case of visual information, and learning how to perform various tasks, ranging from maneuvering vehicles [1], to using tools and cooperating to achieve their goals in complex environments [12]. However, DRL agents, in contrast with “traditional” DL models, cannot be trained using static datasets. Instead, they typically require using interactive simulation environments, through which the agents acquire experience and adapt their behavior in order to learn how to achieve their goals. That is, the agents interact with the environment, by performing various actions, and they collect a reward for each of them. Then, they are optimized, according to the collected experience, in order to maximize the expected reward. The quality of the learned agent critically depends on the quality of the simulation environment used for training. This is even more important in robotics-related applications, in which the agents are then deployed in the real world. Therefore, usually either realistic simulations environments are employed, with the hope that the distribution shift induced by transferring the agent from the simulation to the real world will not greatly impact its performance, or *sim2real* learning methods are used to smooth the transition to the real world [13].

In this work, we study the problem of controlling a drone that carries a camera in order to perform frontal view shooting of human subjects. This problem has been tackled with various approaches in the past, ranging from using face detectors and PID controllers [14], to developing DRL approaches for end-to-end control [15, 16], due to its importance for various emerging applications, such as active perception, human-robot interaction, aerial cinematography, etc. It is worth noting that applying DRL for this task is not straightforward, due to the lack of appropriate simulation environments. Existing approaches usually just employ static datasets that have been extended to support DRL, usually offering a very limited variety of view poses (since they are based on a limited collection of static images) [15], raising significant concerns on the behavior of DRL agents on less constrained environments.

The main contribution of this paper is the development of a realistic simulation environment, by employing a state-of-the-art graphics engine, that will allow for training DRL agents under more challenging open-world scenarios. A snapshot of the developed simulation environment is shown in Fig. 1. Note that in this study we employ a significantly more complex and challenging environment compared to previous approaches, e.g., [15], in order to examine the behavior of DRL methods in an open-world setting, allowing to better understand how DRL methods behave and bringing us one step close to deploying such approaches on real applications. It is worth noting that, to the best of our knowledge, this is the first study in which an open-world simulator is employed for training DRL agents for performing frontal view shooting without using a static posed dataset, demonstrating that DRL can be successfully used, even under this challenging

setting. The developed DRL agent is trained using an established value-based DRL method and employs a specially designed reward shaping approach that was critical for successfully training the developed model. Note that reward shaping is often employed in various challenging RL problems, especially when they involve delayed and sparse rewards, to increase the performance of the agents [17]. An open-source implementation of the developed simulator is provided at <https://github.com/opendr-eu> to further accelerate research on developing DRL approaches.

The rest of the paper is structured as follows. First, the developed simulation environment, along with the proposed control agent and reward shaping approach are described in Section 2. Then, the experimental evaluation is provided in Section 3. Finally, conclusions are drawn in Section 4.

## 2. PROPOSED METHOD

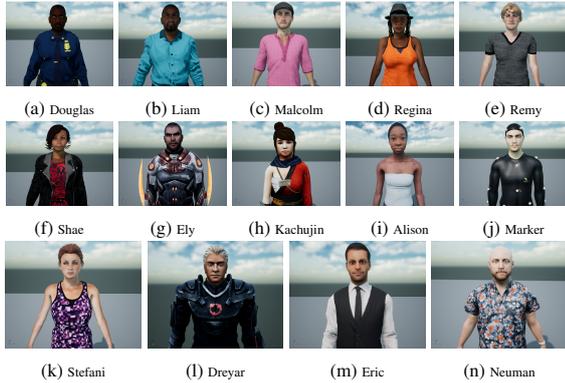
In this Section we describe the developed simulation environment and the capabilities of the employed agent, e.g., available actions, ways of observing and interacting with the environment, etc. Then, we provide a concise description of the employed DRL method, as well as of the proposed reward shaping approach used for training the agent.

### 2.1. Simulation Environment

The simulation environment was developed using the Unreal Engine 4. The environment represents a city that consists of four blocks, as already shown in Fig. 1. Other than the four buildings, bus stops, trees and other smaller props are also included. Furthermore, fourteen distinct 3D models of people were employed and inserted at various spots of the city. The selected models correspond to people of various ethnic groups, with a variety of clothes and features, allowing to examine the behavior of DRL methods under these more challenging conditions. Ten human 3D models were used for training, while the rest of them were used for evaluating the performance of the agent on previously unseen subjects, as shown in Fig. 2.

A drone, equipped with an RGB camera, is also included in the simulation. The purpose of the developed agent is to appropriately control the drone in order to acquire frontal shots of the human models. Note that the drone can perform actions that only affect the position and orientation of the drone, while the orientation of the camera remains fixed, i.e., the agent does not control the gimbal on which the camera is mounted. Therefore, in order to acquire the desired shots, the agent must learn how to appropriately control the drone. The developed environment supports 9 different discrete actions:

- 1) *Forwards*: Move the drone towards the drone’s look direction with speed  $0.3m/s$  for  $0.3s$ ,



**Fig. 2:** Human models used in the developed simulation environment. The first ten (a-j) were used for the training process, while the rest of them (k-n) were used for evaluating the performance of the model. Note the large variations in style, clothing, and features exhibited by the employed human models.

- 2) *Backwards*: Move the drone in the opposite direction than the drone’s look direction with speed  $0.3m/s$  for  $0.3s$ ,
- 3) *Left*: Move the drone to the left with respect to the drone’s look direction with speed  $0.3m/s$  for  $0.3s$ ,
- 4) *Right*: Move the drone to the right with respect to the drone’s look direction with speed  $0.3m/s$  for  $0.3s$ ,
- 5) *Up*: Move the drone up with speed  $0.3m/s$  for  $0.3s$ ,
- 6) *Down*: Move the drone down with speed  $0.15m/s$  for  $0.3s$ ,
- 7) *Rotate left*: Rotate the drone to the left with angular velocity  $10^\circ/s$  for  $0.3s$ ,
- 8) *Rotate right*: Rotate the drone to the right with angular velocity  $10^\circ/s$  for  $0.3s$ ,
- 9) *Stay*: Do not perform any action.

The above actions were implemented using the API provided by the AirSim plugin for Unreal Engine [18]. Also, note that the simulation was running 50 times faster than real time, so each second was equivalent to approximately 10 environment steps. Each episode lasts for 20 seconds, so about 200 steps are performed in each episode.

For every episode, one of the human models is selected at random, which will be the target human for the current episode. The selected human model is rotated to a random orientation so that the background appearing in the shot is different in each episode. The drone is placed in a random position in front of the human model’s face, pointing at it. Some indicative examples of initial positions are shown in Fig. 3. Note again the wide variety of different backgrounds, compared to the uniform background observed by agents used



**Fig. 3:** Observations provided to the agent when an episode begins. A  $200 \times 200$  RGB frame is captured through the camera mounted on the drone at each simulation step.

**Table 1:** Neural network architecture used by the DRL agent

Layer	Output Shape
Input Layer	$200 \times 200 \times 3$
Convolutional Layer ( $8 \times 8$ , stride 4)	$49 \times 49 \times 32$
Convolutional Layer ( $4 \times 4$ , stride 2)	$24 \times 24 \times 64$
Convolutional Layer ( $3 \times 3$ , stride 1)	$22 \times 22 \times 64$
Fully Connected Layer	512
Fully Connected Layer	9

in previous works (as shown in Fig. 1). The episode ends when either the human’s face goes out of the frame, or the drone moves away from the target, or the drone collides with another object or finally when the 20 second time period is depleted from the start of the episode. The RL agent interacts with the developed environment and observes its state through the camera mounted on the drone, which provides one  $200 \times 200$  RGB frame to the agent at each step. The agent is then trained to learn how to appropriately control the drone directly from this raw pixel input.

## 2.2. Deep Reinforcement Learning Agent

At each time step the agent observes a tensor  $\mathbf{x} \in \mathbb{R}^{200 \times 200 \times 3}$  and decides which is the most appropriate action that should be selected in order to maximize the reward obtained from the environment. A value-based DRL approach is employed in this work: a deep learning model  $f_{\mathbf{W}}(\mathbf{x}) \in \mathbb{R}^9$  is used to estimate the Q-value for each available action at each time step [19], where the notation  $\mathbf{W}$  is used to denote the trainable parameters of the model. For any given observation, the agent selects the most profitable action by choosing the action with the largest Q-value. A fast and lightweight network architecture, that can also run on embedded processing platforms that drones typically use [20], was employed for this task. The architecture of the used network is shown in Table 1. The ReLU activation function [21] is used for all the convolutional and dense layers (except from the final one, which does not use any activation function and predicts the Q-values for the 9 possible actions). We also followed the dueling approach, proposed in [22], when designing the DRL agent.

A significant limitation of Q-learning is the instability of the learning process when non-linear functions, such as neural networks, are used to approximate the Q-values. To this end, in this work we used prioritized experience replay [23],

to increase the stability of the training process and reduce the correlation between the data used for training the model. The size of the experience replay pool is set to  $N_{replay} = 500$  and batches of  $N_{batch} = 32$  samples are drawn before each gradient descent update. Furthermore, we use a separate target network for generating the Q-values during the training to avoid feedback loops that can lead to instabilities. The target network is updated every  $N_{target} = 500$  steps.

Finally, we used the Huber loss function for training the DRL model to regress the Q-values:

$$\mathcal{L}(\delta) = \begin{cases} \frac{1}{2}\delta^2, & \text{when } \delta < \delta_{thres} \\ |\delta| - \frac{1}{2}\delta_{thres}, & \text{otherwise} \end{cases}, \quad (1)$$

where  $\delta$  is the error between the target Q-value and the current output of the network. The parameter  $\delta_{thres}$  for the Huber loss was set to 1. For updating the weights of the network, the Adam optimizer [24] with learning rate  $\lambda = 0.00005$  was used. Also, the discount factor  $\gamma$  was set to 0.95. Finally, to explore the solution space a linear exploration policy was used. Exploration starts with an initial rate of  $\epsilon_{init} = 1$  and linearly decreases it to  $\epsilon_{target} = 0.1$  during the first  $N_{explore} = 1,900,000$  training steps. After the initial  $N_{explore}$  steps the exploration rate stays constant to  $\epsilon_{target} = 0.1$ . The agent was trained for 2,000,000 steps.

### 2.3. Reward Shaping

Defining a meaningful reward function is critical for training RL agents. The agent is trained to achieve the desired distance with respect to the selected human subject, as well as to acquire a frontal shot. Therefore, the optimization objective should involve both the distance to the target, as well as the angular error with respect to the human subject. The distance error is defined as:

$$d = \sqrt{(fp_x - cp_x)^2 + (fp_y - cp_y)^2 + (fp_z - cp_z)^2}, \quad (2)$$

where  $(fp_x, fp_y, fp_z)$  denotes the optimal position in which the drone should be in order to take a frontal close-up shot and  $(cp_x, cp_y, cp_z)$  is the current camera position. The angular error is similarly defined as:

$$\theta = \arccos\left(\frac{\mathbf{td}^T \mathbf{ld}}{\|\mathbf{td}\|_2 \|\mathbf{ld}\|_2}\right), \quad (3)$$

where  $\mathbf{td} = (td_x, td_y, td_z)$  is the person’s face direction with respect to the drone’s camera and  $\mathbf{ld} = (ld_x, ld_y, ld_z)$  is the current look direction of the drone’s camera. Therefore, the agents should be optimized with the respect to the combined distance and angular error.

However, simultaneously optimizing these two objectives when training an agent that directly operates on the raw camera input can be especially difficult [15]. To this end, we define an appropriate reward shaping approach that a) further

rewards the agent as it gets closer to the target (by providing an additional  $r_p$  reward), b) encourages the agent to select the stay action (by further providing an additional  $r_s > r_p$  reward), c) punishes the drone (by providing a negative reward equal to  $r_n$ ) when collides with the another object or losses the human subject, d) disentangles the angle from the actual returned reward, as far as the agent stays within certain limits. Therefore, the reward function used in this paper is defined as:

$$r = \begin{cases} (1-d) + r_s + r_p, & \text{when } d < d_1 \text{ and } \theta < \theta_1 \text{ and} \\ & \text{the agent selects “Stay” (action} \\ & \text{8)} \\ (1-d) + r_p, & \text{when } d < d_1 \text{ and } \theta < \theta_1 \\ -r_n, & \text{when } d > d_2 \text{ or } \theta > \theta_2 \text{ or the} \\ & \text{drone collides with another ob-} \\ & \text{ject} \\ 1-d, & \text{otherwise} \end{cases}, \quad (4)$$

where for all the conducted experiments we set:  $r_s = 90$ ,  $r_p = 10$  and  $r_n = 100$ . The error bounds for the distance and angle can be set according to the requirements of each application. In this work, we set  $d_1 = 0.1, \theta_1 = 0.1, d_2 = 1.13$  and  $\theta_2 = 0.4$ . Note that providing the additional  $r_s$  reward when the stay action is selected, allows not only to reach the target position and orientation, but also to train more robust agents that stay in place (select the “stay” action) when the desired shot is achieved and avoid erratic movements. In Section 3, we experimentally verified that this approach indeed leads to stabler agents that avoid performing back and forth movements just to slightly reduce the position error.

## 3. EXPERIMENTAL EVALUATION

The evaluation results are reported in Table 2. Two different evaluation setups were used: a) “train”, where the ten human models that were used during the training process were used for the evaluation, and b) “test”, where four different human models were used. For evaluating the method we ran 500 random episodes, where the agent was allowed to perform actions in a 20 second window. The proposed approach (denoted by the acronym “DRL” in Table 2) is also compared to a baseline agent that only selects the stay action. This allows to evaluate the improvements acquired by using the proposed agent. The total reward acquired through each episode is reported, as well the mean distance error (in meters). The proposed agent is indeed able to accurately control the agent to acquire a frontal shot, both for the train evaluation (the error is reduced from 0.60 to 0.15), as well as for the test evaluation (the error is reduced from 0.61 to 0.16). Note that the agent was capable of successfully generalizing the learned policy to subjects not seen during the training, as highlighted by its performance on the test evaluation.

Furthermore, to qualitative demonstrate the ability of the trained agent to control the trajectory of a drone in order to acquire a frontal shot, while avoiding erratic movements, we

**Table 2:** Drone control evaluation for frontal close-up shooting

Method	Eval. Type	Mean Reward	Mean Error
Baseline	Train	140	0.604
DRL	Train	10,500	0.148
Baseline	Test	137	0.605
DRL	Test	6,250	0.164

provide an example of a control sequence in Fig. 4a. It is worth noting that after acquiring a satisfactory frontal shot, the agent selects the “stay” action, in order to acquire the reward associated with this action ( $r_s$ ), demonstrating the effectiveness of the employed reward shaping approach. Also, in Fig. 4b, we also provide another control trajectory from a different viewing angle, showing both the human subject, as well as the drone. Again, similar conclusions can be drawn: the drone is appropriately controlled to lock its position in front of the human subject.

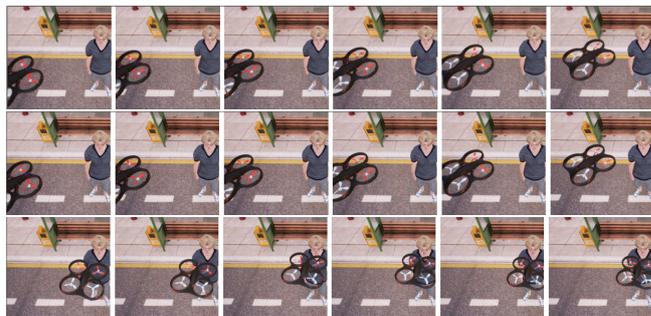
#### 4. CONCLUSIONS

In this paper we presented a realistic simulation environment that can be used for developing deep reinforcement learning methods for various drone-related control tasks. The developed simulation environment goes beyond the static environments used by many existing methods, that largely employed datasets with little to no background variations. This open-world environment, that does not constraint the actions that can be performed, was used to develop an agent capable of controlling a drone, in order to perform frontal view shooting. The effectiveness of the developed approach was demonstrated using both subjects that were seen by the agent during the training, as well novel subjects, which were not seen during the training. The developed environment provides a solid step towards developing effective DRL methods for various robotics-related control tasks, allowing for easily implementing and experimenting with novel DRL approaches. To this end, we also provide an open-source implementation of the developing simulation environment, as well as of the DRL agent, allowing other researchers easily use and extent the methods presented in this paper.

There are several interesting future research directions. First, the proposed method can be used and evaluated in even more challenging settings, in which the human subjects interact with each other and potentially use other objects of the environment, e.g., cars. It is worth noting that this is a scenario that can be easily supported by the developed simulation environment. Furthermore, more advanced network architectures, such as recurrent neural networks, can be employed to ensure that the agents will capture the necessary temporal information needed for anticipating actions from each subject. Finally, the proposed method can be extended to control mul-



(a) View from the drone’s camera



(b) View from a fixed point above the subject (the red rotors represent the front end of the drone)

**Fig. 4:** Two example control sequence of the trained DRL agent from two different perspectives

tiple axes, e.g., both the movement of a drone and the gimbal mounted on it, providing more flexible agents that can achieve shots of higher quality.

**Acknowledgments:** This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR). This publication reflects the authors’ views only. The European Commission is not responsible for any use that may be made of the information it contains.

#### 5. REFERENCES

- [1] Tianhao Zhang, Gregory Kahn, Sergey Levine, and Pieter Abbeel, “Learning deep control policies for autonomous aerial vehicles with mpc-guided policy

- search,” in *IEEE International Conference on Robotics and Automation*, 2016, pp. 528–535.
- [2] Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani, “Deep reinforcement learning framework for autonomous driving,” *Electronic Imaging*, vol. 2017, no. 19, pp. 70–76, 2017.
- [3] Mohammed Al-Qizwini, Iman Barjasteh, Hothaifa Al-Qassab, and Hayder Radha, “Deep learning algorithm for autonomous driving using googlenet,” in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 89–96.
- [4] Nikolai Smolyanskiy, Alexey Kamenev, Jeffrey Smith, and Stan Birchfield, “Toward low-flying autonomous mav trail navigation using deep neural networks for environmental awareness,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 4241–4247.
- [5] A Claesson, D Fredman, L Svensson, M Ringh, J Hollenberg, P Nordberg, M Rosenqvist, T Djarv, S Österberg, J Lennartsson, et al., “Unmanned aerial vehicles (drones) in out-of-hospital-cardiac-arrest,” *Scandinavian journal of trauma, resuscitation and emergency medicine*, vol. 24, no. 1, pp. 124, 2016.
- [6] Tobias Nägeli, Lukas Meier, Alexander Domahidi, Javier Alonso-Mora, and Otmar Hilliges, “Real-time planning for automated multi-view drone cinematography,” *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 132, 2017.
- [7] Nikolaos Passalis, Anastasios Tefas, and Ioannis Pitas, “Efficient camera control using 2d visual information for unmanned aerial vehicle-based cinematography,” in *Proceedings of the IEEE International Symposium on Circuits and Systems*, 2018, pp. 1–5.
- [8] Karl Johan Åström, Tore Hägglund, and Karl J Astrom, *Advanced PID control*, vol. 461, The Instrumentation, Systems, and Automation Society Research Triangle, 2006.
- [9] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.
- [10] Naiyan Wang and Dit-Yan Yeung, “Learning a deep compact image representation for visual tracking,” in *Proceedings of the Advances in Neural Information Processing Systems*, 2013, pp. 809–817.
- [11] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529, 2015.
- [12] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch, “Emergent tool use from multi-agent autocurricula,” *arXiv preprint arXiv:1909.07528*, 2019.
- [13] Fereshteh Sadeghi, Alexander Toshev, Eric Jang, and Sergey Levine, “Sim2real viewpoint invariant visual servoing by recurrent control,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4691–4699.
- [14] Rita Cunha, Miguel Malaca, Vasco Sampaio, Bruno Guerreiro, Paraskevi Nousi, Ioannis Mademlis, Anastasios Tefas, and Ioannis Pitas, “Gimbal control for vision-based target tracking,” in *Proceedings of the European Conference on Signal Processing*, 2019.
- [15] Nikolaos Passalis and Anastasios Tefas, “Deep reinforcement learning for controlling frontal person close-up shooting,” *Neurocomputing*, vol. 335, pp. 37–47, 2019.
- [16] Nikolaos Passalis and Anastasios Tefas, “Continuous drone control using deep reinforcement learning for frontal view person shooting,” *Neural Computing and Applications*, pp. 1–12, 2019.
- [17] Andrew Y Ng, Daishi Harada, and Stuart Russell, “Policy invariance under reward transformations: Theory and application to reward shaping,” in *Proceedings of the International Conference on Machine Learning*, 1999, vol. 99, pp. 278–287.
- [18] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor, “Airsim: High-fidelity visual and physical simulation for autonomous vehicles,” in *Field and Service Robotics*, 2017.
- [19] Hado Van Hasselt, Arthur Guez, and David Silver, “Deep reinforcement learning with double q-learning,” in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
- [20] Nathan Otterness, Ming Yang, Sarah Rust, Eunbyung Park, James H Anderson, F Donelson Smith, Alex Berg, and Shige Wang, “An evaluation of the nvidia tx1 for supporting real-time computer-vision workloads,” in *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium*, 2017, pp. 353–364.
- [21] Xavier Glorot, Antoine Bordes, and Yoshua Bengio, “Deep sparse rectifier neural networks,” in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.
- [22] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas, “Dueling network architectures for deep reinforcement learning,” *arXiv preprint arXiv:1511.06581*, 2015.
- [23] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver, “Prioritized experience replay,” *arXiv preprint arXiv:1511.05952*, 2015.
- [24] Diederik P Kingma and Jimmy Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

## **7.5 Recurrent Bag-of-Features for Visual Information Analysis**

The appended paper follows.

# Recurrent Bag-of-Features for Visual Information Analysis

Marios Krestenitis<sup>a,\*</sup>, Nikolaos Passalis<sup>a</sup>, Alexandros Iosifidis<sup>c</sup>, Moncef Gabbouj<sup>b</sup>, Anastasios Tefas<sup>a</sup>

<sup>a</sup>*Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece*

<sup>b</sup>*Faculty of Information Technology and Communication, Tampere University, Tampere, Finland*

<sup>c</sup>*Electrical and Computer Engineering, Aarhus University, Aarhus, Denmark*

---

## Abstract

Deep Learning (DL) has provided powerful tools for visual information analysis. For example, Convolutional Neural Networks (CNNs) are excelling in complex and challenging image analysis tasks by extracting meaningful feature vectors with high discriminative power. However, these powerful feature vectors are crushed through the pooling layers of the network, that usually implement the pooling operation in a less sophisticated manner. This can lead to significant information loss, especially in cases where the informative content of the data is sequentially distributed over the spatial or temporal dimension, e.g., videos, which often require extracting fine-grained temporal information. A novel stateful recurrent pooling approach, that can overcome the aforementioned limitations, is proposed in this paper. The proposed method is inspired by the well-known Bag-of-Features (BoF) model, but employs a stateful trainable recurrent quantizer, instead of plain static quantization, allowing for efficiently processing sequential data and encoding both their temporal, as well as their spatial aspects. The effectiveness of the proposed Recurrent BoF model to enclose spatio-temporal information compared to other competitive methods is demonstrated using six different datasets and two different tasks.

*Keywords:* Bag-of-Features, Recurrent Neural Networks, Pooling Operators, Activity Recognition

---

\*Corresponding Author

*Email addresses:* mikreste@csd.auth.gr (Marios Krestenitis), passalis@csd.auth.gr (Nikolaos Passalis), alexandros.iosifidis@eng.au.dk (Alexandros Iosifidis), moncef.gabbouj@tuni.fi (Moncef Gabbouj), tefas@csd.auth.gr (Anastasios Tefas)

## 1. Introduction

A vast variety of visual information analysis techniques has been proposed in the field of computer and robotics vision, as one of the most active and continuously expanding research fields. The pipeline of a typical visual information analysis method consists of two fundamental information processing steps: a) feature extraction, which typically extracts lower level information from small spatial or temporal segments of the data, and b) feature aggregation, which fuses the information extracted during the previous step into a compact representation that can be used for various tasks, e.g., classification, retrieval, etc. For example, handcrafted feature vectors can be used to describe local image regions [1], while the set of the extracted features can be then aggregated into a more compact representation using feature aggregation methods, such as the Bag-of-Feature model [2] and Vectors of Locally Aggregated Descriptors (VLAD) [3].

With the advent of deep learning (DL) these two steps were, to some extent, unified and replaced with deep trainable feature extraction layers, e.g., convolutional layers, that are combined with naive pooling operators, e.g., max or average pooling, to lower the complexity of the model and provide translation invariance. Indeed, the outstanding performance of Convolutional Neural Networks (CNNs) in complex and challenging image analysis tasks, has confirmed their ability to extract meaningful feature vectors with high discriminative power for a wide variety of different computer vision tasks [1]. However, these powerful feature vectors are crushed through the pooling layers of the network, that usually implement the pooling operation in a less sophisticated manner, often leading to significant information loss, as further discussed in [4, 5]. This is even more evident in cases where the informative content of the data is sequentially distributed over the spatial or temporal dimension, e.g., videos, which often require extracting fine-grained temporal information, which is discarded by these pooling approaches.

The aforementioned limitations can be better understood through the following example. Consider the task of activity recognition in videos, where the action of *standing up* must be distinguished from the action of *sitting down*. By employing a deep CNN, robust feature vectors can be extracted from every video frame or a sequence of them. However, the pooling layers, as the weak point of the network, dull the expressiveness of the extracted feature vectors and produce less discriminative representations by pooling over the time dimension. For example, assume that a sequence of feature vectors are extracted from a given video instance of action *sitting down*. Let that sequence, notated as  $\mathbf{S}_1 = [\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3]$ , be composed of three feature vectors  $\mathbf{a}_1 = [0, 0, 1]^T$ ,  $\mathbf{a}_2 = [0, 1, 0]^T$ ,

and  $\mathbf{a}_3 = [1, 0, 0]^T$ , which are the feature vectors that correspond to the sub-actions *standing above a chair*, *bending knees* and *sitting on a chair*, respectively. Similarly, consider the same feature vector sequence, but in reverse order, i.e.,  $\mathbf{S}_2 = [\mathbf{a}_3, \mathbf{a}_2, \mathbf{a}_1]$ , that represents a video instance of the activity *standing up*. Also, let  $\mathbf{s}_i$  denote the aggregated representation extracted for the  $i$ -th video. Note that when average or max pooling is applied over both sequences, then the same representation is extracted for both videos i.e.,  $\mathbf{s}_1 = \mathbf{s}_2 = [\max_i[\mathbf{a}_i]_1, \max_i[\mathbf{a}_i]_2, \max_i[\mathbf{a}_i]_3]^T = [1, 1, 1]^T$  for max pooling (where the notation  $[\mathbf{x}]_i$  is used to refer to the  $i$ -th element of vector  $\mathbf{x}$ ) or  $\mathbf{s}_1 = \mathbf{s}_2 = \frac{1}{3} \sum_{i=1}^3 \mathbf{a}_i = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]^T$  for average pooling. Therefore, even though the employed CNN was capable of perfectly recognizing the fundamental sub-actions from still frames, the resulting deep model cannot discriminate between the two actions due to the employed pooling layer. Based on this observation, in this work we characterize the average and max pooling layers as *weak pooling* layers, since they are not capable of capturing the fine-grained spatial or temporal interactions between the feature vectors extracted from a given sequence. Note that in other cases, instead of employing a pooling layer, the extracted feature map can be flattened to a vector and fed to the fully connected layer. However, this approach makes it impossible for the network to handle inputs of arbitrary size, while it significantly reduces the invariance of the network to temporal or spatial translations (the features must always arrive at the exact same moment or position).

Before delving into the details of the proposed pooling method that is capable of overcoming the aforementioned limitations, it is worth revisiting a well-known feature aggregation model, the Bag-of-Features (BoF) model [2], also known as Bag-of-Visual-Words (BoVW). BoF has been extensively used in a wide range of machine learning and computer vision problems [6, 7, 8]. Initially inspired by the Bag-of-Words (BoW) model, which was designed for text analysis, BoF creates a constant-length representation of a multimedia object, e.g., video, audio, time-series, etc., by compiling a histogram over its quantized feature vectors. Assuming a set of objects is given, the usual framework of BoF is as follows:

1. At first, a feature extraction procedure takes place, where every input object is translated to a set of feature vectors. This set defines the *features space*, where every object is projected into.
2. A set of representative feature vectors is learned, which is known as *dictionary* or *codebook*. Every feature vector of the codebook is called codeword, while the learning process is called

dictionary learning.

3. Every feature vector of an input object is quantized based on the learned codebook. This step is referred as *feature encoding* or *feature quantization*.

65 4. A histogram is created by counting the number of feature vectors that were quantized to each codeword.

Following this pipeline, every object can be represented as a fixed-length histogram over the learned codewords. The codewords/dictionary can be either learned using generative/reconstruction approaches [2, 9], or by employing discriminative dictionary learning methods [10, 11], which usually  
70 better fit classification tasks. Note that BoF can be also combined with deep neural networks to provide more powerful trainable pooling layers that can better withstand distribution shifts, while handling inputs of various sizes [12].

Despite the remarkable results in various tasks and the ability to handle variable size inputs, the main drawback of BoF-based methods is the loss of spatial and temporal information, i.e. their  
75 inability to capture the geometry of input data [13]. These drawbacks severely limit the ability of BoF to process temporal or sequential data, such as video data, since it is not capable of capturing the temporal succession of events. Therefore, BoF-based methods are still considered as weak pooling layers, since the quantization of feature vectors to codewords does not take into account the features' sequential order and thus the temporal information is entirely discarded. To overcome  
80 this limitation, the quantization process should take into account the order in which the features arrive, allowing for forming temporal codewords that also capture the interrelation between the feature vectors. As a result, a BoF method employing a stateful recurrent quantizer would be able to quantize the vector  $\mathbf{a}_2$  (bending knees) into a different codeword depending on whether it was following the vector  $\mathbf{a}_1$  (standing above a chair) or the vector  $\mathbf{a}_3$  (sitting on a chair), allowing for  
85 extracting a representation that can discriminate between the standing up action from the sitting down action.

In this paper, a novel stateful recurrent pooling approach that can overcome the aforementioned limitations is proposed. The proposed method is inspired by the BoF model, but employs a stateful trainable recurrent quantizer, instead of a plain static quantization approach. In this way, the  
90 proposed method harness the power of a novel powerful recurrent quantization formulation in order to capture the temporal information of input data, which is crucial in classification tasks, such as

activity recognition, while still maintaining all the advantages of the BoF model. The proposed Recurrent BoF (abbreviated as “ReBoF”) layer is used between the last feature extraction layer and the fully connected layer of a network. Therefore, instead of using weak pooling layers, the extracted feature vectors are quantized to a number of codewords in a recurrent manner, enabling to encode the fine-grained temporal information contained in the original feature vectors. The resulting network can be trained in an end-to-end fashion using plain gradient descent and back-propagation, since the proposed ReBoF formulation is fully differentiable. This allows for building powerful deep learning models for various visual information analysis tasks, as thoroughly demonstrated in this paper. Furthermore, the proposed layer enables the network to operate with arbitrary sized inputs, while it can also reduce the size of the fully connected layer by extracting a compact, yet powerful constant length representation. Finally, it is worth noting that ReBoF can be also used for encoding the spatial information, instead of merely the temporal one. For example, the spatial information encoded in the feature vectors extracted from static images can be encoded by manipulating the extracted feature map as a sequence of feature vectors. This allows for overcoming one long-standing limitation of regular BoF formulation that led to the need of using complicated spatial segmentation schemes [13].

The rest of the paper is structured as follows. In Section 2 the related work is presented and compared to the proposed approach. Next, the ReBoF model is elaborately described in Section 3. The experimental evaluation of ReBoF over six different datasets, along with comparisons to other related techniques, are presented in Section 4. Finally, conclusions are drawn in Section 5.

## 2. Related Work

Over the years, Bag-of-Features has been a widely used method for image analysis tasks. Its ability to provide better scale and position invariance of local keypoint features attracted significant research attention [2]. Moreover, its capability to handle variable sized images made BoF a common choice in image classification and retrieval tasks. The remarkable results in these research areas among with its simplicity motivated researchers to employ BoF in more complex tasks, such as video activity recognition and video semantic retrieval [14, 15]. Video-based human activity recognition consists one of the most challenging problems in computer vision community, that gains increasing interest over the years. Early approaches were focused on crafting discriminative features to describe each activity, such as space-time trajectories of human joints [16]. This is not an easy task, since

the extracted low- or mid-level features must be robust to noise and intra-class variations, such as viewpoint and illumination changes, different motion speeds, etc. Moreover, there are human actions that are associated with specific objects or tools, e.g., cooking, hammering, etc., distinct human poses, e.g., sitting, playing guitar, or executed in a distinguishing environment, e.g., surfing, playing tennis. These cases highlight the need for high-level descriptors that are also capable of capturing the high-level semantic information of the video. The proposed ReBoF model goes beyond these methods, since it not only provides an end-to-end trainable model that allows for extracting representations tailored exactly to the task at hand, but it is also capable of modeling the temporal information of the input feature streams.

Previous efforts tried to tackle some of these issues by exploiting the ability of BoF to process objects of various lengths and providing an orderless representation of the input local keypoint features, extracted from RGB streams, optical flow and joint annotations. Feature descriptors, such as HoG/HoF [17], and MBH [14], were used to perform feature extraction. Furthermore, Dollar et al. proposed a feature extractor ensemble [18], which among with other local features, such as Dense Trajectories (DTs), improved Dense Trajectories (iDTs), Space Time Interest Points (STIPs), and skeleton joint and body shape features [19], were combined with the BoF model to better capture the spatio-temporal information of a video. However, these methods employed BoF just as a post-processing step for aggregating the pre-computed features, while the selection of codewords is usually performed in a fully unsupervised way, leading to less discriminative representations and restricting the accuracy of these approaches.

On the other hand, the abrupt expansion of DL in a variety of computer vision problems has also been imported in activity recognition tasks [20, 21]. One of the initial approaches was to use a neural network, which was pretrained on an image classification task, to extract features from each frame and then merge them to classify the whole video [20]. Donahue et al. proposed to feed the features extracted from each frame to a recurrent layer composed of Long Short-Term Memory (LSTM) units [22], allowing for encoding the temporal dimension of the sequence. Later, more powerful methods employed convolutional networks with 3D kernels (C3D) [23, 24] or fused the RGB frame-based features with features from a stack of optical flow frames, through a two-stream network [25], allowing for effectively capturing the motion features of the video. Li et al. [26] proposed a deformable C3D network enhanced with an attention submodule in order to capture the temporal and spatial irregularities of video streams. Current state-of-the-art methods use Inflated

3D Convolutional (I3D) networks [21], that are capable of effectively combining both the RGB and optical flow streams. In spite of the impressive results, all these methods use powerful feature  
155 extractors combined with weak pooling layers.

The limitations of these approaches led to the development of more advanced pooling layers. Earlier works extended traditional feature aggregation methods into differentiable models that can be combined with DL models and trained in an end-to-end fashion, e.g., Bag-of-Features was extended to Neural BoF pooling [12], while Vector of Locally Aggregated Descriptors (VLAD)  
160 was extended into NetVLAD pooling [27]. However, these approaches still suffer from the same limitations, since they fail to capture the temporal information contained in the input feature stream. It is worth noting that even more advanced pooling approaches, such as attention-based pooling [28], which dynamically adjusts the feature aggregation on each temporal segment, and learnable pooling with context gating [29], cannot extract compact representations that can capture  
165 the temporal information. More recently proposed methods attempt to overcome this limitation by further encoding the spatial information either by employing pyramid-based pooling schemes [30], or by further extending the VLAD model to capture spatio-temporal information [31]. However, these approaches either do not employ memory or use VLAD-based features of significantly higher dimensionality compared to those typically used in BoF-based models. On the other hand, the  
170 proposed method is capable of equipping BoF with memory (by employing a context-aware recurrent quantizer), overcoming all these limitations, while keeping the ability of the BoF model to extract compact, yet discriminative, histogram-based representations.

This work aims to bridge the gap between earlier BoF approaches and the recently proposed DL models by employing a powerful recurrent BoF formulation, which can be readily combined with  
175 DL architectures, as well as effectively process sequential data. The proposed method manages to successfully adapt the BoF model to temporal tasks by overcoming the limitations of other recurrent models, e.g., LSTMs and GRUs, that often fail to correctly model the temporal information extracted from videos, as also noted in [29]. It is worth noting that previous methods successfully managed to combine neural networks with BoF, where the codewords are learned via end-to-end  
180 training of the model [12]. However, they were mainly limited to image analysis problems, while suffering from spatial information loss, due to BoF layer architecture. Furthermore, it has been shown that BoF can be modeled as a (stateless) recurrent neural network that simply sums over the quantized features [32]. In contrast to these approaches, the proposed method employs a stateful

recurrent quantizer that exploits the sequential nature of the extracted features. Therefore proposed ReBoF model retains the valuable assets of BoF models, while, at the same time, it can effectively encode the spatio-temporal characteristics of the data. To the best of our knowledge, ReBoF is the first method that provides a stateful recurrent BoF formulation, which can capture the spatio-temporal dynamics of the data and be enclosed in a convolutional network, providing an end-to-end trainable DL model for several visual information analysis tasks.

### 3. Proposed Method

The proposed recurrent BoF model is analytically derived in Section 3.2, after briefly introducing the standard Neural BoF model in Section 3.1. Then, the suggested methodology to apply ReBoF in two use cases, i.e., video and image classification, is described in Section 3.3

#### 3.1. Standard BoF

Assuming that a set  $\mathcal{X} = \{x_i\}_{i=1}^N$  of  $N$  objects is given,  $N_i$  feature vectors are extracted from each object and notated as  $\mathbf{x}_{ij} \in R^D (j = 1, \dots, N_i)$ , where  $D$  is the dimensionality of the feature space. For example, for image classification tasks each of the  $N$  images is represented by feature vectors extracted using a deep convolutional network or hand-crafted feature descriptors [1].

BoF aims to aggregate these extracted feature vectors into a fixed-length histogram. This is achieved using a two-stage procedure, where during the first stage each feature vector is quantized using a predefined number of codewords and then, during the second stage, the  $N_i$  quantized feature vectors of each object are accumulated to form the final histogram. Note that for the first stage, a codebook  $\mathbf{V} \in R^{N_K \times D}$  must be employed, where  $N_K$  is the number of codewords. This codebook is usually learned by clustering all feature vectors into  $N_K$  clusters [2]. Common clustering algorithms, such as  $k$ -means, can be used in this step, with each centroid,  $\mathbf{v}_k \in R^D (k = 1, \dots, N_K)$ , corresponding to a codeword.

Several feature quantization approaches have been proposed [2, 12]. In this work, we focus on a soft quantization approach that allows for learning the codebook using regular back-propagation, along with the rest of the parameters of the model [12]. This can significantly improve the discriminative power of the extracted representation, since the codebook is fined-tuned for the task at hand. The feature vector  $\mathbf{x}_{ij}$ , extracted from the  $i$ -th object, is quantized by measuring its similarity with

each of the  $N_k$  codewords as following:

$$[\mathbf{d}_{ij}]_k = \exp\left(\frac{-\|\mathbf{v}_k - \mathbf{x}_{ij}\|_2}{\sigma}\right) \in [0, 1], \quad (1)$$

where  $\sigma$  controls the fuzziness of the quantization assignment (also known as the *width* of the Gaussian kernel). Then, the fuzzy membership vector of each feature vector is obtained after normalizing the observed similarities:

$$\mathbf{u}_{ij} = \frac{\mathbf{d}_{ij}}{\|\mathbf{d}_{ij}\|_1} \in R^{N_K}. \quad (2)$$

Finally, the normalized membership vectors are accumulated in a histogram for every object:

$$\mathbf{s}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \mathbf{u}_{ij} \in R^{N_K}. \quad (3)$$

Note that the histogram  $\mathbf{s}_i$  has unit  $l_1$  norm, regardless the number of the extracted feature vectors, and provides an aggregated representation of the feature vectors extracted from the corresponding object. Then, the compiled histogram can be fed to a multilayer perceptron (MLP) to classify the object or used in other tasks, such as regression or retrieval.

### 3.2. Recurrent BoF

Note that the representation extracted using the standard BoF formulation described by (1), (2) and (3) completely discards any spatial or temporal information encoded by the input feature vectors. To overcome this limitation, BoF is extended by employing a recurrent stateful quantizer that can capture and effectively encode the temporal information. Note that in case of sequential data, the feature vector  $\mathbf{x}_{ij}$  corresponds to the  $j$ -th timestep of the  $i$ -th sequence. As we will demonstrate in Subsection 3.3, this is without loss of generality, since the same approach can be also used to capture part of the spatial information of the input.

Before deriving a recurrent quantization approach, it is worth revisiting the quantization process involved in the BoF model from a probabilistic perspective. The probability of observing each feature vector  $\mathbf{x}_{ij}$ , given an input object  $x_i$ , can be trivially estimated using Kernel Density Estimation [9] as:

$$p(\mathbf{x}_{ij}|x_i) = \sum_{k=1}^{N_K} [\mathbf{s}_i]_k K(\mathbf{x}_{ij}, \mathbf{v}_k) \in [0, 1], \quad (4)$$

where  $K(\cdot)$  is a kernel function and the histogram (image-specific parameters)  $\mathbf{s}_i \in \mathbb{R}^{N_K}$  separately adjust the density estimation. Then, the histogram can be calculated using a maximum likelihood estimator:

$$\mathbf{s}_i = \arg \max_{\mathbf{s}} \sum_{j=1}^{N_i} \log \left( \sum_{k=1}^{N_K} [\mathbf{s}]_k K(\mathbf{x}_{ij}, \mathbf{v}_k) \right). \quad (5)$$

It can be easily shown [9], that these parameters can be estimated as  $\mathbf{s}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \mathbf{u}_{ij}$ , where

$$[\mathbf{u}_{ij}]_k = \frac{K(\mathbf{x}_{ij}, \mathbf{v}_k)}{\sum_{l=1}^{N_K} K(\mathbf{x}_{ij}^{(t)}, \mathbf{v}_l)} \in [0, 1], \quad (6)$$

giving rise to the regular BoF with soft-assignments, as described in the previous subsection. This formulation allows for replacing the Gaussian kernel used in (1), which requires tuning the width  $\sigma$  of the kernel, with an easier to use hyperbolic kernel that involves no hyper-parameters. The main reason for this choice is that the Gaussian kernel proved to be quite unstable, when employed in a recurrent quantizer. On the other hand, the proposed hyperbolic-based formulation was significantly stabler and easier to use. Therefore, a hyperbolic (sigmoid) kernel is used to compute the similarity between each feature vector and the codewords:

$$[\mathbf{d}_{ij}]_k = \sigma(\mathbf{x}_{ij}^T \mathbf{v}_k) \in \mathbb{R}, \quad (7)$$

where  $\sigma(x) = \frac{1}{1+\exp(-x)}$  is the logistic sigmoid function. Note that this formulation still ignores the temporal information, since it provides no way to encode the current *state* of the quantizer. Therefore, in order to take into account the temporal information the current state, as expressed by the histogram compiled using the feature vectors fed to the network until the current time-step, (7) is extended to:

$$\mathbf{d}_{ij} = \sigma(\mathbf{V}\mathbf{x}_{ij} + \mathbf{V}_h(\mathbf{r}_{ij} \odot \mathbf{s}_{i,j-1})) \in \mathbb{R}^{N_K}, \quad (8)$$

where  $\mathbf{V}_h \in \mathbb{R}^{N_K \times N_K}$  is weight matrix that is used to transfer the gated histogram vector into the quantization space and is learned during the training process,  $\mathbf{s}_{i,j-1}$  is the histogram extracted from previous quantizations (states) and  $\mathbf{r}_{ij} \in \mathbb{R}^{N_K}$  is the output of a reset gate, introduced to ensure the long-term stability of the model. The reset gate is inspired by the GRU model [33] and is defined as:

$$\mathbf{r}_{ij} = \sigma(\mathbf{W}_r \mathbf{x}_{ij} + \mathbf{U}_r \mathbf{s}_{i,j-1}) \in \mathbb{R}^{N_K}, \quad (9)$$

where  $\mathbf{W}_r \in \mathbb{R}^{N_K \times D}$  and  $\mathbf{U}_r \in \mathbb{R}^{N_K \times N_K}$  are the weight matrices used to implement the reset gate.

Then, similarly to standard BoF approach, the  $l_1$  normalized membership vector is computed as in (2):

$$\mathbf{u}_{ij} = \frac{\mathbf{d}_{ij}}{\|\mathbf{d}_{ij}\|_1}. \quad (10)$$

Note that in order to ensure that quantizer's output is always a properly normalized membership vector, the initial state  $\mathbf{s}_{i,0}$  is set equal to  $\mathbf{s}_{i,0} = \frac{1}{N_K} \mathbf{1}$ , where  $N_K$  is the number of recurrent codewords and  $\mathbf{1} \in \mathbb{R}^{N_K}$  is a vector of all ones. Hence, the fixed-length histogram is recurrently updated as follows:

$$\mathbf{s}_{i,j} = (\mathbf{1} - \mathbf{z}_{ij}) \odot \mathbf{s}_{i,j-1} + \mathbf{z}_{ij} \odot \mathbf{u}_{ij} \in \mathbb{R}^{N_K}, \quad (11)$$

where the update gate  $\mathbf{z}_{ij}$  controls how much the current histogram, which also encodes the state of the quantizer, will be updated and it is defined as:

$$\mathbf{z}_{ij} = \sigma(\mathbf{W}_z \mathbf{x}_{ij} + \mathbf{U}_z \mathbf{s}_{i,j-1}) \in \mathbb{R}^{N_K}, \quad (12)$$

where  $\mathbf{W}_z \in \mathbb{R}^{N_K \times D}$  and  $\mathbf{U}_z \in \mathbb{R}^{N_K \times N_K}$  are parameters of the update gate. Finally, the total histogram is compiled by averaging the intermediate state histograms as:

$$\mathbf{s}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \mathbf{s}_{i,j} \in \mathbb{R}^{N_K}. \quad (13)$$

220 All the parameters of the ReBoF model can be learned using regular back-propagation. Note that ReBoF is capable of recursively processing the input feature vectors, while capturing their temporal information. First, the feature vectors are quantized using the proposed recurrent stateful quantizer, as described by (8) and (10). Then, the quantized vectors are used to appropriately update the state of the quantizer, as given by (11), and finally compile the resulting histogram. It is worth  
225 noting that ReBoF, similarly to all BoF-based models, is capable of processing varying-length input sequences.

The pipeline of the proposed ReBoF pooling method is summarized in Fig. 1. The input feature vectors  $\mathbf{x}_{ij}$  are first fed to the reset gate which controls how much the previous histogram  $\mathbf{s}_{i,j-1}$  will contribute to the quantization process. Note that recurrent connections are plotted in red  
230 in Fig. 1. Then, the proposed recurrent quantizer is employed to extract the membership vector  $\mathbf{u}_{ij}$ , which is then fed to the update gate. The update gate controls how much the previous histogram will be updated based on the output of the recurrent quantizer  $\mathbf{u}_{ij}$  and the previous histogram  $\mathbf{s}_{i,j-1}$ , leading to the updated histogram  $\mathbf{s}_{ij}$ . The final histogram representation of an

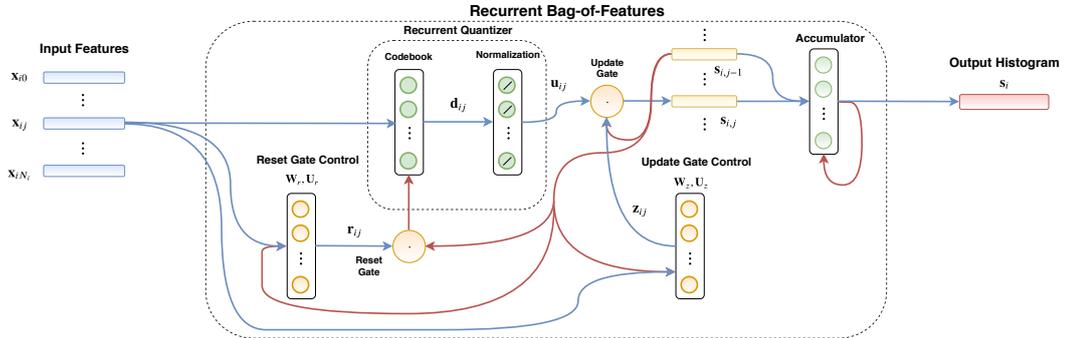


Figure 1: Pipeline of the proposed ReBoF pooling approach. Figure depicts the main information processing steps involved in ReBoF. Red lines denote recurrent connections/information flow, while blue lines denote feed-forward connections/information flow. Figure best viewed in color.

input object is obtained by incrementally accumulating the extracted histogram vectors into the  
 235 final one  $\mathbf{s}_i \in \mathbb{R}^{N_K}$ . It is worth noting that compared to other Neural BoF approaches [12], the  
 proposed one is the first one that employs a recurrent quantizer by incorporating an additional  
 reset gate and update gate. The reset gate is used to equip the quantizer with (resetable) memory,  
 allowing for performing context-aware quantization, while the update gate allows for dynamically  
 240 updating the histogram only when appropriate (e.g., ignoring features that are not relevant to the  
 current context).

### 3.3. Using Recurrent BoF for Visual Information Analysis Tasks

ReBoF can be directly used in any DL architecture without any modification between the last  
 feature extraction layer and the first fully connected one. The extracted representation  $\mathbf{s}_i$  depends  
 only on the number of used codewords  $N_K$  and, as a result, ReBoF is capable of processing inputs  
 245 of variable size without any modification. Three different ways to employ ReBoF in DL models  
 are presented: a) video analysis using CNN frame-based feature extractors, b) image analysis using  
 CNNs and c) spatio-temporal video analysis.

**ReBoF for video analysis:** As presented in 3.2, the advantage of ReBoF is its ability to  
 capture the temporal information of sequential data. This can be readily exploited for tackling  
 250 challenging video analysis problems, e.g., activity recognition, video retrieval, etc. Assuming that  
 every frame of the video is described by an extracted feature vector  $\mathbf{x}_{ij}$  and that the  $i$ -th video  
 is composed of  $N_i$  frames, then ReBoF can be directly applied by feeding to it the sequence of

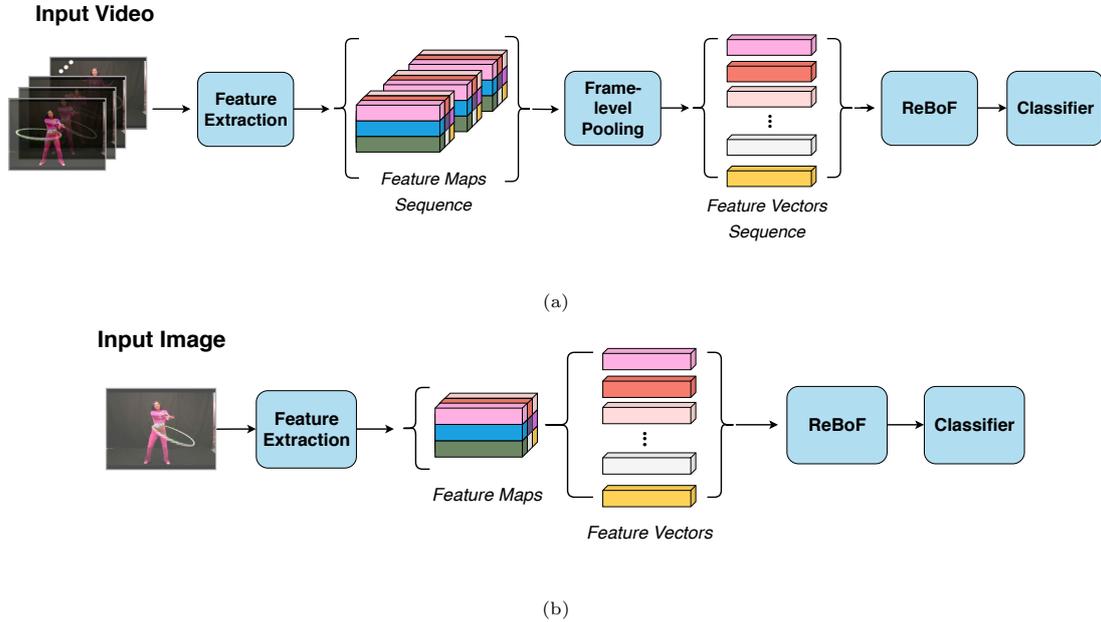


Figure 2: Using Recurrent BoF for two different tasks: (a) video analysis and (b) image analysis

the frame-based feature vectors. This allows ReBoF to process videos of arbitrary duration, while creating fixed-length compact representations of them. This process is illustrated in Fig. 2a, where a feature vector is extracted from every frame of a video. Note that usually a high-dimensional tensor is first extracted from every video frame when convolutional feature extractors are used. This tensor is first aggregated into a feature vector, using a frame-level pooling approach (ReBoF can be also used for this task, as discussed below), and then the feature vectors sequence is fed into the ReBoF model to extract a video-level compact representation. This representation can be then used for the task at hand, e.g., classification.

**ReBoF for image analysis:** Moreover, ReBoF can be utilized also for processing non-sequential input data, while preserving crucial spatial information. For example, in case of image classification, let  $\mathbf{X} \in R^{W \times H \times C}$  be the 3D feature map extracted from a CNN layer, with width, height and number of channels equal to  $W$ ,  $H$  and  $C$  respectively. Several ways to “scan” the image can be employed to flatten the feature map into a set of vectors. In the simplest case considered in this paper, the image is scanned from up to down and left to right. Therefore, a total of  $N_i = W \cdot H$  feature vectors are extracted from each image. Then, this flattened sequence of feature vectors can

be fed into the ReBoF model. This process is illustrated in Fig. 2b. Note that since the feature vectors are always produced following the same scanning pattern, the model can learn the spatial interactions between the extracted feature vectors, effectively capturing the spatial information.

**ReBoF for spatio-temporal analysis:** Finally, it is possible to combine two ReBoFs in order to capture the spatio-temporal content of the input video. Similarly to the image analysis approach, the first ReBoF model can be used to create a histogram for every video frame, i.e., perform the frame-level pooling, as shown in Fig. 2b. Then, a time series of histograms is compiled and fed to the second ReBoF that captures the temporal information of the video (as shown in Fig. 2a) and thus leads to a compact spatio-temporal representation that is subsequently fed to the classifier.

#### 4. Experimental Evaluation

In this section the proposed ReBoF model is evaluated on six different datasets. The experimental procedure along with the results are presented and discussed in the following two main subsections. In the first one, the proposed method is validated over three video datasets for activity recognition, while in the second one, evaluation is conducted over three datasets for image classification.

##### 4.1. Video Activity Recognition

**Datasets:** The proposed ReBoF model is initially validated in 3 different video datasets for activity recognition. The specific task was selected to demonstrate the ability of ReBoF to effectively capture the temporal dimension of video sequences, which is expected to contain essential information for classification tasks.

The first dataset is the UTKinect-Action3D [34] that consists of 10 types of human activities in indoor settings. Samples for each action are collected from 10 different people that perform every activity two times. The demonstrated actions are: *walk*, *sit down*, *stand up*, *pick up*, *carry*, *throw*, *push*, *pull*, *wave hands*, and *clap hands*. For each video instance, three streams were recorded: RGB, depth and skeleton joint locations. The RGB images were used for all of the conducted experiments. Since there is no official training/testing split provided, a 50%-50% subject-based split strategy was employed, i.e., the videos of the first five subjects were included in the training set and the rest of them were used to form the testing set. Hence, a quite challenging setup was created, as the activities belonging in the testing set were performed from unseen subjects.

The second database is the well-known UCF101 dataset[35], that is widely used for benchmarking action recognition models. The dataset contains 13,320 action instances belonging in 101 classes, that can be grouped in five generic categories, namely: 1. Human-Object Interaction, 2. Body-Motion Only, 3. Human-Human Interaction, 4. Playing Musical Instruments, and 5. Sports. UCF101 also contains three different evaluation splits. For all the experiments conducted in this paper, the first split was used.

We also created a challenging and more complex dataset based on the UCF101 to better demonstrate the ability of the proposed method to capture the temporal dynamics of video data. To this end, we compiled a dataset by mixing instances from different activities of the UCF101 dataset together. More specifically, 10 activities of UCF101 (split 1) were selected to be joined together. Every action of this subset was joined with each one of the remaining, leading to 90 complex actions. One can further understand the significance of encoding the temporal information of these instances by considering that a sample of action “A” combined with one of action “B” (let name this complex activity class “AB”) must be separated from samples of complex activities from class “BA”. Note that “AB” and “BA” videos contain the same set of frames (for a specific instance of “A” and “B”), but in a different order. A typical example of such two *complex* actions would be the action sequences of “sitting to a chair” (AB) and “rising from a chair” (BA), where A is the action of getting closer to the chair and B the action of actually being on the chair. A method that does not capture the temporal information cannot discriminate between these two actions, since the videos would contain the same frames, but in the reverse order. Hence, 114 samples were selected for each class, as this was the minimum number of instances contained in the selected initial classes. The selected 10 action classes were the following: *ApplyEyeMakeup*, *ApplyLipstick*, *Billiards*, *BoxingPunchingBag*, *BoxingSpeedBag*, *Haircut*, *Hammering*, *TableTennisShot*, *TennisSwing*, and *Typing*. These classes were selected to offer a challenging dataset of complex activities, where every primary action has familiar content with at least one of the rest. The  $i$ -th sample of initial class “A” is combined with the  $i$ -th sample of initial class “B” and so on, leading to 7,380 training and 2,880 testing data equally balanced among the 90 classes. The compiled dataset is called “Complex UCF” in the rest of this paper.

**Evaluation Setup:** The video analysis approach described in Subsection 3.3 was employed in order to evaluate the proposed method over the first two datasets. Every video instance was uniformly sampled in time in order to extract a predefined number of frames, denoted by  $N_f$ . The number

of extracted frames was set to  $N_f = 30$  for the UTKinect-Action3D dataset and to  $N_f = 40$  for the UCF101 dataset. Shorter videos were looped over as many times as necessary to ensure that each video contains at least  $N_f$  frames, following the methodology described in the relevant literature [24].

Every frame was fed to an Inception V3 model [36], pretrained in ImageNet, and a feature representation was extracted from each frame from the last average pooling layer of the network. Therefore, every frame is represented by a 2048-dimensional feature vector, building a sequence of  $N_f$  features for every video sample. This sequence is then fed to the ReBoF layer, which is followed by a fully connected layer block composed of two fully connected layers. The first one is composed of 512 neurons using the ReLU activation function and dropout with rate of 0.5, while the output (classification) layer has as many neurons as the classes of each dataset and employs the softmax activation function. The cross-entropy loss is used during the training. The resulting network was trained from scratch using the Adam optimizer and a learning rate of  $10^{-5}$ , apart from the pretrained feature extractors which were kept frozen. We also experimentally found out that scaling the extracted histogram by  $N_K$  improved the convergence of the proposed method, especially when training the whole architecture from scratch. This scaling ensures that the gradients from the fully connected layers will not diminish as they are back-propagated to the previous layers. For the UTKinect-Action3D dataset the models were trained for 800 epochs, while for the UCF101, all models were trained for 50 epochs and the training/evaluation procedure was repeated 3 times.

The performance of the proposed method is compared with three other pooling methods, that also use the exact same feature extraction and classification layer blocks (in order to ensure a fair comparison). First, global average pooling over the temporal dimensions of the input sequence is used in place of the proposed ReBoF layer. This method is called “Average Pooling” in the rest of this paper. For the second method, a recently proposed state-of-the-art variant of the BoF method that is adapted for use with DL architectures, the Linear BoF [37], is employed. Furthermore, a powerful recurrent aggregation model, the GRU [33], which is used to aggregate the features, while also capturing the temporal dimension of the data, is also compared to the proposed method. To ensure a fair comparison, we use the same number of codewords or hidden units for the Linear BoF or GRU model, respectively, as the number of codewords used in our method, except otherwise stated.

A slightly different setup was used for the third dataset (Complex UCF101). A 16-frame sliding

360 window, with overlap of 4 frames was applied on every activity instance of UCF101 dataset, creating  
16-frame clips. A 3D ResNeXt-101, pretrained on UCF101, as deployed in [24], was used for feature  
extraction. Every clip was then fed to the network and features were extracted from the last  
average pooling layer of the network. After this step, every video clip is represented by a 2048-  
dimensional vector. Note that contrary to the previous setup, feature vectors already enclose some  
spatio-temporal information, since 3D convolutions were used during the feature extraction. Then,  
365 for every activity instance of class “AB”, 16 sequential feature vectors were selected from each one  
of the classes “A” and “B”, starting from the beginning of the video. If a subsequence of features  
is shorter, then it was looped over. The two subsequences were then stacked, forming a 32-length  
sequence ( $N_f = 32$ ) of feature vectors for the action “AB” of the ComplexUCF dataset.

Similar to the previous setup, the performance of ReBoF is compared to the “Average Pooling”,  
370 ”Linear BoF” and “GRU” methods. All methods share the same classification block as before and  
the networks were trained from scratch, excluding the weights of feature extractors which were  
pretrained. In case of Average Pooling, the network was trained for 50 epochs, while in the rest,  
the training process stopped when 99.9% accuracy was achieved in training set. The evaluation of  
ReBoF and GRU methods is repeated 3 times and the mean accuracy and standard deviation on  
375 the test set are reported.

**Experimental Results:** The experimental evaluation for the UTKinect-Action3D dataset is pro-  
vided in Table 1. The proposed method outperforms the other three evaluated methods, providing  
the highest accuracy (54.64%) using 512 codewords. GRU also achieves its best accuracy (47.71%)  
for the same number of codewords, while for the Linear BoF method the highest performance  
380 (44.47%) is achieved for 256 codewords. Both ReBoF and GRU significantly outperform the Aver-  
age Pooling and Linear BoF methods, since they are capable of effectively modeling the temporal  
dynamics of the input video sequences and distinguishing between similar activities, such as *stand  
up* and *sit down*.

Moreover, in Fig. 3 the effect of using a wider range of codewords and number of GRU units in  
385 the classification accuracy on the UTKinect-Action3D dataset is evaluated using the two methods  
that achieve the highest performance (GRU and ReBoF). In all cases, the proposed method leads  
to higher accuracy compared to the GRU method. Furthermore, the proposed method allows for  
reducing the size of the extracted representation, since it outperforms the best GRU model (512  
units) using just 128 codewords. This allows for reducing the size of the extracted representation

Table 1: UTKinect-Action3D Evaluation

Method	# Codewords - GRU Units	Test Accuracy (%)
Average Pooling	-	40.83
Linear BoF	256	44.47
GRU	512	47.71
ReBoF	512	<b>54.64</b>

and, as a result, the number of parameters in the subsequent fully connected layer. Both methods achieve their best performance for 512-dimensional representations. After this point, the accuracy for both models drops, mainly due to overfitting phenomena.

Similar conclusions can be drawn from the evaluation results on the UCF101 dataset, which are reported in Table 2. The experiments were repeated three times and the mean accuracy and standard deviation on the test set are reported. Even though UCF101 is a less challenging dataset, in terms of temporal dependence, the proposed method still outperforms the rest of the evaluated methods, achieving the highest accuracy of 72.02% for 1024 codewords. The effect of using different number of codewords and recurrent units is also evaluated in Fig. 4. Again, the proposed method outperforms the GRU method regardless the number of used codewords, while it achieves comparable accuracy using 2 to 4 times smaller representations.

Table 2: UCF101 Evaluation

Method	# Codewords - GRU Units	Test Accuracy (%)
Average Pooling	-	70.32 $\pm$ 0.43
Linear BoF	1024	71.11 $\pm$ 0.35
GRU	2048	71.04 $\pm$ 0.20
ReBoF	1024	<b>72.02 <math>\pm</math> 0.68</b>

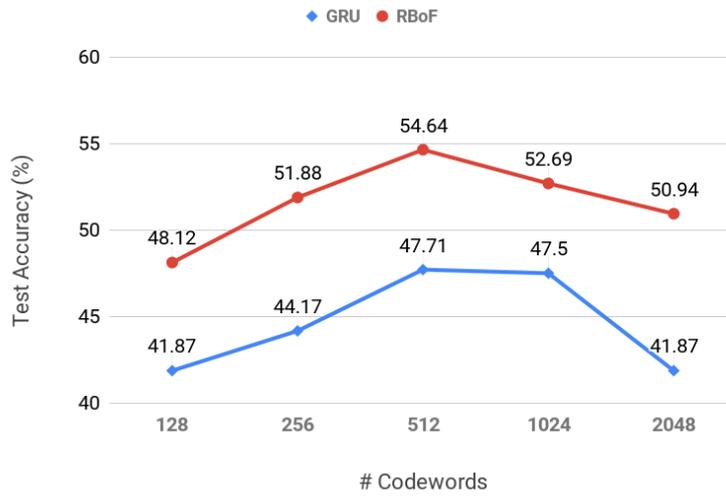


Figure 3: UTKinect-Action3D Evaluation: Effect of using different number of codewords/recurrent units for the ReBoF and GRU methods

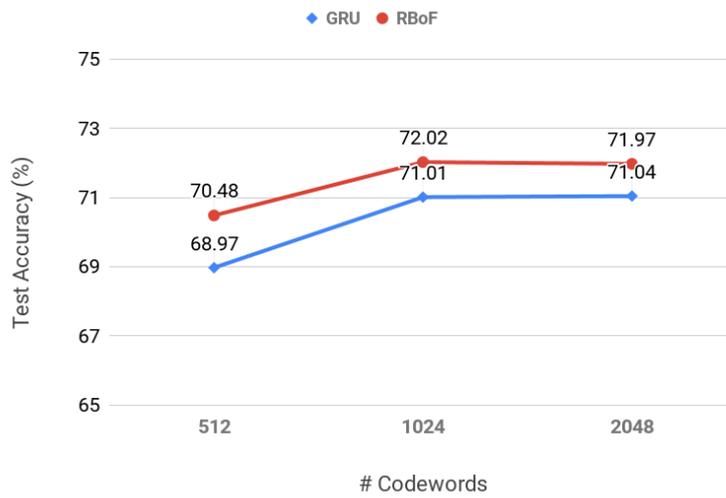


Figure 4: UCF101 Evaluation: Effect of using different number of codewords/recurrent units for the ReBoF and GRU methods

Next, the evaluation on the Complex UCF dataset is provided in Table 3. Note that the Average Pooling and Linear BoF methods fail to overpass the 50% test accuracy barrier, since “AB” activities cannot be distinguished from those of “BA”, due to discarding crucial temporal information. Note that the Average Pooling and Linear BoF methods obtain significantly lower accuracy on this dataset, since capturing the temporal information plays a less crucial role on the other two datasets, in which many videos that belong to certain actions can be categorized just by one frame, leading to a smaller gap between the Average Pooling method and the ones that capture the temporal information, i.e., GRU and ReBoF. Again, ReBoF achieves the highest accuracy equal to 89.29% for 1024 codewords, outperforming the GRU method.

Table 3: Complex UCF101 Evaluation

Method	# Codewords - GRU Units	Test Accuracy (%)
Average Pooling	-	48.95
Linear BoF	1024	43.88
GRU	512	88.86 $\pm$ 2.04
ReBoF	512	89.25 $\pm$ 1.08
GRU	1024	88.62 $\pm$ 1.02
ReBoF	1024	<b>89.29 <math>\pm</math> 0.89</b>

Moreover, qualitative results on Complex UCF101 dataset are presented in Fig. 5 in order to further illustrate the ability of ReBoF to encode efficiently the temporal information of video data. ReBoF is compared with the rest of the methods using the confidence scores for the top- $k$  ( $k = 1, 2, 3$ ) predictions. The corresponding labels are reported for each prediction. Fig. 5 clearly demonstrates the inefficiency of Average Pooling and Linear BoF methods to distinguish the combination of activities “TennisSwing + ApplyLipstick” (“AB”) from the inverted combination of activities “ApplyLipstick + TennisSwing” (“BA”), due to the complete loss of the temporal information. GRU method presents to some extent better performance since the correct predicted label is found on top-2 ranking. Yet, the model is overconfident over the wrong class, probably due to the difficulty of distinguishing the second part (“ApplyLipstick”) of the complex action from a similar one (“ApplyEyeMakeup”). On the other hand, the proposed ReBoF method is more robust

Method	top-1	top-2	top-3
<b>Average Pooling</b>	ApplyLipstick+TennisSwing <b>0.3215</b>	ApplyEyeMakeup+TennisSwing 0.2699	TennisSwing+ApplyEyeMakeup 0.2521
<b>Linear BoF</b>	ApplyLipstick+TennisSwing <b>0.41984</b>	ApplyEyeMakeup+TennisSwing 0.2382	TennisSwing+ApplyEyeMakeup 0.1537
<b>GRU</b>	TennisSwing+ApplyEyeMakeup <b>0.5018</b>	<b>TennisSwing+ApplyLipstick</b> 0.4034	TableTennisShot+ApplyLipstick 0.0344
<b>ReBoF</b>	<b>TennisSwing+ApplyLipstick</b> <b>0.4776</b>	TennisSwing+ApplyEyeMakeup 0.4582	ApplyEyeMakeup+TennisSwing 0.0103

Figure 5: Qualitative results on the Complex UCF101 dataset: For a given video sequence the first three top- $k$  ( $k = 1, 2, 3$ ) prediction scores among with the corresponding predicted label are reported for each of the evaluated methods. In the first column (top-1 results) red and green color is used to denote false and correct predictions, respectively.

to such phenomena, more efficiently capturing the temporal details and managing to accurately classify the input sample, contrary to rest of the methods. It should be also noted that the inverted complex action “ApplyLipstick+TennisSwing” is not present in the first top-3 ranking and thus, the confidence score for the specific class is at least lower than 0.01, signifying the effectiveness of ReBoF’s recurrent quantizer to enclose the temporal characteristics of the video sequence.

Finally, in Table 4 ReBoF is compared to other similar approaches reported in the literature on UCF101 dataset. Activity recognition based on a single RGB frame of the video achieved 67.4% accuracy across all three splits of UCF101. LRCN model presented in [22] improved the results by employing an LSTM to capture the temporal information of the data. The “slow fusion” scheme presented in [20] aimed to conceive the temporal information by applying a CNN network over a time window of the RGB stream, yet the reported results are lower compared to the case of single frame network, failing to exploit the temporal information contained in the video input. On the other hand, the proposed ReBoF formulation manages to effectively capture the spatial information of video streams, outperforming the rest of the evaluated methods. It is worth noting that other approaches that use C3D networks, such as the I3D network [21], can further boost the action recognition performance. However, these more powerful architectures usually also require using

significantly larger datasets in order to be trained effectively and avoid over-fitting, as also demonstrated in Table 4 (e.g., training 3D-ConvNets from scratch leads to lower recognition accuracy), while at the same time these architectures also significantly increase the computational complexity of the model compared to the proposed method.

Table 4: Comparison with other methods on UCF101 (split 1)

Method	Test Accuracy (%)
Single RGB frame (all splits) [22]	67.4
“Slow fusion” spatio-temporal ConvNet (all splits) [20]	65.4
Spatial LRCN (all splits) [22]	68.2
3D-ConvNet (trained from scratch) [33]	51.6
ReBoF	72.02

#### 4.2. Image Classification

**Datasets:** ReBoF is also able to capture the spatial information of the extracted feature vectors, as discussed in Section 3.3. To this end, ReBoF is also evaluated on three image datasets: the MNIST database of handwritten digits (MNIST) [38], the Fashion MNIST database of fashion images [39], and the CIFAR-10 database of color images with varying content [40].

The MNIST dataset is widely used in the computer vision field and contains 60,000 training and 10,000 testing grayscale images of handwritten digits. The size of each image is  $28 \times 28$  pixels and the data are distributed over 10 different classes, one for each digit (0 to 9). The Fashion MNIST is an MNIST-like dataset that contains labeled fashion grayscale images. Similarly to MNIST, it consists of 60,000 training and 10,000 testing images of size  $28 \times 28$  pixels. The CIFAR-10 dataset contains totally 60,000 colour images of size  $32 \times 32$  pixels. The dataset is separated in 50,000 images for training and 10,000 images for testing. Data are equally divided in 10 classes of various generic categories, e.g., *airplane*, *cat*, *ship*, etc.

**Evaluation Setup:** The proposed method is evaluated in the aforementioned image datasets, following the approach presented in Section 3.3. The ReBoF model is combined with a convolutional neural network, creating a model that consists of three main layer blocks: 1. a feature extraction block, 2. a ReBoF layer, and 3. a classification layer block. The feature extraction block is composed

of two convolutional sub-blocks. The first sub-block is composed of two convolutional layers that use  $32\ 3 \times 3$  filters, followed by a  $2 \times 2$  max pooling layer. The second sub-block consists of two additional convolutional layers that use  $64\ 3 \times 3$  filters, followed by a  $2 \times 2$  max pooling layer. In addition, dropout with rate 0.25 is applied to the input of second block. The output of the last convolutional block is used to extract feature vectors for each image. The final feature map is scanned from left to right and from top to bottom to extract feature vectors that are sequentially fed to the ReBoF block, after applying dropout with rate of 0.25. The extracted features are recurrently quantized in  $N_K$ -dimensional vectors and accumulated in a  $N_K$ -length histogram. The output of the ReBoF layer is then fed to the classification layer block that is composed of a hidden fully connected layer with 512 neurons and an output layer with 10 neurons, while dropout with rate of 0.2 and 0.5 is applied to the input of the first and second fully connected layers, respectively. The ReLU activation function is used for all the hidden layers, while the network is trained to minimize the cross-entropy loss. The network is trained from scratch in an end-to-end fashion for 100 epochs using the Adam optimizer and learning rate equal to 0.001.

The proposed method is also compared to using an average pooling baseline, as well as to the Linear BoF method, similarly to the activity recognition task. In the average pooling method, the output of the employed feature extraction block is fed to a global average pooling layer and then forwarded to the employed classification layer block. In this case, no dropout is applied to the feature extractor’s output. In the Linear BoF method, the ReBoF block is substituted by a Linear BoF, while the rest of the network is kept the same to ensure a fair comparison between the methods. Both networks were trained from scratch, using the same training parameters as in the case of ReBoF.

**Evaluation Results:** The proposed method is compared with two other evaluated methods in Table 5. ReBoF outperforms both the Average Pooling and Linear BoF methods in all the evaluated cases, regardless the used dataset. For the MNIST and Fashion MNIST datasets the highest performance is achieved when 512 codewords are used, while for the CIFAR-10 dataset for 128 codewords. Similarly, for the Linear BoF method the best results for the MNIST dataset are achieved when 64 codewords are employed, whereas in the case of Fashion MNIST and CIFAR-10 when 128 codewords are used. The performance improvements are marginal for the simpler MNIST and Fashion MNIST datasets. However, significant accuracy improvements are observed for the more complex CIFAR-10 dataset, confirming the ability of the proposed method to capture

Table 5: Image classification evaluation

Method	MNIST	Fashion MNIST	CIFAR-10
Average Pooling	99.44	92.80	79.71
Linear BoF	99.46	92.97	80.25
ReBoF	<b>99.50</b>	<b>93.28</b>	<b>82.25</b>

(Classification accuracy is reported (%))

the spatial information encoded in the extracted feature vectors. This is also confirmed in the results reported in Fig. 6, where the effect of varying the number of codewords for the ReBoF method is examined. ReBoF achieves remarkable results in the MNIST and Fashion MNIST datasets, even for a small number of codewords. It is worth noting that when a very large number of codewords is used, i.e., 1024 or 2048, the accuracy begins to decrease due to overfitting. The effect of the number of codewords on the model’s accuracy can be also clearly demonstrated for CIFAR-10, which contains larger and more complex images. In this case, the best accuracy is achieved when 128 codewords are used.

Also, the ability of ReBoF method to capture spatial patterns from images is further demonstrated in Fig 7, where qualitative results are presented for the MNIST dataset. In case of Average Pooling method, the prediction model is overconfident over the wrong digit in all of the three evaluated cases. Linear BoF presents a more balanced distribution over the prediction scores, yet it also fails in some cases to classify the input image correctly. On the contrary, ReBoF, which is capable of capturing the spatial information contained in the feature vector sequence extracted from each MNIST digits, leads to the best results, accurately identifying each digit, while also providing the highest confidence on the correct label.

## 5. Conclusions

In this paper a novel Recurrent Bag-of-Features formulation has been proposed, designed to effectively process sequential input data. ReBoF was inspired by the BoF model, but employs a stateful trainable recurrent quantizer, instead of a plain static quantization approach. This enables ReBoF to harness the power of a powerful recurrent quantizer that is able to capture the temporal

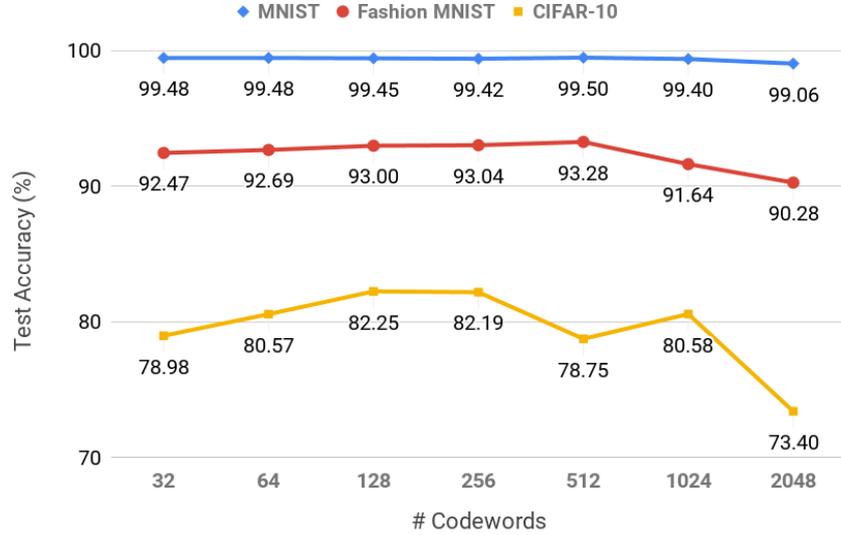


Figure 6: Test accuracy (%) of ReBoF in three datasets for different number of codewords

Input	Average Pooling	Linear BoF	ReBoF	Ranking
	8: <b>0.8842</b>	8: <b>0.5824</b>	<b>2: 0.9848</b>	top-1
	<b>2:</b> 0.0654	<b>2:</b> 0.4172	8: 0.0134	top-2
	9: 0.0483	9: 9.4e-05	0: 0.0008	top-3
	<b>2:</b> <b>0.7978</b>	<b>2:</b> <b>0.5845</b>	<b>7: 0.5897</b>	top-1
	<b>7:</b> 0.2014	<b>7:</b> 0.4151	2: 0.4097	top-2
	3: 0.0003	0: 0.0001	3: 0.0003	top-3
	<b>4:</b> <b>0.7532</b>	<b>9:</b> <b>0.8120</b>	<b>9:</b> <b>0.9273</b>	top-1
	<b>9:</b> 0.2467	4: 0.1879	4: 0.07226	top-2
	8: 5.4e-05	8: 1.8e-05	7: 0.0002	top-3

Figure 7: Qualitative results on the MNIST dataset: Top-k predictions for  $k = 1, 2, 3$  among with the corresponding confidence scores are reported for the three evaluated methods. Correct and false predictions are denoted with green and red color, respectively, in the first row (top-1 predictions) for each sample.

510 information of input data, which is crucial in classification tasks, such as activity recognition, while  
still maintaining all the advantages of the BoF model. ReBoF can be directly used between the last  
feature extraction layer and the fully connected layer of a network, while the resulting architecture  
can be trained in an end-to-end fashion using back-propagation, allowing for building powerful deep  
learning models for various visual information analysis tasks. The performance of ReBoF model was  
515 extensively evaluated in various activity recognition tasks using three video datasets. In all cases,  
the proposed method outperformed the rest of the evaluated methods, demonstrating the ability of  
ReBoF to capture the temporal information and increase the classification accuracy. Furthermore,  
ReBoF is also capable of encoding the spatial information, as it was also demonstrated in image  
classification tasks using three more datasets.

520 ReBoF opens several interesting future research directions. First, the proposed approach for  
activity recognition can be further enhanced by combining a ReBoF layer over the spatial dimen-  
sion, followed by a ReBoF layer over the temporal dimension. In this way, the spatio-temporal  
information can be more properly encoded by creating a space-time histogram, further improving  
the performance over the applications presented in this paper. Second, ReBoF can be employed  
525 to replace the intermediate weak pooling layers of CNNs, aiming to increase the accuracy of the  
models. Finally, employing the proposed method for other tasks, such as video retrieval and hash-  
ing, is expected to boost the performance of existing methods by extracting compact, yet more  
discriminative representations.

## Acknowledgments

530 This project has received funding from the European Union’s Horizon 2020 research and in-  
novation programme under grant agreement No 871449 (OpenDR). This publication reflects the  
authors’ views only. The European Commission is not responsible for any use that may be made  
of the information it contains.

## References

- 535 [1] L. Nanni, S. Ghidoni, S. Brahmam, Handcrafted vs. non-handcrafted features for computer  
vision classification, *Pattern Recognition* 71 (2017) 158–172.

- [2] J. Sivic, A. Zisserman, Video google: A text retrieval approach to object matching in videos, in: Proc. Int. Conf. on Computer Vision, 2003, p. 1470.
- [3] H. Jegou, F. Perronnin, M. Douze, J. Sánchez, P. Perez, C. Schmid, Aggregating local image descriptors into compact codes, *IEEE Trans. on Pattern Analysis and Machine Intelligence* 34 (9) (2012) 1704–1716.
- [4] N. Passalis, A. Tefas, Learning bag-of-features pooling for deep convolutional neural networks, in: Proc. IEEE Int. Conf. on Computer Vision, 2017, pp. 5755–5763.
- [5] C.-Y. Ma, M.-H. Chen, Z. Kira, G. AlRegib, Ts-lstm and temporal-inception: Exploiting spatiotemporal dynamics for activity recognition, *Signal Processing: Image Communication* 71 (2019) 76–87.
- [6] F. B. Silva, R. d. O. Werneck, S. Goldenstein, S. Tabbone, R. d. S. Torres, Graph-based bag-of-words for classification, *Pattern Recognition* 74 (2018) 266–285.
- [7] N. Passalis, A. Tefas, Learning bag-of-embedded-words representations for textual information retrieval, *Pattern Recognition* 81 (2018) 254–267.
- [8] I. F. J. Ghalyan, Estimation of ergodicity limits of bag-of-words modeling for guaranteed stochastic convergence, *Pattern Recognition* 99 (2020) 107094.
- [9] S. Bhattacharya, R. Sukthankar, R. Jin, M. Shah, A probabilistic representation for efficient large scale visual recognition tasks, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2011, pp. 2593–2600.
- [10] A. Iosifidis, A. Tefas, I. Pitas, Discriminant bag of words based representation for human action recognition, *Pattern Recognition Letters* 49 (2014) 185–192.
- [11] A. Iosifidis, A. Tefas, I. Pitas, Multidimensional sequence classification based on fuzzy distances and discriminant analysis, *IEEE Trans. on Knowledge and Data Engineering* 25 (11) (2012) 2564–2575.
- [12] N. Passalis, A. Tefas, Neural bag-of-features learning, *Pattern Recognition* 64 (2017) 277–294.

- [13] S. Lazebnik, C. Schmid, J. Ponce, Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories, in: Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, Vol. 2, 2006, pp. 2169–2178.
- 565 [14] Y.-G. Jiang, C.-W. Ngo, J. Yang, Towards optimal bag-of-features for object categorization and semantic video retrieval, in: Proc. ACM Int. Conf. on Image and Video Retrieval, 2007, pp. 494–501.
- [15] A. Iosifidis, A. Tefas, I. Pitas, View-invariant action recognition based on artificial neural networks, IEEE Trans. on Neural Networks and Learning Systems 23 (3) (2012) 412–424.
- 570 [16] Y. Yacoob, M. J. Black, Parameterized modeling and recognition of activities, Computer Vision and Image Understanding 73 (2) (1999) 232–247.
- [17] I. Laptev, M. Marszałek, C. Schmid, B. Rozenfeld, Learning realistic human actions from movies, in: Proc. IEEE Conf. on Computer Vision & Pattern Recognition, 2008, pp. 1–8.
- [18] P. Dollár, V. Rabaud, G. Cottrell, S. Belongie, Behavior recognition via sparse spatio-temporal features, 2005.
- 575 [19] A. Jalal, Y.-H. Kim, Y.-J. Kim, S. Kamal, D. Kim, Robust human activity recognition from depth video using spatiotemporal multi-fused features, Pattern Recognition 61 (2017) 295–308.
- [20] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, L. Fei-Fei, Large-scale video classification with convolutional neural networks, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2014, pp. 1725–1732.
- 580 [21] J. Carreira, A. Zisserman, Quo vadis, action recognition? a new model and the kinetics dataset, arXiv:1705.07750 [cs]ArXiv: 1705.07750.  
URL <http://arxiv.org/abs/1705.07750>
- [22] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, T. Darrell, K. Saenko, Long-term recurrent convolutional networks for visual recognition and description, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2015, p. 2625–2634. doi:10.1109/CVPR.2015.7298878.  
URL <http://ieeexplore.ieee.org/document/7298878/>
- 585

- [23] S. Ji, W. Xu, M. Yang, K. Yu, 3d convolutional neural networks for human action recognition, *IEEE Trans. on Pattern Analysis and Machine Intelligence* 35 (1) (2013) 221–231. doi:10.1109/TPAMI.2012.59.
- [24] K. Hara, H. Kataoka, Y. Satoh, Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?, in: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2018, pp. 6546–6555.
- [25] C. Feichtenhofer, A. Pinz, A. Zisserman, Convolutional two-stream network fusion for video action recognition, arXiv:1604.06573 [cs]ArXiv: 1604.06573.  
URL <http://arxiv.org/abs/1604.06573>
- [26] J. Li, X. Liu, M. Zhang, D. Wang, Spatio-temporal deformable 3d convnets with attention for action recognition, *Pattern Recognition* 98 (2020) 107037.
- [27] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, J. Sivic, Netvlad: Cnn architecture for weakly supervised place recognition, in: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2016, pp. 5297–5307.
- [28] R. Girdhar, D. Ramanan, Attentional pooling for action recognition, in: *Proc. Advances in Neural Information Processing Systems*, 2017, pp. 34–45.
- [29] A. Miech, I. Laptev, J. Sivic, Learnable pooling with context gating for video classification, arXiv preprint arXiv:1706.06905.
- [30] P. Wang, Y. Cao, C. Shen, L. Liu, H. T. Shen, Temporal pyramid pooling-based convolutional neural network for action recognition, *IEEE Trans. on Circuits and Systems for Video Technology* 27 (12) (2016) 2613–2622.
- [31] Y. Xu, Y. Han, R. Hong, Q. Tian, Sequential video vlad: training the aggregation locally and temporally, *IEEE Trans. on Image Processing* 27 (10) (2018) 4933–4944.
- [32] A. Richard, J. Gall, A bag-of-words equivalent recurrent neural network for action recognition, *Computer Vision and Image Understanding* 156 (2017) 79–91. doi:10.1016/j.cviu.2016.10.014.

- 615 [33] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio,  
Learning phrase representations using rnn encoder-decoder for statistical machine translation,  
arXiv preprint arXiv:1406.1078.
- [34] L. Xia, C. Chen, J. Aggarwal, View invariant human action recognition using histograms of 3d  
joints, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition Workshops, 2012,  
620 pp. 20–27.
- [35] K. Soomro, A. R. Zamir, M. Shah, UCF101: A dataset of 101 human actions classes from  
videos in the wild, arXiv preprint arXiv:1212.0402.
- [36] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture  
for computer vision, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2016,  
625 pp. 2818–2826.
- [37] N. Passalis, A. Tefas, Training lightweight deep convolutional neural networks using bag-of-  
features pooling, IEEE Trans. on Neural Networks and Learning Systems.
- [38] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al., Gradient-based learning applied to docu-  
ment recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
- 630 [39] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine  
learning algorithms (2017). arXiv:cs.LG/1708.07747.
- [40] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, Tech. rep.,  
Citeseer (2009).

## **7.6 Human Action Recognition using Recurrent Bag-of-Features Pooling**

The appended paper follows.

# Human Action Recognition using Recurrent Bag-of-Features Pooling

Marios Krestenitis<sup>1</sup>, Nikolaos Passalis<sup>1</sup> Alexandros Iosifidis<sup>2</sup>,  
Moncef Gabbouj<sup>3</sup>, and Anastasios Tefas<sup>1</sup>

<sup>1</sup> Dept. of Informatics, Aristotle University of Thessaloniki, Greece  
{mikreste, passalis, tefas}@csd.auth.gr

<sup>2</sup> Dept. of Engineering, Electrical and Computer Engineering, Aarhus University,  
Denmark, ai@eng.au.dk

<sup>3</sup> Faculty of Information Technology and Communication, Tampere University,  
Finland, moncef.gabbouj@tuni.fi

**Abstract.** Bag-of-Features (BoF)-based models have been traditionally used for various computer vision tasks, due to their ability to provide compact semantic representations of complex objects, e.g., images, videos, etc. Indeed, BoF has been successfully combined with various feature extraction methods, ranging from handcrafted feature extractors to powerful deep learning models. However, BoF, along with most of the pooling approaches employed in deep learning, fails to capture the temporal dynamics of the input sequences. This leads to significant information loss, especially when the informative content of the data is sequentially distributed over the temporal dimension, e.g., videos. In this paper we propose a novel stateful recurrent quantization and aggregation approach in order to overcome the aforementioned limitation. The proposed method is inspired by the well-known Bag-of-Features (BoF) model, but employs a stateful trainable recurrent quantizer, instead of plain static quantization, allowing for effectively encoding the temporal dimension of the data. The effectiveness of the proposed approach is demonstrated using three video action recognition datasets.

## 1 Introduction

Computer vision is one of the most active and continuously expanding research fields, while with the advent of deep learning (DL) many powerful visual information analysis methods for high dimensional data have been recently proposed [8]. The typical pipeline of a visual information analysis approach involves at least the following two steps: a) feature extraction, in which lower level information is extracted from small spatial or temporal segments of the data, and b) feature aggregation, in which the extracted information is fused into a compact representation that can be used for the subsequent tasks, e.g., classification, retrieval, etc. DL unified, to some extent, these two steps by employing deep trainable feature extraction layers, e.g., convolutional layers, that are combined with naive pooling operators, e.g., max or average pooling, to lower the complexity of the

model and provide translation invariance. Indeed, the outstanding performance of Convolutional Neural Networks (CNNs) in complex and challenging image analysis tasks, has confirmed their ability to extract meaningful feature vectors with high discriminative power [8]. However, these powerful feature vectors are crushed through the pooling layers of the network, that usually implement the pooling operation in a less sophisticated manner. As we will demonstrate through this paper, this can lead to significant information loss, especially in cases where the informative content of the data is sequentially distributed over the spatial or temporal dimension, e.g., videos, which often requires extracting fine-grained temporal information, which is discarded by these pooling approaches.

The aforementioned limitations can be better understood through the following example. Consider the task of activity recognition in videos, where the action of *sitting down* must be distinguished from the action of *standing up*. Feature vectors can be extracted from every video frame or a sequence of them by using a deep CNN. However, the pooling layers, as the weak point of the network, dull the expressiveness of the extracted feature vectors and produce less discriminative representations by pooling over the time dimension. For example, assume that a sequence of feature vectors is extracted from a given video instance of action *sitting down*. Let that sequence, notated as  $\mathbf{S}_1 = [\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3]$ , be composed of three feature vectors  $\mathbf{a}_1 = [0, 0, 1]^T$ ,  $\mathbf{a}_2 = [0, 1, 0]^T$ , and  $\mathbf{a}_3 = [1, 0, 0]^T$ , which are the feature vectors that correspond to the sub-actions *standing above a chair*, *bending knees* and *sitting on a chair*, respectively. Similarly, consider the same feature vector sequence, but in reverse order, i.e.,  $\mathbf{S}_2 = [\mathbf{a}_3, \mathbf{a}_2, \mathbf{a}_1]$ , that represents a video instance of the activity *standing up*. Also, let  $\mathbf{s}_i$  denote the aggregated representation extracted for the  $i$ -th video. Note that when average or max pooling is applied over both sequences, then the same representation is extracted for both videos, i.e.,  $\mathbf{s}_1 = \mathbf{s}_2 = [\max_i[\mathbf{a}_i]_1, \max_i[\mathbf{a}_i]_2, \max_i[\mathbf{a}_i]_3]^T = [1, 1, 1]^T$  for max pooling (where the notation  $[\mathbf{x}]_i$  is used to refer to the  $i$ -th element of vector  $\mathbf{x}$ ) or  $\mathbf{s}_1 = \mathbf{s}_2 = \frac{1}{3} \sum_{i=1}^3 \mathbf{a}_i = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]^T$  for average pooling. Therefore, even though the employed CNN was capable of perfectly recognizing the fundamental sub-actions from still frames, the resulting deep model cannot discriminate between the two actions due to the employed pooling layer. Therefore average and max pooling layers are not capable of capturing the fine-grained spatial or temporal interactions between the feature vectors extracted from a given sequence. Note that in other cases, instead of employing a pooling layer, the extracted feature map may be flatten to a vector and fed to the subsequent fully connected layer. However, this approach makes it impossible for the network to handle inputs of arbitrary size, while it significantly reduces the invariance of the network to temporal shifts (the features must always arrive at the exact same moment).

In this paper, we propose a novel stateful recurrent pooling approach that is capable of overcoming these limitations. The proposed method builds upon the well-known Bag-of-Feature (BoF) model [11], which is capable of creating a constant-length representation of a multimedia object, e.g., video, audio, etc., by compiling a histogram over its quantized feature vectors. Therefore, every object is represented using a fixed-length histogram over the learned codewords.

The codewords/dictionary can be either learned using generative/reconstruction approaches [11], or by employing discriminative dictionary learning methods [4], which usually better fit classification tasks. This scheme found application in numerous computer vision tasks, including scene recognition [7], texture classification [16], etc. BoF can be also combined with deep neural networks to provide more powerful trainable pooling layers that can better withstand distribution shifts, while handling inputs of various sizes [9]. Despite its remarkable success in various tasks and its ability to handle variable size inputs, the main drawback of BoF-based methods is the loss of spatial and temporal information, as well as their inability to capture the geometry of input data [7].

These drawbacks severely limit the ability of BoF to process temporal or sequential data, such as video data, since it is not capable of capturing the temporal succession of events. To overcome this limitation, we suggest that the quantization process should take into account the order in which the features arrive, allowing for forming temporal codewords that also capture the interrelation between the feature vectors. As a result, a BoF method employing a stateful recurrent quantizer would be able to quantize the vector  $\mathbf{a}_2$  - “bending knees” (given in the previous examples) into a different codeword depending on whether it was following the vector  $\mathbf{a}_1$  - “standing above a chair” or the vector  $\mathbf{a}_3$  - “sitting on a chair”. In this way, it would be possible to extract a representation that can discriminate the standing up action from the sitting down action.

The proposed method is inspired by the BoF model, but employs a stateful trainable recurrent quantizer, instead of a plain static quantization approach to overcome the limitation of existing BoF formulations. In this way, the proposed method harnesses the power of a novel powerful recurrent quantization formulation in order to capture the temporal information of input data, which is crucial in classification tasks, such as activity recognition, while still maintaining all the advantages of the BoF model. In this work, the proposed Recurrent BoF (abbreviated as “ReBoF”) layer is used between the last feature extraction layer and the fully connected layer of a network. Therefore, instead of using other naive pooling layers, that can lead to significant loss of temporal information, the extracted feature vectors are quantized to a number of codewords in a recurrent manner, enabling us to encode the fine-grained temporal information contained in the original feature vectors. The resulting network can be trained in an end-to-end fashion using plain gradient descent and back-propagation, since the proposed ReBoF formulation is fully differentiable.

This allows for building powerful deep learning models for various visual information analysis tasks, as thoroughly demonstrated in this paper, while at the same time keeping the overall space and time complexity low compared to competitive approaches. In this way, the proposed method holds the credentials for providing fast and efficient human-centric perception methods for various embedded and robotics applications [6]. To the best of our knowledge, in this paper we propose the first stateful recurrent Bag-of-Features model that is capable of effectively modeling the temporal dynamics of video sequences. It is also worth

noting that existing BoF formulations, e.g., [9], merely provide models that fully ignore the temporal information.

The rest of the paper is structured as follows. In Section 2 the proposed Recurrent BoF (ReBoF) method is analytically derived. Then, the experimental evaluation of the ReBoF method is provided in Section 3. A thorough discussion on how ReBoF could be used for practical applications, along with its limitations, are provided in Section 4, while conclusions are drawn and future work is discussed in Section 5.

## 2 Proposed Method

In this Section, we first briefly introduce the regular (non-recurrent) Bag-of-Features model. Then, we derived the proposed Recurrent Bag-of-Features formulation, draw connections with the regular Bag-of-Features model and discuss how it can be used for video classification tasks.

### 2.1 Bag-of-Features

Let  $\mathcal{X} = \{x_i\}_{i=1}^N$  be a set of  $N$  videos to be represented using the standard BoF model. From each video  $N_i$  feature vectors are extracted:  $\mathbf{x}_{ij} \in R^D$  ( $j = 1, \dots, N_i$ ), where  $D$  is the dimensionality of each feature vector. BoF provides a way to efficiently aggregate these features into a fixed-length histogram. To this end, each feature vector is first quantized into a predefined number of codewords, by employing a codebook  $\mathbf{V} \in R^{N_K \times D}$ , where  $N_K$  is the number of codewords. This codebook is usually learned by clustering all feature vectors into  $N_K$  clusters [11]. Clustering algorithms, such as  $k$ -means, can be used to this end, with each centroid,  $\mathbf{v}_k \in R^D$  ( $k = 1, \dots, N_K$ ), corresponding to a codeword. Then, the quantized feature vectors of each object are accumulated to form the final histogram. Even though several feature quantization approaches have been proposed [11], this work focuses on using a soft quantization approach that allows for learning the codebook using regular back-propagation, along with the rest of the parameters of the model [10]. This can significantly improve the discriminative power of the model, since the codebook is adapted for the task at hand.

More specifically, each feature vector  $\mathbf{x}_{ij}$ , extracted from the  $i$ -th object, is quantized by measuring its similarity with each of the  $N_K$  codewords as:

$$[\mathbf{d}_{ij}]_k = \exp\left(\frac{-\|\mathbf{v}_k - \mathbf{x}_{ij}\|_2}{\sigma}\right) \in [0, 1], \quad (1)$$

where  $\sigma$  controls the fuzziness of the quantization process. Then, for each feature vector we obtain a fuzzy membership vector, after normalizing the observed similarities as:

$$\mathbf{u}_{ij} = \frac{\mathbf{d}_{ij}}{\|\mathbf{d}_{ij}\|_1} \in R^{N_K}. \quad (2)$$

Finally, the final histogram is extracted by accumulating all the normalized membership vectors as:

$$\mathbf{s}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \mathbf{u}_{ij} \in R^{N_K}. \quad (3)$$

The histogram  $\mathbf{s}_i$  has unit  $l_1$  norm, regardless the number of the extracted feature vectors, and provides an efficient representation of the feature vectors extracted from the corresponding video. This histogram is then fed to a multilayer perceptron (MLP) to classify the video. Note that the extracted histogram can be also used in other tasks, such as regression or retrieval.

## 2.2 Proposed Recurrent BoF

The histogram extracted using the regular BoF formulation, as described previously, discards any spatial or temporal information encoded by the feature vectors. To overcome this limitation, in this work we propose using a recurrent stateful quantizer, which allows for capturing and effectively encoding the temporal information expressed by the order in which the feature vectors arrive to the model. Note that in the case of video, we assume that the feature vector  $\mathbf{x}_{ij}$  corresponds to the  $j$ -th timestep of the  $i$ -th video sequence. Before deriving the proposed recurrent quantization approach, it is worth examining, from a probabilistic perspective, the quantization process involved in the BoF model. Using Kernel Density Estimation [1], we can estimate the probability of observing the feature vector  $\mathbf{x}_{ij}$ , given an input object  $x_i$ , as:

$$p(\mathbf{x}_{ij}|x_i) = \sum_{k=1}^{N_K} [\mathbf{s}_i]_k K(\mathbf{x}_{ij}, \mathbf{v}_k) \in [0, 1], \quad (4)$$

where the histogram  $\mathbf{s}_i \in \mathbb{R}^{N_K}$  separately adjust the density estimation, while  $K(\cdot)$  is a kernel function. Then, a maximum likelihood estimator can be used to actually calculate the histogram:

$$\mathbf{s}_i = \arg \max_{\mathbf{s}} \sum_{j=1}^{N_i} \log \left( \sum_{k=1}^{N_K} [\mathbf{s}]_k K(\mathbf{x}_{ij}, \mathbf{v}_k) \right). \quad (5)$$

The involved parameters (histogram) can be estimated as  $\mathbf{s}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \mathbf{u}_{ij}$ , as shown in [1], where

$$[\mathbf{u}_{ij}]_k = \frac{K(\mathbf{x}_{ij}, \mathbf{v}_k)}{\sum_{l=1}^{N_K} K(\mathbf{x}_{ij}^{(t)}, \mathbf{v}_l)} \in [0, 1]. \quad (6)$$

Using this formulation, we can re-derive the regular BoF with soft-assignments. Also, note that we can also replace the Gaussian kernel used in (1), which typically requires tuning the width  $\sigma$ , with an easier to use hyperbolic kernel, which

does not require tuning any hyper-parameter. The hyperbolic kernel also proved to be stabler and easier to use when the proposed method was combined with deep neural networks. Therefore, we can now measure the similarity between each feature vector and the codewords in order to quantize the feature vectors as:

$$[\mathbf{d}_{ij}]_k = \sigma(\mathbf{x}_{ij}^T \mathbf{v}_k) \in \mathbb{R}^{N_K}, \quad (7)$$

where

$$\sigma(x) = \frac{1}{1 + \exp(-x)}, \quad (8)$$

is the logistic sigmoid function. Note that this formulation still ignores the temporal information, since it provides no way to encode the current *state* of the quantizer. Therefore, we extend (7) in order to take into account the temporal information, as expressed by the histogram extracted until the current step, as:

$$[\mathbf{d}_{ij}]_k = \sigma(\mathbf{V} \mathbf{x}_{ij} + \mathbf{V}_h(\mathbf{r}_t \odot \mathbf{s}_{i,j-1})) \in \mathbb{R}^{N_K}, \quad (9)$$

where  $\mathbf{V}_h \in \mathbb{R}^{N_K \times N_k}$  is a weight matrix that is used to transfer the gated histogram vector into the quantization space,  $\mathbf{s}_{i,j-1}$  is the histogram extracted from previous quantizations (state) and  $\mathbf{r}_j \in \mathbb{R}^{N_K}$  is the output of a reset gate, introduced to ensure the long-term stability of the model. The additional parameters introduced in this formulation are learned during the training process using back-propagation. The proposed method also employs a reset gate, inspired by the GRU model [2], to further increase the stability of the learning process. Therefore, the reset gate is defined as:

$$\mathbf{r}_{ij} = \sigma(\mathbf{W}_r \mathbf{x}_{ij} + \mathbf{U}_r \mathbf{s}_{i,j-1}) \in \mathbb{R}^{N_K}, \quad (10)$$

where  $\mathbf{W}_r \in \mathbb{R}^{N_K \times D}$  and  $\mathbf{U}_r \in \mathbb{R}^{N_K \times N_K}$  are the weight matrices used to implement the reset gate.

Then, the  $l_1$  normalized membership vector is computed similarly to the regular BoF model as:

$$\mathbf{u}_{ij} = \frac{\mathbf{d}_{ij}}{\|\mathbf{d}_{ij}\|_1}. \quad (11)$$

Note that the initial state  $\mathbf{s}_{i,0}$  is initialized to:

$$\mathbf{s}_{i,0} = \frac{1}{N_K} \mathbf{1}, \quad (12)$$

where  $N_K$  is the number of codewords and  $\mathbf{1} \in \mathbb{R}^{N_K}$  is a vector of all ones. This ensures that quantizer's output will be always a properly normalized membership vector. Therefore, the histogram is recurrently updated as:

$$\mathbf{s}_{i,j} = (\mathbf{1} - \mathbf{z}_{ij}) \odot \mathbf{s}_{i,j-1} + \mathbf{z}_{ij} \odot \mathbf{u}_{ij} \in \mathbb{R}^{N_K}. \quad (13)$$

The update gate  $\mathbf{z}_{i,j}$ , which controls how much the current histogram will be updated, is defined as:

$$\mathbf{z}_{ij} = \sigma(\mathbf{W}_z \mathbf{x}_{ij} + \mathbf{U}_z \mathbf{s}_{i,j-1}) \in \mathbb{R}^{N_K}, \quad (14)$$

where  $\mathbf{W}_z \in \mathbb{R}^{N_K \times D}$  and  $\mathbf{U}_z \in \mathbb{R}^{N_K \times N_K}$  are parameters of the update gate. Finally, to compile the final histogram we average all the intermediate histograms as:

$$\mathbf{s}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \mathbf{s}_{i,j} \in \mathbb{R}^{N_K}. \quad (15)$$

Back-propagation can be directly used to learn all the parameters of the proposed ReBoF model. Note that ReBoF allows for capturing their temporal information, since it is capable of recursively processing the input feature vectors. First, the proposed recurrent stateful quantizer is employed to quantize the input feature vectors, as described by (9) and (11). Then, these vectors are employed to update the state of the quantizer, as described by (13), and allowing for compiling the resulting histogram. It is worth noting that ReBoF, similarly to all BoF-based models, is capable of processing varying-length input sequences.

ReBoF provides a significant advantage over existing BoF formulations, since it allows to capture the temporal information of sequential input data. This allows the ReBoF model to effectively tackle challenging video analysis problems, e.g., video retrieval, activity recognition, etc. Therefore, to apply ReBoF we: a) use a convolutional neural network to extract a feature vector  $\mathbf{x}_{ij}$  from each frame of a video, and b) feed the extracted feature vectors to the ReBoF model in order to extract a compact representation for each video. This allows ReBoF to process videos of arbitrary duration, while creating fixed-length compact representations of them. Note that the whole architecture can be trained in an end-to-end fashion, since the proposed ReBoF formulation is fully differentiable.

### 3 Experimental Evaluation

The proposed method was evaluated using three video activity recognition datasets, the UTKinect-Action3D [15] dataset, the UCF101 dataset [12] and a more challenging dataset, the Complex UCF101, which was designed to evaluate the ability of the methods to capture the temporal dimension of video sequences, as it will be described below. The UTKinect-Action3D [15] consists of 10 types of human activities in indoor settings. Samples for each action are collected from 10 different people that perform every activity two times. The provided RGB frames were used for all of the conducted experiments. Since there is no official training/testing split provided, a 50%-50% subject-based split strategy was employed, i.e., the videos of the first five subjects were included in the training set and the rest of them were used to form the testing set. Hence, a quite challenging setup was created, as the activities belonging in the testing set were performed from unseen subjects. The UCF101 dataset [12] is widely used for benchmarking action recognition models. The dataset contains 13,320 action instances belonging in 101 classes, that can be grouped in five generic categories. For all the experiments conducted in this paper, the first evaluation split of the dataset was used.

We also created a challenging and more complex dataset based on the UCF101 dataset to better demonstrate the ability of the proposed method to capture the

temporal dynamics of video data. To this end, we compiled a dataset by mixing instances from different activities of the UCF101 dataset together. More specifically, 10 activities of UCF101 (split 1) were selected to be joined together. Every action of this subset was joined with each one of the remaining, leading to 90 complex actions. One can further understand the significance of encoding the temporal information of these instances by considering that a sample of action “A” combined with one of action “B” (let name this complex activity class “AB”) must be separated from samples of complex activities from class “BA”. Note that “AB” and “BA” videos contain the same set of frames (for a specific instance of “A” and “B”), but in a different order. Hence, 114 samples were selected for each class, as this was the minimum number of instances contained in the selected initial classes. The selected 10 action classes were the following: *ApplyEyeMakeup*, *ApplyLipstick*, *Billiards*, *BoxingPunchingBag*, *BoxingSpeedBag*, *Haircut*, *Hammering*, *TableTennisShot*, *TennisSwing*, and *Typing*. Then, the  $i$ -th sample of initial class “A” is combined with the  $i$ -th sample of initial class “B” and so on, leading to 7,380 training and 2,880 testing data equally balanced among the 90 classes. The compiled dataset is called “Complex UCF” in the rest of this paper.

Table 1: UTKinect-Action3D Evaluation

Method	# Codewords / GRU Units	Test Accuracy (%)
Average Pooling	-	40.83
GRU	512	47.71
ReBoF	512	<b>54.64</b>

Table 2: UCF101 Evaluation

Method	# Codewords / GRU Units	Test Accuracy (%)
Avg. Pooling	-	70.32 $\pm$ 0.43
GRU	2048	71.04 $\pm$ 0.20
ReBoF	1024	<b>72.02 <math>\pm</math> 0.68</b>

For the UTKinect-Action3 and UCF101 datasets, every video instance was uniformly sampled in time in order to extract a predefined number of frames,

Table 3: Complex UCF101 Evaluation

Method	# Codewords / GRU Units	Test Accuracy (%)
Average Pooling	-	48.95
GRU	512	88.86 $\pm$ 2.04
ReBoF	512	89.25 $\pm$ 1.08
GRU	1024	88.62 $\pm$ 1.02
ReBoF	1024	<b>89.29</b> $\pm$ 0.89

denoted by  $N_f$ . The number of extracted frames was set to  $N_f = 30$  for the UTKinect-Action3D dataset and to  $N_f = 40$  for the UCF101 dataset. Shorter videos were looped over as many times as necessary to ensure that each video contains at least  $N_f$  frames, following the methodology described in the relevant literature [3]. Then, an Inception V3 model [13], pretrained in ImageNet, was used to extract a feature representation from each frame from the last average pooling layer of the network. Therefore, from each video, a sequence of  $N_f$  feature vectors was extracted. This sequence was then fed to the proposed ReBoF layer, followed by a hidden fully connected layer with 512 neurons and dropout with rate 0.5, as well as by the final classification layer. The ReLU activation function was used for the hidden layer, while the cross-entropy loss was used for training the model. The network was trained using the Adam optimizer and a learning rate of  $10^{-5}$ , apart from the pretrained feature extractors which were kept frozen. Furthermore, note that we also experimentally found out that scaling the extracted histogram by  $N_K$  significantly improved the convergence of the proposed method, especially when training from scratch. This scaling ensures that the gradients from the fully connected layers will not diminish as they are back-propagated to the previous layers. The network was trained for 800 epochs for the UTKinect-Action3D dataset and for 500 epochs for the UCF101 (the training/evaluation procedure was also repeated 3 times and the mean and standard deviation is reported). For the Complex UCF101 a slightly different setup was used. First, a 16-frame sliding window, with overlap of 4 frames, was applied on every activity instance of UCF101 dataset, while a pretrained 3D ResNeXt-101 [3] was used to extract a feature vector from each window. The features were extracted from the last average pooling layer of the network. Therefore, a 32-length sequence ( $N_f = 32$ ) of 2048-dimensional feature vectors were extracted for each video action. The training process stopped when 99.9% accuracy was achieved in training set (for the average pooling baseline, the network was trained for 50 epochs). The evaluation of ReBoF and GRU methods was repeated 3 times and the mean accuracy and standard deviation on the test set are reported.

The performance of the proposed method was also compared to two other established pooling methods. The same feature extractors and classification block was used to ensure a fair comparison between the methods. First, global average pooling (denoted by “Average Pooling” in the rest of the paper), over the temporal dimensions of the input sequence, was used instead of the proposed ReBoF method. Furthermore, a more powerful recurrent aggregation model, a GRU [2], was also employed to aggregate the extracted features.

First, the proposed method was evaluated on the UTKinect-Action3D, while the results are reported in Table 1. The proposed method greatly outperforms the other two evaluated methods, leading to the highest accuracy (54.64%) using 512 codewords. For the GRU also 512 units were employed, since at this point GRU achieved its best accuracy, which is however significantly lower (47.71%) compared to the proposed ReBoF method. Both ReBoF and GRU outperform the plain Average Pooling, since they are capable of effectively modeling the temporal dynamics of the input video sequences allowing for better discriminating between similar activities, such as *stand up* and *sit down*.

Moreover, in Fig. 1 the effect of using a wider range of codewords and number of GRU units in the classification accuracy on the UTKinect-Action3D dataset is evaluated using the two methods that achieve the highest performance (GRU and ReBoF). In all cases, the proposed method leads to higher accuracy compared to the GRU method. Furthermore, the proposed method allows for reducing the size of the extracted representation, since it outperforms the best GRU model (512 units) using just 128 codewords. This allows for reducing the size of the extracted representation and, as a result, the number of parameters in the subsequent fully connected layer. Both methods achieve their best performance for 512-dimensional representations. After this point, the accuracy for both models drops, mainly due to overfitting phenomena.

Again, similar conclusions can be drawn from the evaluation results on the UCF101 dataset, which are reported in Table 2. Note that even though UCF101 is a less challenging dataset, in terms of temporal dependence, the proposed method still outperforms the rest of the evaluated methods, achieving the highest accuracy (72.02%) for 1024 codewords. Again, all the methods were tuned to use the best number of codewords/representation length to ensure a fair comparison. Note that the proposed method outperforms the GRU while using representations with half the size of the ones used by the GRU. The effect of using different number of codewords and recurrent units is also evaluated in Fig. 2. Again, the proposed method outperforms the GRU method regardless the number of used codewords, while it achieves comparable accuracy using 2 to 4 times smaller representations.

Finally, the proposed method was also evaluated on the Complex UCF dataset. The results are provided in Table 3. As expected, Average Pooling fails to overpass the 50% test accuracy, since “AB” activities cannot be distinguished from those of “BA”, due to employing global averaging, which completely discards the temporal information. On the other hand, ReBoF again achieves the highest accuracy (89.29% when 1024 codewords are used). It is worth noting that even

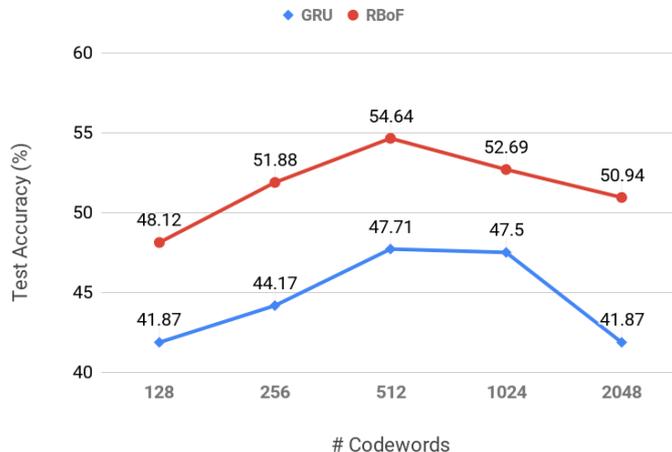


Fig. 1: UTKinect-Action3D Evaluation: Effect of using different number of codewords/recurrent units for the ReBoF and GRU methods

though spatio-temporal information is already encapsulated in the extracted feature vectors, since 3D kernels were used in feature extraction block, GRU again achieves lower accuracy compared to the ReBoF for both representation sizes.

## 4 Discussion

In this section, some further practical details regarding the employment of ReBoF method are presented, providing more insight into the proposed method compared to our previous works, e.g., [5]. A more thorough inspection of equations (9), (10), (13) and (14) implies that ReBoF shares some similarities with GRU units [2], since both ReBoF and GRUs use a similar update-reset gate structure. Therefore, the RBoF method can be implemented by modifying an existing (and optimized) GRU implementation by a) replacing the output activation function (in order to ensure the quantization of the feature vectors), b) setting the initial state  $s_{i,0}$  (to ensure that the histogram vector will maintain a unit  $l^1$  norm) and c) appropriately initializing the codebook. This allows us to simply modify existing and highly optimized GRU implementations to provide highly efficient ReBoF implementations. As far as the number of employed codewords, a rule of thumb, based on the current work’s analysis, could be to initialize ReBoF with 512 codewords, since it can provide a balanced trade-off among feature space dimensionality and model accuracy.

Furthermore, the advantages of employing ReBoF architecture in the corresponding framework for video analysis might be limited under certain circumstances. In case that temporal information is not crucial to distinguish different

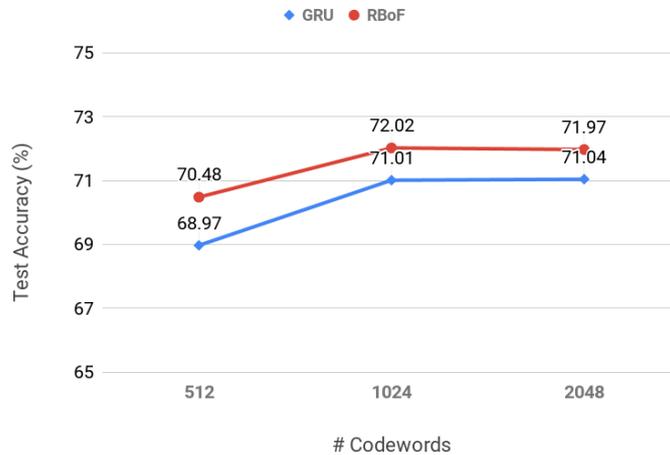


Fig. 2: UCF101 Evaluation: Effect of using different number of codewords/recurrent units for the ReBoF and GRU methods

video instances, the employment of ReBoF is not guaranteed that will lead to significantly higher performance compared to a simpler architecture designed to analyze video instances by a single or a small set of frames. Yet, any method that discards temporal information is incapable to cope with tasks where crucial information is contained over temporal dimension of data. This is clearly demonstrated via the experimental results on UCF101 and Complex UCF101 datasets. In the first case, the employment of ReBoF slightly improves the model accuracy, since performing classification of the videos of UCF101 can be considered a not so challenging task in terms of temporal dependence, given that in many cases we can achieve quite high recognition accuracy just using one frame from the videos. On the contrary, in the second case of Complex UCF101, enclosing temporal information is crucial to distinguish different video activities and thus, the employment of ReBoF leads to significant performance improvements compared to methods that discard temporal information. Finally, it should be noted that feeding to the ReBoF layer feature vectors that a priori enclose temporal information might lead to improvements to some extent however, this could also disproportionately increase overall the complexity of the models.

## 5 Conclusions

A novel stateful Recurrent Bag-of-Features (ReBoF) model was proposed in this paper. ReBoF employs a stateful trainable recurrent quantizer, instead of a plain static quantization approach, as the one used in existing BoF formulations. This

allows ReBoF to capture the temporal information of the input data, which is crucial in classification tasks, such as activity recognition, while still maintaining all the advantages of the BoF model. ReBoF can be directly used in DL models and the resulting architecture can be trained in an end-to-end fashion using back-propagation, allowing for building powerful deep learning models for various visual information analysis tasks.

ReBoF opens several interesting future research directions. First, ReBoF can be also used for encoding the spatial information, instead of merely the temporal one. For example, the spatial information encoded in the feature vectors extracted from static images can be encoded by manipulating the extracted feature map as a sequence of feature vectors. This allows for overcoming one long-standing limitation of regular BoF formulation that led to the need of using complicated spatial segmentation schemes [7]. Furthermore, activity recognition can be further enhanced by combining a ReBoF layer over the spatial dimension, followed by a ReBoF layer over the temporal dimension. In this way, the spatio-temporal information can be more properly encoded by creating a spatiotemporal histograms. Finally, using ReBoF for other tasks, such as video retrieval and hashing [14], is expected to further boost the performance of existing methods by extracting compact, yet more discriminative representations.

## References

1. Bhattacharya, S., Sukthankar, R., Jin, R., Shah, M.: A probabilistic representation for efficient large scale visual recognition tasks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2593–2600 (2011)
2. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
3. Hara, K., Kataoka, H., Satoh, Y.: Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6546–6555 (2018)
4. Iosifidis, A., Tefas, A., Pitas, I.: Discriminant bag of words based representation for human action recognition. *Pattern Recognition Letters* **49**, 185–192 (2014)
5. Krestenitis, M., Passalis, N., Iosifidis, A., Gabbouj, M., Tefas, A.: Recurrent bag-of-features for visual information analysis. *Pattern Recognition* p. 107380 (2020)
6. Lane, N.D., Bhattacharya, S., Mathur, A., Georgiev, P., Forlivesi, C., Kawsar, F.: Squeezing deep learning into mobile and embedded devices. *IEEE Pervasive Computing* **16**(3), 82–88 (2017)
7. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. vol. 2, pp. 2169–2178 (2006)
8. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. *Nature* **521**(7553), 436 (2015)
9. Passalis, N., Tefas, A.: Learning bag-of-features pooling for deep convolutional neural networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 5755–5763 (2017)
10. Passalis, N., Tefas, A.: Neural bag-of-features learning. *Pattern Recognition* **64**, 277–294 (2017)

11. Sivic, J., Zisserman, A.: Video google: A text retrieval approach to object matching in videos. In: Proceedings of the International Conference on Computer Vision. p. 1470 (2003)
12. Soomro, K., Zamir, A.R., Shah, M.: UCF101: A dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:1212.0402 (2012)
13. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2818–2826 (2016)
14. Tang, J., Lin, J., Li, Z., Yang, J.: Discriminative deep quantization hashing for face image retrieval. *IEEE Transactions on Neural Networks and Learning Systems* (2018)
15. Xia, L., Chen, C., Aggarwal, J.: View invariant human action recognition using histograms of 3d joints. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 20–27 (2012)
16. Zhang, J., Marszałek, M., Lazebnik, S., Schmid, C.: Local features and kernels for classification of texture and object categories: A comprehensive study. *International Journal of Computer Vision* **73**(2), 213–238 (2007)

## **7.7 Temporal Attention-Augmented Graph Convolutional Network for Efficient Skeleton-Based Human Action Recognition**

The appended paper follows.

# Temporal Attention-Augmented Graph Convolutional Network for Efficient Skeleton-Based Human Action Recognition

Negar Heidari and Alexandros Iosifidis

Department of Engineering, Aarhus University, Denmark  
Emails: {negar.heidari, alexandros.iosifidis}@eng.au.dk

**Abstract**—Graph convolutional networks (GCNs) have been very successful in modeling non-Euclidean data structures, like sequences of body skeletons forming actions modeled as spatio-temporal graphs. Most GCN-based action recognition methods use deep feed-forward networks with high computational complexity to process all skeletons in an action. This leads to a high number of floating point operations (ranging from 16G to 100G FLOPs) to process a single sample, making their adoption in restricted computation application scenarios infeasible. In this paper, we propose a temporal attention module (TAM) for increasing the efficiency in skeleton-based action recognition by selecting the most informative skeletons of an action at the early layers of the network. We incorporate the TAM in a light-weight GCN topology to further reduce the overall number of computations. Experimental results on two benchmark datasets show that the proposed method outperforms with a large margin the baseline GCN-based method while having  $\times 2.9$  less number of computations. Moreover, it performs on par with the state-of-the-art with up to  $\times 9.6$  less number of computations.

## I. INTRODUCTION

Human action recognition has been a very popular research topic in recent years. RGB videos and different types of modalities such as depth, optical flow and human body skeletons can be used for this task [1], [2], [3]. Compared to other data modalities, human body skeletons encode compact and high level information representing the human pose and joints' motion while being invariant to viewpoint variations, motion speed, human appearance, and body scale [4], and it is robust to context noise. Considering the success of existing pose estimation techniques [5], [6], [7] and the availability of depth cameras [8], obtaining the human body skeleton data is much easier than before. Thus, skeleton-based human action recognition has attracted an increasing research interest in recent years and many deep learning-based methods were proposed to model both the spatial and temporal evolution of skeletons in a sequence. Some methods use recurrent neural network (RNN) architectures, like the Long Short-Term Memory (LSTM) network [9], which are suitable to model temporal dynamics [1], [2], [10], [11], [12], [13] and utilize the skeleton data as vectors formed by the body joints coordinates. Methods employing Convolutional Neural Networks (CNN) [14], [15], [16], [17], [18], [19] reorganize the body joints' coordinates of each pose to a 2D map which is a suitable

input format for CNNs. The high model complexity in all these methods make their training and inference processes very time consuming [16], [20]. Besides, these methods are not able to completely benefit from the non-euclidean structure of the skeleton data which represents the spatial body joints connections.

Graph Convolutional Networks (GCNs) have been very successful when applied to many pattern recognition tasks [21], [22], [23], [24], [25], [26], [27], [28], by generalizing the convolution operation from grid data into graph data structures. Recently, significant results have been obtained by employing GCNs for skeleton-based human action recognition [29], [30], [31], [32], [33], [34], [35]. In these methods, an action is represented as a sequence of body poses and each body pose is represented by a skeleton. The skeleton data is treated as a graph which models the spatial relationship between different body joints and the temporal dynamics in an action are expressed by a sequence of skeletons.

Most of the recently proposed GCN-based methods use deep feed forward networks to model the spatio-temporal features of body skeletons. Considering that these methods process all the body skeletons in a sequence depicting the performed action, this approach is not efficient in terms of memory consumption and computation time. While memory efficiency can be increased by employing network compression, weight pruning, quantization and low-rank approximation approaches [36], [37], [38], [39], by processing a large number of body skeletons in each sequence, the number of floating point operations (FLOPs) is still large. Thus, to address both memory and computational efficiency in skeleton-based human action recognition, not only we need more compact and lightweight network architectures, but also the number of FLOPs should be minimized. Hence, processing fewer body skeletons for action recognition is a large step towards increasing the computational efficiency of both training and inference processes.

In this paper, we argue that all body skeletons in a temporal sequence are not equally important for recognizing actions. For each action class, there exist body poses which are the most informative for the action, and we can extract sufficient information for action recognition by focusing on the skeletons of these body poses only. Our goal is to increase computational

efficiency while performing on par, or even better, compared to models which utilize all the body skeletons in a sequence for action recognition. In this regard, we propose a GCN-based model which is capable to select a subset of body skeletons for human action recognition. To select the most informative skeletons, we propose a trainable temporal attention module which measures the importance of each skeleton in a sequence and we employ this module in the GCN-based spatio-temporal model to increase its efficiency. The main contributions of our work are the following:

- We propose a temporal attention module (TAM) to extract the most informative skeletons in an action sequence, leading to increased computational efficiency in both the training phase and inference.
- We experimentally show that employing our proposed TAM with a light-weight GCN topology leads to state-of-the-art performance levels in two widely adopted datasets for skeleton-based action recognition.
- We show that a subset of skeletons is as informative as the full skeleton sequence, since our method performs on par with the state-of-the-art methods while increasing computational efficiency by a factor of up to  $\times 9.6$ . This is a major advantage of our method compared to the state-of-the-art, as it is suitable for real time action recognition under restricted computation scenarios.

The remainder of the paper is organized as follows. Section II discusses the related works and section III introduces the baseline method ST-GCN [30]. Section IV describes the proposed method. The conducted experiments and results are presented in section V, and the concluding remarks are drawn in section VI.

## II. RELATED WORK

Data driven skeleton-based human action recognition methods which use deep learning models are mainly categorized into RNN-based, CNN-based and GCN-based methods. RNN-based methods [1], [2], [10], [11], [12], [13] mostly employ LSTM networks [9] to model the temporal dynamics of skeletons' sequence. In these methods, the sequence of skeletons is introduced to the model as a sequence of vectors. Each vector is formed by the concatenated 3D coordinates of all body joints of skeleton. ST-LSTM [2] is one representative RNN-based method which captures the body joints' relationship in both temporal and spatial domain. CNN-based methods [40], [14], [15], [16], [17], [18], [19] convert the sequence of skeletons into pseudo-images by reorganizing the joints' coordinates into a 2D map and employ a state-of-the-art CNN model like ResNet [41], [42] to extract temporal and spatial features. Although, the CNN-based methods are easier to train than RNN-based methods, they employ deep network architectures with large receptive fields to perceive the semantics of the input map. Besides, since all these RNN-based and CNN-based methods convert the skeletons into a regular grid or a sequence, they cannot utilize the complex, irregular and non-Euclidean structure of the skeleton data.

Recently, many GCN-based methods for skeleton-based human action recognition have been proposed which treat the human skeleton data as a graph structure representing the body joints (graph nodes) and their natural connections with bones (graph edges). Since the GCNs are able to capture the embedded features in irregular structured data, the GCN-based methods achieve higher performance in skeleton-based human action recognition compared RNN-based and CNN-based methods. Spatio-temporal graph convolutional network (ST-GCN) [30] is the first GCN-based method proposed for action recognition. It receives the sequence of body skeletons directly as input and employs the GCNs' [21] aggregation rule to extract the spatial features of each skeleton in a sequence, while the temporal dynamics are modeled by using a temporal graph with fixed connections. Several methods have been proposed based on ST-GCN for skeleton based human action recognition which mostly focus on exploiting adaptive spatial graphs. 2s-AGCN [29] is one of the state-of-the-art methods proposed on top of ST-GCN [30]. This method adaptively learns the topology of the graph in different layers for each sequence of skeletons in an end-to-end manner. It updates the graph structure of the skeleton with a spatial attention and also a data dependent graph. Besides, it uses a two-stream framework to utilize both joints features and bones features in parallel. It introduces joints and bones data into two different models and, finally, the SoftMax scores of the two model are added to obtain the fused score and predict the action label. In DGNN [34] the skeleton data is represented as a directed acyclic graph to benefit from the relationship between joints and bones based on their kinematic dependency and it improved 2s-AGCN [29] by utilizing the motion data and also making the graph structure adaptive. GCN-NAS [31] follows a neural architecture search approach to explore an optimal GCN-based model in terms of graph structures. That is, it explores the search space to determine the best graph structure at each layer of the network, while it still employs the ST-GCN network topology. DPRL+GCNN [32] aims to select the most representative skeletons from the input sequence using deep reinforcement learning. It adjusts the chosen skeletons progressively by evaluating the models' performance in action recognition. Actional-Structural GCN (AS-GCN) [33] proposed an inference module based on encoder-decoder structure to capture richer action-specific correlations and also structural links between the joints in skeletons.

## III. SPATIAL TEMPORAL GRAPH CONVOLUTIONAL NETWORK (ST-GCN)

In this section, we describe ST-GCN model [30] as our baseline. It is formulated over a sequence of skeleton graphs with multiple layers of spatio-temporal graph convolution operations. The model receives the joint coordinate vectors and the constructed spatio-temporal graph as input and applies GCN to capture the embedded patterns of the graph and extract high level features of the skeletons. Each convolution layer integrates the human body information along both the spatial and temporal dimensions. The extracted features are

then passed to a fully connected layer which classifies the skeleton sequence to the action class using a standard SoftMax classifier. The entire model is trained with backpropagation to minimize the classification loss. The network architecture of ST-GCN [30] is constructed by 9 layers of spatio-temporal convolution units. In each layer the Resnet module is also applied and the output is followed by a batch normalization layer. In the following, the graph construction, and the spatial and temporal convolution units are described.

### A. Graph Construction

Each skeleton is represented as a graph which forms the hierarchical representation of skeleton sequences and the spatial patterns and temporal dynamics of these graphs are captured automatically. The spatio-temporal graph is constructed using the 2D or 3D human body joint coordinates as nodes, and spatial and temporal edges. The spatial edges correspond to the natural connectivity of body joints in the skeleton, and the temporal edges connect the same body joints across consecutive skeletons. Fig. 1 (right) shows the spatio-temporal skeleton graph.

Formally, an undirected spatio-temporal graph on a sequence of skeletons is denoted as  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , where the set of nodes:

$$\mathcal{V} = \{\nu_{ti} \mid t, i \in \mathbb{Z}, 1 \leq t \leq T, 1 \leq i \leq N\}$$

indicates  $N$  body joints of a skeleton in a sequence of  $T$  time steps and  $\mathcal{E}$  is the set of spatial (intra-skeleton) and temporal (inter-skeleton) connections. The graph structure is captured by the adjacency matrix  $\mathbf{A} \in \mathbb{R}^{N \times N}$  which is a symmetric binary matrix defined as  $\mathbf{A}_{ij} = 1$  if there is a connection between nodes  $\nu_{ti}$  and  $\nu_{tj}$  in time step  $t$ , otherwise  $\mathbf{A}_{ij} = 0$ . Given  $\mathbf{A}$  as the spatial graph adjacency matrix which represents the joint connections in a single skeleton, the normalized adjacency matrix  $\tilde{\mathbf{A}}$  with self connections is computed as:

$$\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}}, \quad (1)$$

where  $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$  and  $\tilde{\mathbf{D}}$  is the diagonal degree matrix of  $\tilde{\mathbf{A}}$ , i.e.  $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$ .

The body motions are grouped as concentric and eccentric, the spatially localized structure of a single skeleton is used to design the spatial partitioning process which divides the neighbor nodes of each body joint into three subsets: 1) the root node itself; 2) the nodes connected to the root which are closer to the skeletons' center of gravity than the root node; 3) the remaining roots' neighbors. The center of gravity (COG) is the average of all the skeleton joints' coordinates in a skeleton. Fig. 1 (left) shows the neighboring sets of each graph node which are presented with different colors [30], [29].

### B. Spatial Graph Convolution

Given the input spatial features of  $i^{th}$  joint,  $f_{in}(\nu_i)$ , the spatial graph convolution is defined as [30]:

$$f_{out}(\nu_i) = \sum_{\nu_j \in \mathcal{N}_i} \frac{1}{Z_{ij}} f_{in}(\nu_j) w(l_i(\nu_j)), \quad (2)$$

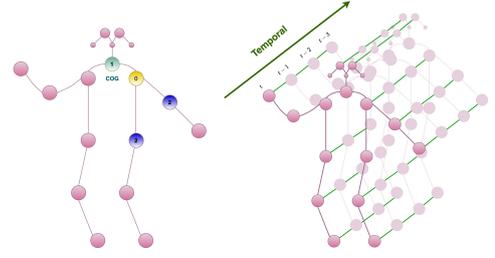


Fig. 1. Illustration of Spatio-temporal graph (right) in TA-GCN, and the spatially partitioned skeleton (left). Each partition set is illustrated in a different color.

where  $\mathcal{N}_i$  is the set of intra-frame neighbor vertices of  $\nu_i$  which are directly connected to it. As mentioned in III-A, ST-GCN uses a partitioning method to divide the neighbor vertices of each node into three subsets. According to this partitioning process,  $l_i$  in (2) maps each neighbor vertex into one of these three subsets and  $w$  is the weight matrix which is unique for each neighbor set and maps the target nodes' features into a new subspace. Note that the number of neighbor subsets and weight matrices are fixed while the number of neighbor nodes in each subset is varied for each node.  $Z_{ij}$  is a normalization factor which balances the contribution of each set of neighbor nodes in producing  $f_{out}(\nu_i)$  which contains the output features of the target node  $\nu_i$ .

In terms of implementation, the  $C$ -dimensional input feature vector for the node  $\nu_i$  is denoted as  $\mathbf{x}_i = f_{in}(\nu_i)$  and  $\mathbf{X} \in \mathbb{R}^{C_{in} \times T \times N}$  represents the input feature tensor for a sequence of skeletons, where  $C_{in}$  denotes the number of input channels,  $T$  is the number of skeletons and  $N$  is the number of body joints in each skeleton. The model receives the feature tensor  $\mathbf{X}$  as input and updates the nodes' feature vectors by applying the spatial convolution to produce  $\mathbf{X}' \in \mathbb{R}^{C_{out} \times T \times N}$  with  $C_{out}$  channels as output. By employing the layer-wise propagation rule of GCNs proposed in [21], the spatial convolution is formally defined as [30]:

$$\mathbf{X}' = ReLU \left( \sum_p (\hat{\mathbf{A}}_p \otimes \mathbf{M}_p) \mathbf{X} \mathbf{W}_p \right), \quad (3)$$

where  $\otimes$  is the element-wise product of two matrices. According to the partitioning process described in section III-A, each node has 3 subsets of neighbors. Therefore, the adjacency matrix  $\tilde{\mathbf{A}}$  is defined as the summation of 3 different adjacency matrices which are indexed by  $p$  as follows:

$$\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N = \sum_p \mathbf{A}_p \quad (4)$$

$\mathbf{A}_0 = \mathbf{I}_N$  represents the nodes' self connections and the normalized adjacency matrix for each subset is defined as:

$$\hat{\mathbf{A}}_p = \sum_p \mathbf{D}_p^{-\frac{1}{2}} \mathbf{A}_p \mathbf{D}_p^{-\frac{1}{2}}, \quad (5)$$

where  $\mathbf{D}_{(ii)_p} = \sum_j \mathbf{A}_{(ij)_p} + \varepsilon$ , with  $\varepsilon = 0.001$  used to avoid empty rows in degree matrix.  $\mathbf{M}_p \in \mathbb{R}^{N \times N}$  is

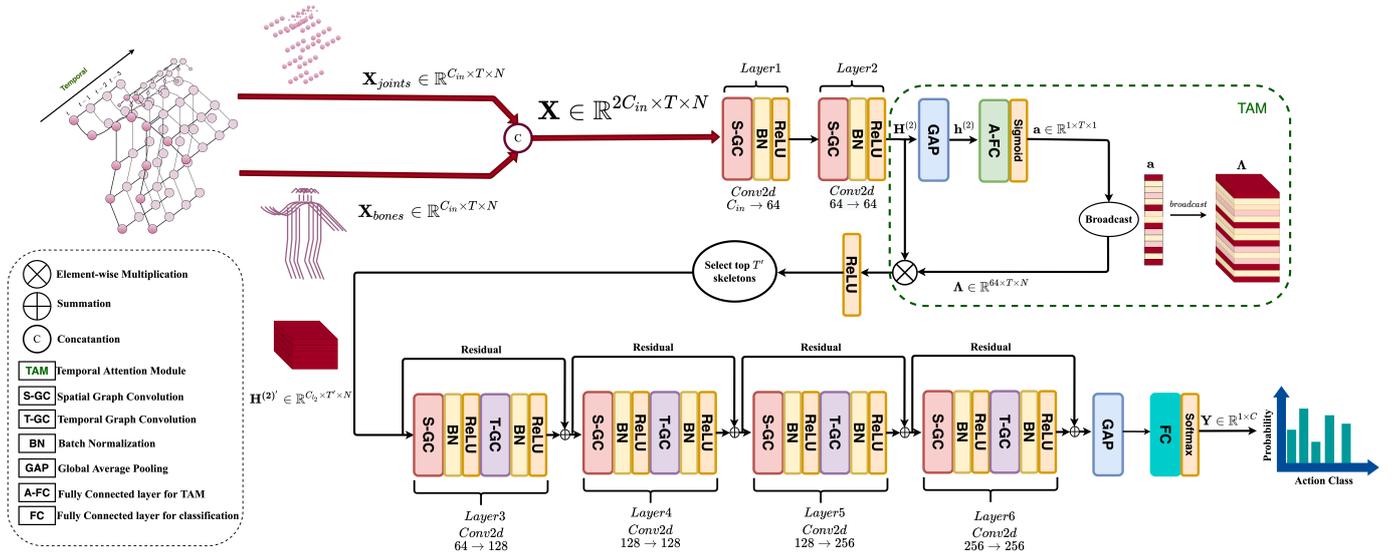


Fig. 2. Illustration of the proposed model diagram. The whole model architecture is composed of 6 spatio-temporal layers and the temporal attention module places after the first two layers to select the most informative skeletons. The highlighted skeletons' indices are sorted in descending manner and then extracted from the  $\mathbf{H}^{(2)}$ . The 3<sup>rd</sup> to 6<sup>th</sup> layers employ both spatial and temporal convolution.

a learnable attention map which highlights the elements of each adjacency matrix and it is initialized as an all-one matrix.  $\mathbf{W}_p \in \mathbb{R}^{C_{out} \times C_{in}}$  denotes the weight matrix which transforms the node features of each partition. In practice, a standard 2D convolution is used which performs  $C_{out} \times 1 \times 1$  convolutions, then the resulting tensor is multiplied with the masked normalized adjacency matrix on the last dimension  $N$ .

### C. Temporal Graph Convolution

To capture the temporal dynamics in a skeleton sequence and consider the motions taking place in an action, we need to propagate information related to the nodes' features in other skeletons of the sequence. To this end, a temporal convolution is applied on the output tensor of the spatial convolution step. In temporal dimension, each node is only connected to its corresponding node in its previous and next skeletons, so the number of neighbors for each node is fixed to 2. Therefore, a 2D convolution with a predefined temporal kernel size  $K_t$  is applied to the output of (3),  $\mathbf{X}'$ , to aggregate the features of each body joint in different time steps.

## IV. TEMPORAL ATTENTION-AUGMENTED GRAPH CONVOLUTIONAL NETWORK (TA-GCN)

In this section we describe an end-to-end temporal attention-augmented GCN model for efficient skeleton-based human action recognition. In order to extract discriminative features in temporal dimension of data, we propose TAM which highlights the most informative skeletons in a sequence. Based on this process, we apply skeleton selection to highly reduce the overall computational cost both during training and inference.

Similar to most of the recently proposed works [30], [29], the proposed model is composed of multiple layers of spatio-temporal units. In each unit, the embedded spatial features in

each skeleton are first extracted by employing a standard 2D convolution as described in Eq. (3). Motivated by [29], the learnable spatial attention map  $M_p$  is added (and not element-wise multiplied) to the graph adjacency matrix  $A_p$ . Adding the attention map to the graph adjacency matrix has the advantage that, not only the strength of the existing graph edges can be highlighted, but relationships between graph nodes which are not connected in the adjacency matrix based on the physical connectivity of the human body skeleton joints can also be captured [29]. Such connections can be important in effectively describing actions, e.g. encoding the relationships between the joints of the human arms and the head is particularly important for describing actions like 'hand waving'. Thus, the spatial attention module is employed to selectively focus on the most informative joints in each skeleton and the spatial convolution in Eq. (3) takes the following form:

$$\mathbf{X}' = \text{ReLU} \left( \sum_p (\hat{A}_p + M_p) \mathbf{X} \mathbf{W}_p \right) \quad (6)$$

The TAM takes a tensor  $\mathbf{H}^{(l)} \in \mathbb{R}^{C_l \times T \times N}$  as input, which can be the input data, i.e.  $\mathbf{H}^{(0)} = \mathbf{X}$ , or the output of the  $l^{\text{th}}$  hidden layer of the network. First, two average pooling operations in both feature and spatial dimensions is performed to produce the  $\mathbf{h}^{(l)} \in \mathbb{R}^{1 \times T \times 1}$ , which denotes the average feature value of each skeleton in the sequence. Then, this tensor is introduced to a fully connected layer which transforms features in temporal dimension  $T$  to produce the attention tensor  $\mathbf{a} \in \mathbb{R}^{1 \times T \times 1} = \{a_0, a_1, \dots, a_T\}$  as follows:

$$\mathbf{a} = \text{Sigmoid} \left( \mathbf{h}^{(l)} \Theta \right), \quad (7)$$

where  $\Theta \in \mathbb{R}^{T \times T}$  denotes the learnable transformation matrix, and the resulted attention tensor  $\mathbf{a}$  indicates the importance of

each skeleton in the sequence.

To highlight the most informative skeletons, we create the attention tensor  $\mathbf{\Lambda} \in \mathbb{R}^{C_l \times T \times N}$ , a duplicated version of attention map  $\mathbf{a}$  with  $C_l \times N$  copies, which is subsequently dot multiplied to  $\mathbf{H}^{(l)}$  as follows:

$$\hat{\mathbf{H}}^{(l)} = \text{ReLU}(\mathbf{H}^{(l)} \otimes \mathbf{\Lambda}), \quad (8)$$

where  $\otimes$  denotes element-wise multiplication. To select a subset of  $T'$  skeletons from  $\hat{\mathbf{H}}^{(l)}$ , the values in the attention map  $\mathbf{a}$  are sorted in descending order and the skeletons corresponding to the  $T'$  highest attention values are selected to be introduced into next layers of the network. To improve the computational efficiency of model in both training and inference phases, it would be preferable to use the TAM in the early layers of the network so that the next layers will process less number of skeletons.

The proposed model is composed of 6 GCN layers and 1 TAM. The first two GCN layers, map data into 64 dimensional feature space using spatial convolution followed by batch normalization layer and element-wise ReLU activation function. The mapped data is then introduced into the TAM which selects a subset of skeletons. In order to find the most discriminative skeletons using the TAM, the temporal convolution operation which smoothens the features in temporal domain is not used in the first two GCN layers. The 3<sup>rd</sup> and 4<sup>th</sup> layers change data dimensions from 64 to 128 and the last two GCN layers increase the number of channels to 256. In the last 4 GCN layers, both spatial and temporal convolutions are applied and each operation is followed by batch normalization and element-wise ReLU activation function. All the GCN layers except the first two layers utilize the ResNet [42] module to benefit from the input skeleton data too. The strides of the temporal convolution layers in 3<sup>rd</sup> and the 5<sup>th</sup> GCN layers are set to 2 as pooling layer. At the end, the refined spatio-temporal features are introduced into a global average pooling layer which produces an output feature vector of size  $256 \times 1$  for each sequence of skeletons and it is introduced into a fully connected layer which is equipped by a SoftMax classifier to classify the action. The model is trained with backpropagation in an end-to-end manner to minimize the classification error while it is learning to select the most discriminative skeletons.

Motivated by the method in [29], which utilizes both joints and bones features to enhance the classification performance, we also explore the skeleton bones' length and direction as the second-order information. Each bone is represented as a 3D vector bounded with two joints. The source joint is the one that is closer to the skeletons' center of gravity than the target joint. Therefore, each bone pointing from its source joint to the target joint holds both the length and direction information between two joints. After extracting the bone features from the skeleton data, the two feature tensors are concatenated on their first dimension and the merged joint-bone tensor of size  $2C_{in} \times T \times N$  is introduced to the model as input. The overall architecture of the proposed method is shown in Fig. 2.

## V. EXPERIMENTS

In this section, we describe experiments evaluating the performance of the proposed TA-GCN model in skeleton based HAR. We conducted experiments on two widely adopted datasets for evaluating the performance of skeleton-based action recognition methods. These datasets are:

1) *The NTU-RGB+D* [10]: is the largest multi-modality indoor-captured action recognition dataset. The dataset includes RGB videos, infrared videos, 3D skeletons and depth sequences. The 3D skeleton data is captured by the Microsoft Kinect-v2 camera and is used in our experiments. This dataset consists of 56,880 video clips from 60 different human action classes which are captured from three different views. Each skeleton is represented by 25 joints which are featured by 3D coordinate values. In our experiments, we follow exactly the same data splits and benchmark evaluations in [10]. In Cross-View (CV) benchmark, the training set contains 37,920 samples captured from cameras two and three and the test set contains 18,960 samples captured from the first camera. In Cross-Subject (CS) evaluation, 40,320 videos which represent 20 different action classes are used for training and the remaining 16,560 videos are used for testing. The number of frames for each sample is 300 and for the samples which have less than 300 frames, the frame sequence is repeated until it reaches 300 frames. In practice, the input data is a tensor of size  $(3 \times 300 \times 25)$ .

2) *The Kinetics-Skeleton* [43]: is a very large action recognition dataset that contains 300,000 video clips of 400 different human actions collected from YouTube. The 2D joints' coordinates  $(x, y)$  on every frame and their confidence score  $c$  are estimated using the public OpenPose toolbox [7]. Each skeleton in this dataset contains 18 body joints and each body joint is represented by a 3D vector  $(x, y, c)$ . The number of frames for each sample is fixed to 300 in a similar way explained for NTU-RGB+D dataset and the input data would be a tensor of size  $(3 \times 300 \times 18)$ . We use the Kinetics skeleton data which is provided by [30]. The training and validation sets contain 240,000 and 20,000 skeleton sequences, respectively.

### A. Experimental setting

All experiments were conducted on PyTorch deep learning framework [44] with 4 GRX 1080-ti GPUs and batch size of 32 and 128 for NTU-RGB+D and Kinetics datasets, respectively. The SGD optimizer is employed to optimize the model with Cross-entropy loss function through backpropagation with weight decay set to 0.0001. We followed exactly the same setting explained by authors for the baseline method [30] and the state-of-the-art [29]. In more details, the learning rate is not fixed through all epochs. For NTU-RGB+D dataset, it starts with 0.1 and it is divided by 10 at epochs 30, 40 while the total number of training epochs is fixed to 50. For Kinetics dataset, it starts with 0.1 and it is divided by 10 at epochs 45, 55 and the total number of training epochs is fixed to 65. For the Kinetics-Skeleton dataset we don't perform data-augmentation method which is used in [30]. In more details, we utilize all the 300 skeletons in a sequence.

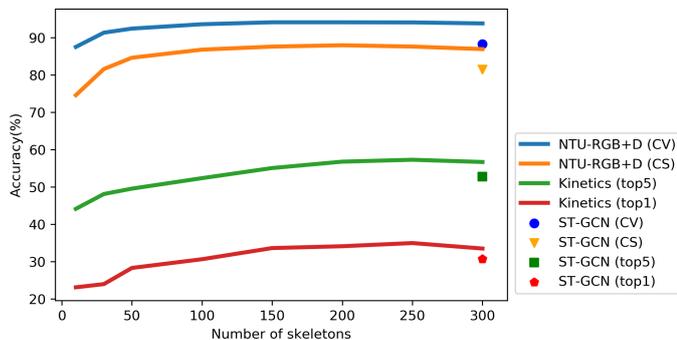


Fig. 3. The classification accuracy of the proposed TA-GCN method for a varying number of skeletons ( $T'$ ) selected by the proposed TAM. The performance of the the baseline ST-GCN is also provided.

### B. Do we need all skeletons for action recognition?

The hyperparameter  $T'$  of the proposed method defines the number of skeletons which are selected by TAM. To evaluate the model’s performance with different number of selected skeletons, we applied experiments with varying values of  $T' = \{10, 30, 50, 100, 150, 200, 250, 300\}$ . Fig. 3 shows the obtained performance in terms of classification accuracy on Kinetics-Skeletons dataset and NTU-RGB+D dataset with both CV and CS benchmarks. As can be seen in Fig. 3, by increasing the number of skeletons, the models’ accuracy is increased gradually. For NTU-RGB+D dataset, the best performance is equal to 94.2% and 87.97% for CV and CS benchmarks, respectively. This performance is achieved by selecting 150 most informative skeletons and increasing the number of skeletons doesn’t improve the performance necessarily. This confirms our hypothesis that the model does not need to process all available skeletons to perform action classification. Since the Kinetics-Skeleton dataset is more challenging than NTU-RGB+D, both top1 and top5 accuracies are reported. The best top1 and top5 accuracies are 34.95% and 57.28%, respectively, which are achieved by selecting 250 skeletons. Overall, we can observe that by applying the proposed TAM for skeleton selection we can achieve competitive performance even for a small number of selected skeletons. For example, the performance in NTU-RGB-D (CV) for  $T' = 50$  is equal to 92.44%, compared to 93.83% corresponding to using all  $T = 300$  skeletons. This is an advantage for application scenarios with computational restrictions.

### C. Comparison with the state-of-the-art methods

We compare the performance of the proposed method with the of state-of-the-art methods in Table. I and II on the NTU-RGB+D and Kinetics-Skeletons datasets, respectively. Table I is divided in three blocks grouping the methods in three categories, i.e. RNN-based, CNN-based and GCN-based methods, respectively. As can be seen, CNN-based methods perform better than RNN-based methods in general, while the state-of-the-art performance is achieved by GCN-based methods.

TABLE I  
COMPARISONS OF THE CLASSIFICATION ACCURACY WITH STATE-OF-THE-ART METHODS ON THE TEST SET OF NTU-RGB+D DATASET

Method	CS(%)	CV(%)	#Streams	Skel.sel.
HBRNN [1]	59.1	64.0	5	✗
Deep LSTM [10]	60.7	67.3	1	✗
ST-LSTM [2]	69.2	77.7	1	✗
STA-LSTM [11]	73.4	81.2	1	✓
VA-LSTM [12]	79.2	87.7	1	✗
ARRN-LSTM [13]	80.7	88.8	2	✗
Two-Stream 3DCNN [14]	66.8	72.6	2	✗
TCN [15]	74.3	83.1	1	✗
Clips+CNN+MTLN [16]	79.6	84.8	1	✗
Synthesized CNN [17]	80.0	87.2	1	✗
3scale ResNet152 [18]	85.0	92.3	1	✗
CNN+Motion+Trans [19]	83.2	89.3	2	✗
ST-GCN [30]	81.5	88.3	1	✗
DPRL+GCNN [32]	83.5	89.8	1	✓
AS-GCN [33]	86.8	94.2	2	✗
2s-AGCN [29]	88.5	95.1	2	✗
GCN-NAS [31]	89.4	95.7	2	✗
DGNN [34]	89.9	96.1	4	✗
<b>TA-GCN (<math>T' = 150</math>)</b>	87.7	94.2	1	✓
<b>2s-TA-GCN (<math>T' = 150</math>)</b>	88.5	95.1	2	✓
<b>4s-TA-GCN (<math>T' = 150</math>)</b>	89.91	95.8	4	✓

TABLE II  
COMPARISONS OF THE CLASSIFICATION ACCURACY WITH STATE-OF-THE-ART METHODS ON THE TEST SET OF KINETICS-SKELETON DATASET

Method	Top1(%)	Top5(%)	#Streams	Skel.sel.
Deep LSTM [10]	16.4	35.3	1	✗
TCN [15]	20.3	40.0	1	✗
ST-GCN [30]	30.7	52.8	1	✗
AS-GCN [33]	34.8	56.5	2	✗
2s-AGCN [29]	36.1	58.7	2	✗
DGNN [34]	36.9	59.6	4	✗
GCN-NAS [31]	37.1	60.1	2	✗
1s-TA-GCN ( $T' = 250$ )	34.95	57.28	1	✓
2s-TA-GCN ( $T' = 250$ )	36.1	58.72	2	✓
4s-TA-GCN ( $T' = 250$ )	36.9	59.77	4	✓

The results for the NTU-RGB+D dataset in Table I show that the proposed method, TA-GCN, outperforms all the CNN-based and RNN-based methods with a large margin. Besides, TA-GCN outperforms ST-GCN, which is the baseline in GCN-based methods, and DPRL+GCNN by a large margin in both CV and CS benchmarks. Compared to AS-GCN, the proposed method achieves higher accuracy in CS benchmark and a similar performance in CV benchmark. Here we should also note that the only competing GCN-based method that performs skeleton selection, i.e. DPRL+GCNN [32], performs poorly compared to all variants of the proposed method. The TA-GCNs’ competitive methods are 2s-AGCN and GCN-NAS and the best performing method is DGNN. For Kinetics-Skeleton dataset (Table II), the proposed method outperforms the RNN-based methods, ST-GCN and AS-GCN methods and it has competitive performance with 2s-AGCN. The best performing methods on this dataset are GCN-NAS and DGNN. When the proposed model is trained using two different data streams joints and bones, (2s-TA-GCN), it achieves similar top1 accuracy with 2s-AGCN and its top5 accuracy exceeds

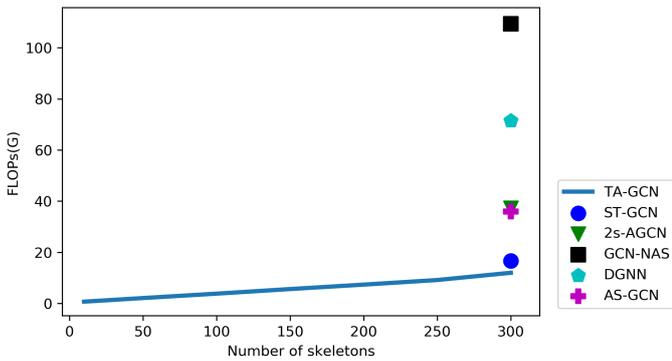


Fig. 4. Computational complexity comparison between the proposed method, when it selects different number of skeletons, and all the state-of-the-art methods which utilize all the skeletons of the input sequence.

TABLE III  
COMPARISONS OF THE COMPUTATIONAL COMPLEXITY WITH STATE-OF-THE-ART METHODS ON THE ON CV BENCHMARK OF NTU-RGB+D DATASET. THE NUMBER OF FLOPS AND PARAMS FOR 1s-TA-GCN ( $T' = 150$ ) ARE 5.64G AND 2.24M, RESPECTIVELY.

Method	FLOPs	# Params
1s-TA-GCN ( $T' = 150$ )	$\times 1$	$\times 1$
2s-TA-GCN ( $T' = 150$ )	$\times 2$	$\times 2$
4s-TA-GCN ( $T' = 150$ )	$\times 4$	$\times 4$
ST-GCN	$\times 2.9$	$\times 1.3$
AS-GCN	$\times 6.3$	$\times 3.2$
2s-AGCN	$\times 6.6$	$\times 3$
DGNN	$\times 12.6$	$\times 3.6$
GCN-NAS	$\times 19.3$	$\times 8.9$

2s-AGCN slightly. The 4s-TA-GCN which is trained using four data streams outperforms DGNN (top5) and achieves competitive performance with GCN-NAS while it has 4.8 times less computational complexity. Table. III shows the computational complexity comparison in terms of FLOPs and model parameters (Params) between the GCN-based competing methods on NTU-RGB+D (CV) dataset. Since the source code of the DPRL-GCNN method is not provided by the authors, it is not mentioned in the Table. III. Fig. 4, illustrates the computational complexity (FLOPs) of our method, when it selects different number of skeletons, and also the state-of-the-art methods which process all the skeletons. It can be seen that TA-GCN has less computational complexity compared to all the competing methods, even when it process all the skeletons in the sequence (i.e. for  $T' = 300$ ).

Columns 3 and 4 in Tables I and II indicate the number of network streams used by each GCN-based method and whether each method performs skeleton selection (or otherwise employs all  $T = 300$  skeletons). As can be seen, methods employing more than one streams generally outperform those with one stream. Although these methods achieve higher classification accuracy compared to single-stream methods, they have high computational complexity. Similar to our proposed method, the ST-GCN and DPRL-GCNN methods train only one network stream. They utilize only the joint data to train the model and we fuse both joint and bone information in

the first layer (input) which prevents increasing number of model parameters and FLOPs. The results show that not only does our method outperform the baseline ST-GCN in terms of classification accuracy, but also it has 2.9 times less FLOPs and 1.3 times less number of parameters. Therefore, TA-GCN can be a strong and efficient baseline for GCN-based human action recognition which achieves better performance than ST-GCN. The 2s-AGCN, GCN-NAS methods are built on top of ST-GCN method and train two networks using joint and bone data separately which doubles the number of model parameters and FLOPs. DGNN which has the best classification accuracy on both datasets trains 4 different networks with joint, bone, joint-motion and bone-motion data. DGNN and GCN-NAS have 12.6 and 19.3 times more FLOPs than the proposed method, respectively, and from 3.6 to 8.9 times more parameters.

To conduct a fair comparison, we also trained TA-GCN method with 4 different data streams separately. In Table. I, II, 2s-TA-GCN shows the ensembled softmax scores of the trained models with joint and bone data and 4s-TA-GCN shows the classification accuracy of the model when the softmax scores of all the 4 streams, joint, bone, joint-motion and bone-motion, are ensembled. The results indicate that training multiple models and ensembling the softmax scores can increase the accuracy by at least 1%. 2s-TA-GCN has a similar accuracy with 2s-AGCN while it has 3.3 times less FLOPs and 1.5 times less parameters. 4s-TA-GCN outperforms the competing methods 2s-AGCN and GCN-NAS with 1.65, 4.82 times less FLOPs, respectively. As can be seen in Table. III, GCN-NAS has the maximum number of FLOPs and it has competitive performance with 2s-TA-GCN which has with 9.6 times less computational complexity. In comparison with state-of-the-art, our proposed TA-GCN achieves good performance with much less computational complexity which makes TA-GCN applicable for many practical tasks with limited computational capacity.

## VI. CONCLUSION

In this paper, we proposed a temporal attention-augmented GCN to improve the computational efficiency in skeleton-based action recognition. Our method trains the attention mechanism to select the most informative skeletons for each action in an end-to-end manner. On two widely used benchmark datasets, the proposed method has competitive performance with the state-of-the-art, while being up to 10 times less computationally complex, and it outperforms the baseline in terms of both classification accuracy and computational complexity with a large margin. Therefore, it can be an efficient and strong baseline for skeleton-based action recognition.

## ACKNOWLEDGMENT

This work received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR). This publication reflects the authors’ views only. The European Commission is not responsible for any use that may be made of the information it contains.

## REFERENCES

- [1] Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *IEEE conference on computer vision and pattern recognition*, 2015, pp. 1110–1118.
- [2] J. Liu, A. Shahrourdy, D. Xu, and G. Wang, "Spatio-temporal lstm with trust gates for 3d human action recognition," in *European conference on computer vision*. Springer, 2016, pp. 816–833.
- [3] A. Iosifidis, A. Tefas, and I. Pitas, "View-invariant action recognition based on artificial neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 3, pp. 412–424, 2012.
- [4] F. Han, B. Reily, W. Hoff, and H. Zhang, "Space-time representation of people based on 3d skeletal data: A review," *Computer Vision and Image Understanding*, vol. 158, pp. 85–105, 2017.
- [5] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *IEEE Conference on Computer Vision and Pattern Recognition*. Ieee, 2011, pp. 1297–1304.
- [6] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5693–5703.
- [7] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7291–7299.
- [8] Z. Zhang, "Microsoft kinect sensor and its effect," *IEEE multimedia*, vol. 19, no. 2, pp. 4–10, 2012.
- [9] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.
- [10] A. Shahrourdy, J. Liu, T.-T. Ng, and G. Wang, "Ntu rgb+ d: A large scale dataset for 3d human activity analysis," in *IEEE conference on computer vision and pattern recognition*, 2016, pp. 1010–1019.
- [11] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu, "An end-to-end spatio-temporal attention model for human action recognition from skeleton data," in *AAAI conference on artificial intelligence*, 2017.
- [12] P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue, and N. Zheng, "View adaptive recurrent neural networks for high performance human action recognition from skeleton data," in *IEEE International Conference on Computer Vision*, 2017, pp. 2117–2126.
- [13] L. Li, W. Zheng, Z. Zhang, Y. Huang, and L. Wang, "Skeleton-based relational modeling for action recognition," *arXiv preprint arXiv:1805.02556*, vol. 1, no. 2, p. 3, 2018.
- [14] H. Liu, J. Tu, and M. Liu, "Two-stream 3d convolutional neural network for skeleton-based action recognition," *arXiv preprint arXiv:1705.08106*, 2017.
- [15] T. S. Kim and A. Reiter, "Interpretable 3d human action analysis with temporal convolutional networks," in *IEEE conference on computer vision and pattern recognition workshops*. IEEE, 2017, pp. 1623–1631.
- [16] Q. Ke, M. Bennamoun, S. An, F. Sohel, and F. Boussaid, "A new representation of skeleton sequences for 3d action recognition," in *IEEE conference on computer vision and pattern recognition*, 2017, pp. 3288–3297.
- [17] M. Liu, H. Liu, and C. Chen, "Enhanced skeleton visualization for view invariant human action recognition," *Pattern Recognition*, vol. 68, pp. 346–362, 2017.
- [18] B. Li, Y. Dai, X. Cheng, H. Chen, Y. Lin, and M. He, "Skeleton based action recognition using translation-scale invariant image mapping and multi-scale deep cnn," in *IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 2017, pp. 601–604.
- [19] C. Li, Q. Zhong, D. Xie, and S. Pu, "Skeleton-based action recognition with convolutional neural networks," in *IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 2017, pp. 597–600.
- [20] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," in *International Conference on Learning Representations*, 2014.
- [21] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *International Conference on Learning Representations*, 2017.
- [22] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in neural information processing systems*, 2015, pp. 2224–2232.
- [23] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International conference on machine learning*, 2016, pp. 2014–2023.
- [24] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Advances in neural information processing systems*, 2016, pp. 1993–2001.
- [25] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in neural information processing systems*, 2017, pp. 1024–1034.
- [26] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5115–5124.
- [27] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, "Neural relational inference for interacting systems," *International conference on machine learning*, 2018.
- [28] N. Heidari and A. Iosifidis, "Progressive graph convolutional networks for semi-supervised node classification," *arXiv preprint arXiv:2003.12277*, 2020.
- [29] L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Two-stream adaptive graph convolutional networks for skeleton-based action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 026–12 035.
- [30] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *AAAI conference on artificial intelligence*, 2018.
- [31] W. Peng, X. Hong, H. Chen, and G. Zhao, "Learning graph convolutional network for skeleton-based human action recognition by neural searching," in *AAAI conference on artificial intelligence*, 2020, pp. 2669–2676.
- [32] Y. Tang, Y. Tian, J. Lu, P. Li, and J. Zhou, "Deep progressive reinforcement learning for skeleton-based action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5323–5332.
- [33] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, and Q. Tian, "Actional-structural graph convolutional networks for skeleton-based action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3595–3603.
- [34] L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Skeleton-based action recognition with directed graph neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7912–7921.
- [35] X. Gao, W. Hu, J. Tang, J. Liu, and Z. Guo, "Optimized skeleton-based action recognition via sparsified graph regression," in *ACM International Conference on Multimedia*, 2019, pp. 601–610.
- [36] B. Zhang, J. Han, Z. Huang, J. Yang, and X. Zeng, "A real-time and hardware-efficient processor for skeleton-based action recognition with lightweight convolutional neural network," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 66, no. 12, pp. 2052–2056, 2019.
- [37] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in neural information processing systems*, 2015, pp. 3123–3131.
- [38] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *International Conference on Learning Representations*, 2016.
- [39] D. T. Tran, A. Iosifidis, and M. Gabbouj, "Improving efficiency in convolutional neural network with multilinear filters," *Neural Networks*, vol. 105, pp. 328–339, 2018.
- [40] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, and J. Sun, "Cascaded pyramid network for multi-person pose estimation," in *IEEE conference on computer vision and pattern recognition*, 2018, pp. 7103–7112.
- [41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [42] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *AAAI conference on artificial intelligence*, 2017.
- [43] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev *et al.*, "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.
- [44] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

## **7.8 Progressive Spatio-Temporal Graph Convolutional Network for Skeleton-Based Human Action Recognition**

The appended paper follows.

# PROGRESSIVE SPATIO-TEMPORAL GRAPH CONVOLUTIONAL NETWORK FOR SKELETON-BASED HUMAN ACTION RECOGNITION

Negar Heidari and Alexandros Iosifidis

Department of Electrical and Computer Engineering, Aarhus University, Denmark

## ABSTRACT

Graph convolutional networks (GCNs) have been very successful in skeleton-based human action recognition where the sequence of skeletons is modeled as a graph. However, most of the GCN-based methods in this area train a deep feed-forward network with a fixed topology that leads to high computational complexity and restricts their application in low computation scenarios. In this paper, we propose a method to automatically find a compact and problem-specific topology for spatio-temporal graph convolutional networks in a progressive manner. Experimental results on two widely used datasets for skeleton-based human action recognition indicate that the proposed method has competitive or even better classification performance compared to the state-of-the-art methods with much lower computational complexity.

**Index Terms**— Graph Convolutional Network, Human Action Recognition, Neural Architecture Search, Efficiency

## 1. INTRODUCTION

Skeleton-based human action recognition has attracted great attention in recent years. Compared to other data modalities such as RGB videos, optical flow and depth images, skeletons are invariant to viewpoint and illumination variations, context noise, and body scale while they encode high level features of the body poses and motions in a simple and compact graph structure [1]. Recently, many deep learning approaches have been proposed to model the spatial and temporal evolution of a sequence of skeletons which can be easily obtained by the available pose estimation techniques [2, 3, 4]. The existing methods are generally categorized into Convolutional Neural Network (CNN)-based methods, Recurrent Neural Network (RNN)-based methods and Graph Convolutional Network (GCN)-based methods. RNN-based methods mostly utilize Long Short-Term Memory (LSTM) networks [5] to model the temporal evolution of the skeletons which represented as vectors of body joint coordinates [6, 7, 8, 9, 10, 11]. CNN-based methods represent each skeleton as a 2D matrix which is a suitable input data form for CNNs and model the temporal dynamics by large receptive fields in a deep network [12, 13, 14, 15, 16, 17]. The main drawbacks of RNN-based and CNN-based methods are their high model complexity and lack of ability to benefit from the graph structure of skeletons. Many GCN-based methods have been proposed in recent years which utilize the well-known pose estimation toolboxes, such as OpenPose [4], to extract a sequence of skeletons from a video and employ GCN [18] to capture the spatio-temporal connections between the body

joints and extract high level features for the human action recognition task. The GCN-based methods have been very successful in processing non-Euclidean data structures and they have reached significant results in skeleton-based human action recognition. The first work in this area is ST-GCN [19] which receives a sequence of skeletons as input and after constructing a spatio-temporal graph, it applies several graph convolutions in both spatial and temporal domains to aggregate action-related information in both dimensions. 2s-AGCN [20] is proposed on top of ST-GCN and improves the performance in human action recognition by learning the graph structure adaptively using the similarity between the graph joints in addition to their natural connections in a body skeleton. Moreover, it uses both joint and bone features of body skeletons in a two-stream network topology that fuses the SoftMax scores at the last layer of each stream to predict the action. AS-GCN [21] captures richer action-specific correlations by an inference module which has an encoder-decoder structure. DPRL+GCNN [22] and TA-GCN [23] aim to make the inference process more efficient by selecting the most informative skeletons from the input sequence. GCN-NAS method [24] employs the neural architecture search approach to automatically design the GCN model. It explores a search space with multiple dynamic graph modules and different powers of Laplacian matrix (multi-hop modules) in order to improve the representational capacity. However, this method does not optimize the model topology in terms of number of layers and neurons in each layer. Thus, similar to ST-GCN and 2s-AGCN, GCN-NAS also trains a 10-layer model formed by layers of pre-defined sizes and suffers from a long training time and high computational complexity.

Although all the recently proposed GCN-based methods have reached a promising performance in human action recognition, they are not suitable for real-time applications with limited computational resources due to their high computational complexity. Finding an optimized and compact topology for the model can make a large step towards addressing the computational and memory efficiency in these methods. However, in order to find the best network topology which is able to reach state-of-the-art performance, an extensive set of experiments training different network topologies is needed. While for other types of deep learning models neural architecture search methods have been proposed for automatically finding good network topologies [25, 26], for ST-GCNs there exist no such methods to date. In this paper, we propose a method for automatically finding an optimized and problem-specific network topology for spatio-temporal GCNs (ST-GCNs) which work with non-Euclidean data structures. The proposed method grows the network topology progressively and concurrently optimizes the network parameters. Experimental results show that the proposed method achieves competitive performance compared to the state-of-the-art methods while it builds a compact model with up to 15 times less number of parameters compared to the existing methods leading to a much more efficient inference process.

This work received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR). This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

## 2. SPATIO-TEMPORAL GCN

In this section, the ST-GCN method [19] is introduced as a baseline of our method. It receives a sequence of  $T_{in}$  skeletons  $\mathbf{X} \in \mathbb{R}^{C_{in} \times T_{in} \times V}$  as input, where  $C_{in}$  is the number of input channels, and  $V$  is the number of joints in each skeleton. The spatio-temporal graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  is constructed, where  $\mathcal{V}$  is the node set of 2D or 3D body joint coordinates and  $\mathcal{E}$  encodes spatial and temporal connections between the joints as graph edges. Spatial edges are the natural connections between the body joints and temporal edges connect the same body joints in consecutive skeletons. In such a graph structure, each node can have different number of neighbors. To create a fixed number of weight matrices in convolution operation, the ST-GCN method uses the spatial partitioning process to fix the neighborhood size of each node. This process categorizes the neighboring nodes of each joint into 3 subsets: 1) the root node itself, 2) the neighboring nodes which are closer to the skeleton's center of gravity (COG) than the root node and 3) the nodes which are farther away from the COG than the root node. Figure 1 shows a spatio-temporal graph (left) and the 3 neighboring subsets around a node with different colors. In this setting, each skeleton has 3 binary adjacency matrices (one for each neighboring node type) which are indexed by  $p$  and represent the spatial graph structure in each time step.

The normalized adjacency matrix  $\hat{\mathbf{A}}_p \in \mathbb{R}^{V \times V}$  is defined as:

$$\hat{\mathbf{A}}_p = \mathbf{D}_p^{-\frac{1}{2}} \mathbf{A}_p \mathbf{D}_p^{-\frac{1}{2}}, \quad (1)$$

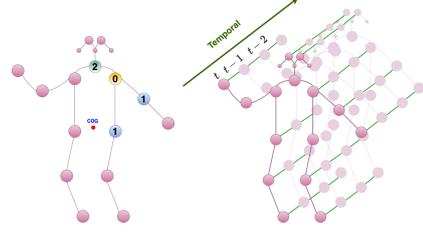
where  $\mathbf{D}_{(ii)_p} = \sum_j \mathbf{A}_{(ij)_p} + \varepsilon$ , is the degree matrix and  $\varepsilon = 0.001$  ensures that there would be no empty rows in  $\mathbf{D}_{(ii)_p}$ . The adjacency matrix of the root node denoting the nodes' self-connections is set to  $\mathbf{A}_1 = \mathbf{I}_V$ . In each layer  $l$ , first a spatial convolution is applied on the output of the previous layer  $\mathbf{H}^{(l-1)}$  to update the nodes' features of each skeleton using the layer-wise propagation rule of GCN [18]:

$$\mathbf{H}_s^{(l)} = ReLU \left( \sum_p (\hat{\mathbf{A}}_p \otimes \mathbf{M}_p^{(l)}) \mathbf{H}^{(l-1)} \mathbf{W}_p^{(l)} \right), \quad (2)$$

where  $\mathbf{M}_p^{(l)} \in \mathbb{R}^{V \times V}$  is a learnable attention matrix which highlights the most important joints in a skeleton for each action and  $\otimes$  is the element-wise product of two matrices. The input of the first layer is defined as  $\mathbf{H}^{(0)} = \mathbf{X}$ . Each neighboring subset is mapped with a different weight matrix  $\mathbf{W}_p^{(l)} \in \mathbb{R}^{C^{(l)} \times C^{(l-1)}}$  and  $\mathbf{H}_s^{(l)} \in \mathbb{R}^{C^{(l)} \times T^{(l)} \times V}$  with  $C^{(l)}$  channels is introduced to the temporal convolution. The temporal dynamics in a sequence of skeletons are captured by propagating the features of the nodes through the time domain. The goal of temporal convolution is to benefit from the motions taking place in each action class to predict the labels. Based on the constructed spatio-temporal graph, each node of a skeleton is connected into its corresponding node in the next and previous skeletons. Therefore, the temporal convolution aggregates node features in  $K$  consecutive skeletons by employing a standard 2D convolution with a predefined kernel size  $K \times 1$ .

## 3. PROPOSED METHOD

In this section we describe the proposed data-driven method for learning a problem-dependant topology for ST-GCN in an end-to-end manner. The Progressive ST-GCN (PST-GCN) method builds a compact ST-GCN in terms of both width and depth by employing the proposed ST-GCN augmentation module which increases the receptive field in spatial and temporal convolutions progressively.



**Fig. 1.** Illustration of Spatio-temporal graph (right), and the neighboring subsets in spatial partitioning process in different colors (left).

### 3.1. ST-GCN augmentation module (ST-GCN-AM)

The goal of each model's layer with temporal convolution, spatial convolution and residual connection is to extract features from a sequence of skeletons by capturing the spatial structure in each skeleton and the motions taking place in each action class. The proposed ST-GCN-AM is responsible for building a new layer of the ST-GCN model in a progressive manner. Let us assume the ST-GCN model is already formed by  $l$  layers. ST-GCN-AM receives the existing model and its output  $\mathbf{H}^{(l)} \in \mathbb{R}^{C^{(l)} \times T^{(l)} \times V}$  as input and applies an iterative training process (indexed by  $t$ ) by forming a spatial convolution which updates the skeletons' features as follows:

$$\mathbf{H}_{s,t}^{(l+1)} = ReLU \left( \sum_p (\hat{\mathbf{A}}_p + \mathbf{M}_{p,t}^{(l+1)}) \mathbf{H}^{(l)} \mathbf{W}_{p,t}^{(l+1)} \right). \quad (3)$$

Inspired by 2s-AGCN [20], the attention matrix  $\mathbf{M}_{p,t}^{(l+1)}$  is added to the normalized adjacency matrix  $\hat{\mathbf{A}}_p$  to both highlight the existing connections between the joints and add potentially important connections between the disconnected nodes. The spatial convolution defined in Eq. 3 is a standard 2D convolutional operation which performs  $1 \times 1$  convolutions to map the features of the  $l^{th}$  layer with  $C^{(l)}$  channels, into a new space with  $C^{(l+1)}$  channels. At the iteration  $t = 1$ , we set  $C_t^{(l+1)} = S$ , where  $S$  is a hyper-parameter of the method. Therefore, the size of each convolutional filter is  $C^{(l)} \times 1 \times 1$  and in order to have  $C_t^{(l+1)}$  channels in the output, we need  $C_t^{(l+1)}$  different filters of that size. Accordingly, the number of parameters in spatial convolution operation is  $C_t^{(l+1)} \times C^{(l)} \times 1 \times 1$ . The 2D spatial convolution is followed by a batch normalization layer and an element-wise ReLU activation function. The output of the spatial convolution  $\mathbf{H}_{s,t}^{(l+1)}$  is introduced to a temporal convolution which aggregates the features through the sequence in order to capture the temporal dynamics. Similar to spatial convolution, it is also a standard 2D convolution with  $C_t^{(l+1)}$  different filters, each of size  $C_t^{(l+1)} \times K \times 1$ , which aggregates the features of  $K$  consecutive skeletons with all  $C_t^{(l+1)}$  channels and it is followed by a batch normalization layer. The number of skeletons  $T^{(l+1)}$  of the layer depends on the stride and kernel size of the temporal convolution. Similar to the ST-GCN [19] and 2s-AGCN [20] models, a residual connection is added to each layer to stabilize the model by summing up the input of the layer with the layers' output. Since the number of channels is changed in each layer by the spatial convolution, the residual connection also transforms the input of the layer by a 2D convolution with  $C_t^{(l+1)}$  filters of size  $C^{(l)} \times 1 \times 1$  to have the same channel dimension as the layer's output. The residual output is added to the temporal convolution output and it is followed by an element-wise activation function ReLU to produce the output of the

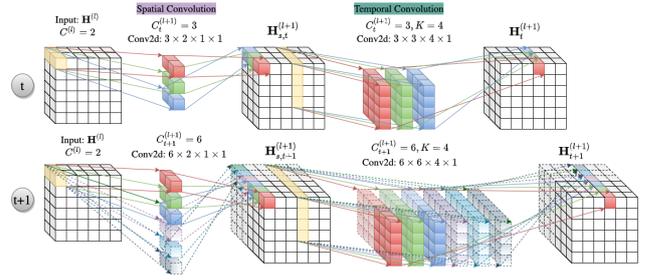
layer  $\mathbf{H}_t^{(l+1)}$  at iteration  $t$  which is of size  $C_t^{(l+1)} \times T^{(l+1)} \times V$ . In order to classify the extracted features, the global average pooling (GAP) operation is applied on  $\mathbf{H}_t^{(l+1)}$  to produce a feature vector of size  $C_t^{(l+1)} \times 1$  which is introduced to the fully connected classification layer. The classification layer is a linear transformation which maps features from  $C_t^{(l+1)}$  to  $N_{class}$  dimensional subspace where  $N_{class}$  denotes the number of classes. All the model’s parameters are fine-tuned based on back-propagation using the categorical loss value on training data and then, the classification accuracy  $\mathcal{A}_t^{(l+1)}$  on validation data is recorded.

At iteration  $t > 1$ , the network topology grows in terms of width by increasing the number of output channels in the current layer by  $S$  so that the filter size in all the 2D convolutional operations (spatial, temporal and residual) is increased. For the spatial convolution and the residual connection,  $C_t^{(l+1)}$  convolutional filters of size  $C_t^{(l)} \times 1 \times 1$  are needed. The first  $(t - 1)S$  filters are initialized by the learned parameters at iteration  $t - 1$  and the second newly added  $S$  filters are initialized randomly. The output shape of the spatial convolution is of size  $C_t^{(l+1)} \times C_t^{(l)} \times 1 \times 1$  which is introduced into the temporal convolution. Therefore, for the temporal convolution  $C_t^{(l+1)}$  filters of size  $C_t^{(l+1)} \times 1 \times 1$  are needed. As illustrated in Figure 2, both the width and length of the filters are increased in temporal convolution. The first  $(t - 1)S$  filters of size  $(t - 1)S \times 1 \times 1$  are initialized using the learned parameters at iteration  $t - 1$  (illustrated with solid structure) and the newly added parameters are initialized randomly (illustrated with dotted structure). The size of the spatial attention matrix  $\mathbf{M}_{p,t}^{(l+1)}$  does not change, since it only depends on the number of joints in each skeleton. Therefore, all its parameters are initialized to those obtained at iteration  $t - 1$ . The fully connected layer transforms the extracted features with a weight matrix of size  $C_t^{(l+1)} \times N_{class}$ , in which the first  $(t - 1)S \times N_{class}$  parameters are initialized to those from the previous iteration and the remaining ones are initialized randomly.

After initializing the new weights, all the model parameters are fine-tuned by back-propagation and the new classification accuracy  $\mathcal{A}_t^{(l+1)}$  of the model on validation data is recorded. Accordingly, the width progression of  $(l + 1)$  layer is evaluated based on the performance improvement rate, i.e.  $\alpha_w = (\mathcal{A}_t^{(l+1)} - \mathcal{A}_{t-1}^{(l+1)}) / \mathcal{A}_{t-1}^{(l+1)}$ . If the model performance has been increased at iteration  $t$  by increasing the number of output channels in the  $(l + 1)^{th}$  layer, its parameters are saved and the next iteration augmenting the layer by adding  $S$  new channels starts. If increasing the number of output channels in the current layer does not improve the model’s performance, i.e. when  $\alpha_w < \epsilon_w$  with  $\epsilon_w > 0$ , the newly added parameters are removed, all the model parameters are set to those obtained at iteration  $t - 1$ , and the width progression for the  $(l + 1)^{th}$  layer stops.

### 3.2. Progressive ST-GCN method

The progressive ST-GCN method (PST-GCN) optimizes the network topology in terms of both width and depth by employing the proposed ST-GCN-AM. PST-GCN algorithm starts with an empty network and returns the ST-GCN model with an optimized topology. The learning process starts by introducing the input data  $\mathbf{X} \in \mathbb{R}^{C_{in} \times T_{in} \times V}$  into the ST-GCN-AM which starts building the first layer with a fixed number of output channels and increases the width of the layer progressively until the model’s performance converges. In an iterative process, the algorithm builds several layers by employing the proposed module. Let us assume that the width progression is terminated at  $(l + 1)^{th}$  layer. At this point,



**Fig. 2.** Illustration of the proposed ST-GCN-AM building the model in layer  $l$  at iteration  $t$  and  $t + 1$ . It is assumed that  $C^{(l)} = 2$ ,  $S = 3$  and  $K = 4$ . Conv2d is the abbreviation for standard 2D convolutional operation.

the model’s performance is recorded and the rate of improvement is given by  $\alpha_d = (\mathcal{A}^{(l+1)} - \mathcal{A}^{(l)}) / \mathcal{A}^{(l)}$ , where  $\mathcal{A}^{(l)}$ ,  $\mathcal{A}^{(l+1)}$  denote the model’s performance before and after adding the  $(l + 1)^{th}$  layer to the model, respectively. If the addition of the last layer does not improve the model’s performance, i.e. when  $\alpha_d < \epsilon_d$  with  $\epsilon_d > 0$ , the newly added layer is removed and the algorithm stops growing the networks’ topology. As the final step, the model is fine-tuned using both training and validation sets.

## 4. EXPERIMENTS

We conducted experiments on two widely used large-scale datasets for evaluating the skeleton-based action recognition methods. The NTU-RGB+D [8] is the largest indoor-action-captured action recognition dataset which contains different data modalities including the 3D skeletons captured by Kinect-v2 camera. It contains 56,000 action clips from 60 different action classes and each action clip is captured by 3 cameras with 3 different views, and provides two different benchmarks, cross-view (CV) and cross-subject (CS). The CV benchmark contains 37,920 action clips captured from cameras 2 and 3 as training set and 18,960 action clips captured from the first camera as test set. In the CS benchmark, the actors in training and test set are different. The training set contains 40,320 clips and test set contains 16,560 clips. In this dataset, the number of joints in each skeleton is 25 and each sample has a sequence of 300 skeletons with 3 different channels each. The Kinetics-Skeleton [27] dataset is a widely used action recognition dataset which contains the skeleton data of 300,000 video clips of 400 different actions collected from YouTube. In this dataset each skeleton in a sequence has 18 joints which are estimated by the OpenPose toolbox [4] and each joint is featured by its 2D coordinates and confidence score. We use the data splits provided by [19] in which the training and test sets contain 240,000 and 20,000 samples, respectively.

The experiments are conducted on PyTorch deep learning framework [28] with 4 GRX 1080-ti GPUs, SGD optimizer and cross-entropy loss function. The model is trained for 50 epochs and batch size of 64 on NTU-RGB+D dataset, and the learning rate is initiated by 0.1 and it is divided by 10 at epochs 30, 40. The number of training epochs and batch size for Kinetics-Skeleton data set are set to 65 and 128, respectively while the learning rate starts from 0.1 and it is divided by 10 at epochs 45, 55. The hyper-parameters  $\epsilon_w$  and  $\epsilon_d$  are both set to  $1e - 4$ . Since we need a validation set to evaluate our progressive method in each step, we divided the training set of each dataset into train and validation subsets. The validation set is formed by randomly selecting 20% of the training samples in each

**Table 1.** Comparisons of the classification accuracy with state-of-the-art methods on the test set of NTU-RGB+D dataset

Method	CS(%)	CV(%)	#Streams
HBRNN [6]	59.1	64.0	5
Deep LSTM [8]	60.7	67.3	1
ST-LSTM [7]	69.2	77.7	1
STA-LSTM [9]	73.4	81.2	1
VA-LSTM [10]	79.2	87.7	1
ARRN-LSTM [11]	80.7	88.8	2
Two-Stream 3DCNN [12]	66.8	72.6	2
TCN [13]	74.3	83.1	1
Clips+CNN+MTLN [14]	79.6	84.8	1
Synthesized CNN [15]	80.0	87.2	1
3scale ResNet152 [16]	85.0	92.3	1
CNN+Motion+Trans [17]	83.2	89.3	2
ST-GCN [19]	81.5	88.3	1
DPRL+GCNN [22]	83.5	89.8	1
AS-GCN [21]	86.8	94.2	2
2s-AGCN [20]	88.5	95.1	2
1s-TA-GCN [23]	87.97	94.2	1
2s-TA-GCN [23]	88.5	95.1	2
GCN-NAS [24]	89.4	95.7	2
<b>PST-GCN</b>	87.9	94.33	1
<b>2s-PST-GCN</b>	88.68	95.1	2

action class and the remaining samples form the training set.

The performance of the proposed method is compared with the state-of-the-art methods in Tables 1 and 2 on NTU-RGB+D and Kinetics-Skeleton datasets, respectively. Besides, the computational complexity of our method is compared with GCN-based methods in terms of floating point operations (FLOPs) and model parameters in Table.3. Since the source code of DPRL+GCN is not provided by the authors, its computational complexity is not reported in Table 3. In order to compare our method with state-of-the-art methods which utilize two different data streams such as 2s-AGCN, AS-GCN and GCN-NAS, we also trained our model with two data streams, joints and bones, and reported the results as 2s-PST-GCN. The bone features represent the information of both length and direction of the connection between two joints. In two stream training, the PST-GCN finds the optimized network topology for the first data stream (joints) and the SoftMax scores are recorded. The optimized network topology is then fine-tuned using the second data stream (bones) for 30 epochs and the obtained SoftMax scores obtained in this step are fused by the recorded SoftMax scores to enhance the classification performance. The network topology which is used in ST-GCN, 2s-AGCN and GCN-NAS methods, comprise of 10 ST-GCN layers and one classification layer where the number of channels in ST-GCN layers are fixed to (64, 64, 64, 64, 128, 128, 128, 256, 256, 256), respectively. The optimized network topologies which are found by PST-GCN for NTU-RGB+D dataset in CV and CS benchmarks is a 8 layer network. The channel sizes in CV benchmark are (100, 80, 100, 40, 60, 80, 60, 80) and in CS benchmarks are (100, 80, 60, 100, 60, 100, 140, 80). The network topology which is found for Kinetics-skeleton dataset has 9 layers of sizes (80, 100, 100, 120, 100, 120, 140, 160, 180), respectively.

As can be seen in Table 1, the proposed PST-GCN method outperforms all the RNN-based and CNN-based methods with a large margin on NTU-RGB+D dataset in both CV and CS benchmarks. Compared to GCN-based methods, our method outperforms the ST-GCN and DPRL+GCN methods with a large margin while it has  $\times 2.3$  less number of FLOPs and  $\times 4.95$  less number of parameters than ST-GCN, according to Table.3. Compared to AS-GCN which trains a two stream model, PST-GCN performs better in both CV and CS benchmarks while it trains the model using only one

**Table 2.** Comparisons of the classification accuracy with state-of-the-art methods on the test set of Kinetics-Skeleton dataset

Method	Top1(%)	Top5(%)	#Streams
Deep LSTM [8]	16.4	35.3	1
TCN [13]	20.3	40.0	1
ST-GCN [19]	30.7	52.8	1
AS-GCN [21]	34.8	56.5	2
2s-AGCN [20]	36.1	58.7	2
1s-TA-GCN [23]	34.95	57.28	1
2s-TA-GCN [23]	36.1	58.72	2
GCN-NAS [24]	37.1	60.1	2
<b>PST-GCN</b>	34.71	57.08	1
<b>2s-PST-GCN</b>	35.53	58.2	2

**Table 3.** Comparisons of the computational complexity with GCN-based state-of-the-art methods.

Methods	#G FLOPs	#M Params
ST-GCN [19]	16.7	3.12
AS-GCN [21]	35.92	7.24
2s-AGCN [20]	37.32	6.9
1s-TA-GCN (NTU-CV-CS) [23]	5.64	2.24
2s-TA-GCN (NTU-CV-CS) [23]	11.28	4.48
1s-TA-GCN (Kinetics) [23]	9.17	2.24
2s-TA-GCN (Kinetics) [23]	18.34	4.48
GCN-NAS [24]	109.26	20.13
<b>PST-GCN (NTU-CV)</b>	7.2	0.63
<b>2s-PST-GCN (NTU-CV)</b>	14.4	1.26
<b>PST-GCN (NTU-CS)</b>	9.57	0.92
<b>2s-PST-GCN (NTU-CS)</b>	19.14	1.84
<b>PST-GCN (Kinetics)</b>	5.67	1.96
<b>2s-PST-GCN (Kinetics)</b>	11.34	3.92

data stream and it has  $\times 4.98$  less number of FLOPs and  $\times 11.49$  less number of parameters. The 2s-PST-GCN also outperforms AS-GCN with  $\times 2.49$  and  $\times 5.74$  less number of FLOPs and parameters, respectively and it performs on par with 2s-AGCN, 1s-TA-GCN and 2s-TA-GCN in both datasets. The TA-GCN method selects 150 and 250 most informative skeletons from the NTU-RGB+D and Kinetics-skeleton datasets, respectively. Therefore, it has less number of FLOPs than our method on NTU-RGB+D dataset. However, our proposed method is still more efficient in terms of number of parameters. The best performing method on both datasets is GCN-NAS which has also the highest computational complexity compared to our method and all other state-of-the-art methods. 2s-PST-GCN has competitive performance with GCN-NAS on NTU-RGB+D while it has  $\times 7.5$  less number of FLOPs and  $\times 15.97$  less number of parameters. On Kinetics-skeleton dataset, Table.2, the PST-GCN outperforms the RNN-based methods and ST-GCN method with a large margin while it has less computational complexity. Besides, the 2s-PST-GCN outperforms AS-GCN and performs on par with 2s-AGCN while it has less computational complexity in both cases.

## 5. CONCLUSION

In this paper, we proposed a method which builds a problem-dependant compact network topology for spatio-temporal graph convolutional networks in a progressive manner. The proposed method grows the network topology in terms of both width and depth by employing a learning process which is guided by the model’s performance. Experimental results on two large-scale datasets show that the proposed method performs on par, or even better, compared to more complex state-of-the-art methods in terms of classification performance and computational complexity. Therefore, it can be a strong baseline for optimizing the network topology in ST-GCN based methods.

## 6. REFERENCES

- [1] Fei Han, Brian Reily, William Hoff, and Hao Zhang, "Space-time representation of people based on 3D skeletal data: A review," *Computer Vision and Image Understanding*, vol. 158, pp. 85–105, 2017.
- [2] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake, "Real-time human pose recognition in parts from single depth images," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1297–1304.
- [3] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang, "Deep high-resolution representation learning for human pose estimation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5693–5703.
- [4] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7291–7299.
- [5] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber, "LSTM: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2016.
- [6] Yong Du, Wei Wang, and Liang Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1110–1118.
- [7] Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang, "Spatio-temporal LSTM with trust gates for 3D human action recognition," in *European Conference on Computer Vision*. Springer, 2016, pp. 816–833.
- [8] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang, "NTU RGB+D: A large scale dataset for 3D human activity analysis," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1010–1019.
- [9] Sijie Song, Cuiling Lan, Junliang Xing, Wenjun Zeng, and Jiaying Liu, "An end-to-end spatio-temporal attention model for human action recognition from skeleton data," in *AAAI Conference on Artificial Intelligence*, 2017, pp. 4263–4270.
- [10] Pengfei Zhang, Cuiling Lan, Junliang Xing, Wenjun Zeng, Jianru Xue, and Nanning Zheng, "View adaptive recurrent neural networks for high performance human action recognition from skeleton data," in *IEEE International Conference on Computer Vision*, 2017, pp. 2117–2126.
- [11] Lin Li, Wu Zheng, Zhaoxiang Zhang, Yan Huang, and Liang Wang, "Skeleton-based relational modeling for action recognition," *arXiv preprint arXiv:1805.02556*, vol. 1, no. 2, pp. 3, 2018.
- [12] Hong Liu, Juanhui Tu, and Mengyuan Liu, "Two-stream 3D convolutional neural network for skeleton-based action recognition," *arXiv preprint arXiv:1705.08106*, 2017.
- [13] Tae Soo Kim and Austin Reiter, "Interpretable 3D human action analysis with temporal convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 1623–1631.
- [14] Qihong Ke, Mohammed Bennamoun, Senjian An, Ferdous Sohel, and Farid Boussaid, "A new representation of skeleton sequences for 3D action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3288–3297.
- [15] Mengyuan Liu, Hong Liu, and Chen Chen, "Enhanced skeleton visualization for view invariant human action recognition," *Pattern Recognition*, vol. 68, pp. 346–362, 2017.
- [16] Bo Li, Yuchao Dai, Xuelian Cheng, Huahui Chen, Yi Lin, and Mingyi He, "Skeleton based action recognition using translation-scale invariant image mapping and multi-scale deep CNN," in *IEEE International Conference on Multimedia & Expo Workshops*, 2017, pp. 601–604.
- [17] Chao Li, Qiaoyong Zhong, Di Xie, and Shiliang Pu, "Skeleton-based action recognition with convolutional neural networks," in *IEEE International Conference on Multimedia & Expo Workshops*, 2017, pp. 597–600.
- [18] Thomas N Kipf and Max Welling, "Semi-supervised classification with graph convolutional networks," *International Conference on Learning Representations*, 2017.
- [19] Sijie Yan, Yuanjun Xiong, and Dahua Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *AAAI Conference on Artificial Intelligence*, 2018, pp. 7444–7452.
- [20] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu, "Two-stream adaptive graph convolutional networks for skeleton-based action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12026–12035.
- [21] Maosen Li, Siheng Chen, Xu Chen, Ya Zhang, Yanfeng Wang, and Qi Tian, "Actional-structural graph convolutional networks for skeleton-based action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3595–3603.
- [22] Yansong Tang, Yi Tian, Jiwen Lu, Peiyang Li, and Jie Zhou, "Deep progressive reinforcement learning for skeleton-based action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5323–5332.
- [23] Negar Heidari and Alexandros Iosifidis, "Temporal Attention-Augmented Graph Convolutional Network for Efficient Skeleton-Based Human Action Recognition," in *International Conference on Pattern Recognition*, 2020.
- [24] Wei Peng, Xiaopeng Hong, Haoyu Chen, and Guoying Zhao, "Learning graph convolutional network for skeleton-based human action recognition by neural searching," in *AAAI Conference on Artificial Intelligence*, 2020, pp. 2669–2676.
- [25] Barret Zoph and Quoc V. Le, "Neural Architecture Search with Reinforcement Learning," in *International Conference on Learning Representations*, 2017.
- [26] Dat Thanh Tran, Serkan Kiranyaz, Moncef Gabbouj, and Alexandros Iosifidis, "Heterogeneous multilayer generalized operational perceptron," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 3, pp. 710–724, 2019.
- [27] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al., "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.
- [28] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in pytorch," in *NeurIPS Workshop*, 2017.

## **7.9 On the spatial attention in Spatio-Temporal Graph Convolutional Networks for skeleton-based human action recognition**

The appended paper follows.

# On the spatial attention in Spatio-Temporal Graph Convolutional Networks for skeleton-based human action recognition

Negar Heidari and Alexandros Iosifidis

Department of Electrical and Computer Engineering, Aarhus University, Denmark

Emails: {negar.heidari, alexandros.iosifidis}@eng.au.dk

**Abstract**—Graph convolutional networks (GCNs) achieved promising performance in skeleton-based human action recognition by modeling a sequence of skeletons as a spatio-temporal graph. Most of the recently proposed GCN-based methods improve the performance by learning the graph structure at each layer of the network using a spatial attention applied on a predefined graph Adjacency matrix that is optimized jointly with model’s parameters in an end-to-end manner. In this paper, we analyze the spatial attention used in spatio-temporal GCN layers and propose a symmetric spatial attention for better reflecting the symmetric property of the relative positions of the human body joints when executing actions. We also highlight the connection of spatio-temporal GCN layers employing additive spatial attention to bilinear layers, and we propose the spatio-temporal bilinear network (ST-BLN) which does not require the use of predefined Adjacency matrices and allows for more flexible design of the model. Experimental results show that the three models lead to effectively the same performance. Moreover, by exploiting the flexibility provided by the proposed ST-BLN, one can increase the efficiency of the model.

**Index Terms**—Graph Convolutional Network, Bilinear Network, Spatial Attention, Human Action Recognition, Spatio-Temporal Graph

## I. INTRODUCTION

Considering the availability of depth cameras and successful pose estimation toolboxes such as OpenPose [1], each action can be represented by a sequence of body poses, each of which can be represented by a skeleton. Compared to other data modalities which are used for human action recognition, such as RGB videos, depth images and optical flow, human body skeleton represents the body pose and motion in a compact graph structure which is robust to context noise and invariant to body scale, view point variations, lighting conditions and background context [2]. Thus, skeleton-based human action recognition has become of great interest in recent years. Many of the recently proposed methods either use skeletons data in conjunction with other data modalities, such as RGB images [3], or only utilize the skeleton data to efficiently extract high level features for human action recognition [4]–[6].

Many deep learning methods have been proposed recently for skeleton-based human action recognition which are mainly categorized into Recurrent Neural Network (RNN)-based, Convolutional Neural Network (CNN)-based, and Graph Convolutional Network (GCN)-based methods. RNN-based methods are the earliest methods proposed in this field which

mostly utilize Long Short-Term Memory (LSTM) networks [7] to model the temporal dynamics of the sequence of skeletons [8]–[13]. These methods represent each skeleton in a sequence as a vector which is formed by concatenated human body joints’ coordinates. The CNN-based methods [14]–[19] employ the state-of-the-art CNN methods to extract the spatial and temporal features of the sequence of skeletons and they represent each skeleton in a sequence as a pseudo-image which is formed by reorganizing the body joints’ coordinates into a 2D matrix. Since RNN-based and CNN-based methods convert the skeleton data, which has an irregular structure, into a regular sequence or grid structure, it cannot benefit from the non-Euclidean structure of the skeleton sequences.

Recently, several GCN-based methods have been proposed for skeleton-based human action recognition and they achieved state-of-the-art performance [20]–[23] by utilizing the graph structure of the skeleton data. In skeleton-based human action recognition, the temporal dynamics of each action are represented by a sequence of skeletons and each human body skeleton is modeled as a graph which encodes the spatial structure of human body joints and their natural connections. Spatio-temporal GCN (ST-GCN) method [20] is the first GCN-based method proposed for skeleton-based human action recognition which receives a sequence of skeletons as input and employs GCN layers to capture both spatial and temporal features in actions by exploiting the graph structure of input data. The performance of ST-GCN method has been improved by several methods which build on top of ST-GCN. These methods mostly try to adaptively learn the graph structure in each GCN layer in an end-to-end manner, based on the features of input data [21]–[23]. 2s-AGCN [22] learns the graph structure adaptively based on the graph joints’ similarity in the input data and also the existing physical connections in the body. GCN-NAS [23] is a neural architecture search method which explores the search space with different graph modules to improve the representational capacity of the GCN layers. Moreover, the attention mechanism has been employed in these methods in order to highlight the most important connections in body skeleton for each action class. AS-GCN [24] extended the skeleton graphs to represent both structural links and actional linked and proposed an actional-structural graph convolutional network, which has an encoder-decoder structure, to capture richer dependencies from actions. DPRL+GCNN [25] and TA-GCN [26] select the most informative skeletons in a sequence

to make the inference process more efficient.

In this paper, we analyze the attention mechanisms of existing spatio-temporal GCN networks used for human action recognition. Based on this analysis, we make two contributions. First, we argue that the attention mechanism in GCN layers using an additive formulation should lead to a symmetric attention mask, as the learned attentions corresponding to a pair of nodes should reflect the symmetric relationship of the corresponding human body joint positions in a human body pose. We propose a symmetric attention mechanism that is shown to perform on par with the original one for different structures of the model. Second, we propose a spatio-temporal bilinear layer which allows for more flexible design of the model for skeleton-based human action recognition. We show that models with spatio-temporal bilinear layers perform on par with spatio-temporal GCN networks without requiring the design of graph structures to encode relationships of the joints of the body skeletons.

The remainder of the paper is organized as follows. Section II introduces the ST-GCN method which is the background of our work. Section III describes the spatial attention used in GCN-based human action recognition and Section IV analyzes its properties and proposes an alternative symmetric spatial attention. Section V describes the spatio-temporal bilinear network, which is motivated by the analysis of the spatial attention in GCN-based human action recognition methods. The experimental results are provided in section VI and the conclusions are drawn in section VII.

## II. SPATIO-TEMPORAL GCN (ST-GCN)

In this section, the ST-GCN method [20] is introduced as the baseline of many recent GCN-based methods for skeleton-based human action recognition. This method receives a sequence of skeletons  $\mathbf{X} \in \mathbb{R}^{C^{in} \times T \times V}$  encoding the human body poses comprising an action as input, where  $C^{in}$  is the number of input channels,  $T$  is the number of skeletons in the sequence, and  $V$  is the number of joints in each skeleton. It then models the sequence of skeletons as a spatio-temporal graph and applies multiple GCN layers with spatial and temporal convolutions to extract high level features for classifying the input skeleton sequence to a set of pre-defined action classes. A spatio-temporal graph on a sequence of skeletons is denoted as  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , where  $\mathcal{V}$  is the set of the human body joints and  $\mathcal{E}$  denotes both spatial and temporal edges connecting the body joints within each skeleton and between skeletons. The spatial edges are the natural physical connections between human body joints and the temporal edges connect each body joint of a skeleton to its corresponding joint of previous and subsequent skeletons. Therefore, the spatial graph is constructed based on the human body structure in which each node can have different number of neighbors, while in the temporal graph each node has two fixed neighbors, i.e. the nodes corresponding to the same joint in the previous and next time step. Figure 1 (right) shows the spatio-temporal graph on a sequence of skeletons.

Since the body motions can be grouped to concentric and eccentric, ST-GCN employs a spatial partitioning process to

divide the neighboring set of each node into three subsets based on their spatial arrangement in a single skeleton. This partitioning process fixes the node degrees in the spatial graph and defines the partitions as: 1) the root node itself, 2) the root node's neighbors which are closer to the skeleton's center of gravity than the root node and 3) the remaining root node's neighbors that are farther from the skeleton's center of gravity. The skeleton center of gravity (COG) is denoted as the average of all skeleton joints' coordinates. In Figure 1 (left) a spatially partitioned graph is illustrated in which the nodes in different neighboring subsets (partitions) are shown with different colors and the center of gravity is shown as a red dot.

According to the partitioning process, the graph structure in each skeleton is captured by three binary Adjacency matrices, which are hereafter indexed by  $p$ , each of which encodes the structure of a neighboring node subset. The Adjacency matrix of the first partition (corresponding to the root nodes) indicates the nodes' self-connections and is set to  $\mathbf{A}_1 = \mathbf{I}_V$ . Each adjacency matrix is normalized as  $\hat{\mathbf{A}}_p = \mathbf{D}_p^{-\frac{1}{2}} \mathbf{A}_p \mathbf{D}_p^{-\frac{1}{2}}$ , where  $\mathbf{D}^{(ii)}_p = \sum_j \mathbf{A}_{(ij)}_p + \varepsilon$  denotes the degree matrix and  $\varepsilon = 0.001$  is used to avoid division with zero that can be caused by empty rows in  $\mathbf{D}^{(ii)}_p$ . The element  $\hat{\mathbf{A}}_{p,ij}$  indicates whether the vertex  $j$  is in the  $p^{th}$  neighboring subset of vertex  $i$ .

Each GCN layer updates the human body features through both spatial and temporal domains by applying spatial and temporal convolutions on the output of the previous layer. The spatial convolution is defined based on the layer-wise propagation rule of GCNs [27] and the  $l^{th}$  layer spatial convolution is given by:

$$\mathbf{H}_s^{(l)} = ReLU \left( \sum_p \left( \hat{\mathbf{A}}_p \otimes \mathbf{M}_p^{(l)} \right) \mathbf{H}^{(l-1)} \mathbf{W}_p^{(l)} \right), \quad (1)$$

where  $\otimes$  is the element-wise product of two matrices and  $\mathbf{H}^{(l-1)} \in \mathbb{R}^{C^{(l-1)} \times T^{(l-1)} \times V}$  denotes the output of the  $(l-1)^{th}$  layer which is introduced to the  $l^{th}$  layer as input. The input of the first layer is defined as  $\mathbf{H}^{(0)} = \mathbf{X}$ . Since there are three spatial graphs for each sample, the GCN's propagation rule is applied on each graph with a different weight matrix  $\mathbf{W}_p^{(l)} \in \mathbb{R}^{C^{(l)} \times C^{(l-1)}}$ , where  $C^{(l-1)}$  and  $C^{(l)}$  denote the number of channels in layers  $l-1$  and  $l$ , respectively.  $\mathbf{M}_p^{(l)} \in \mathbb{R}^{V \times V}$  is a learnable attention matrix which highlights the most important connections in a skeleton for each action. It is initialized as an all-one matrix and is element-wise multiplied to its corresponding adjacency matrix of each graph partition. The spatial convolution is in practice a 2D convolution operation which performs  $C^{(l)}$  convolutions with filters of size  $C^{(l-1)} \times 1 \times 1$  on the input tensor and the resulting output is a tensor of size  $C^{(l)} \times T^{(l-1)} \times V$  which is multiplied with the masked attention-based adjacency matrix  $\hat{\mathbf{A}}_p \otimes \mathbf{M}_p^{(l)}$  on the last dimension  $V$ .

To benefit from the motions taking place in each action for label prediction, the output of the spatial convolution  $\mathbf{H}_s^{(l)}$  is introduced to the temporal convolution block which is also a 2D convolution that applies  $C^{(l)}$  convolutions with filters of size  $C^{(l)} \times k \times 1$  to capture the temporal dynamics

in the sequence of skeletons by propagating the information through the time domain while keeping the number of channels unchanged. Based on the filter size  $k$ , the temporal convolution propagates the features through  $k$  consecutive skeletons and reduces the number of skeletons or keeps it unchanged depending on the stride and padding size. Accordingly, the output of temporal convolution is of size  $C^{(l)} \times T^{(l)} \times V$  in which  $T^{(l)}$  denotes the temporal dimension of sequence in layer  $l$ . In ST-GCN method, the kernel size for temporal convolution in each layer is set to  $k = 9$ . The extracted spatio-temporal features of the last ST-GCN layer are introduced to a fully connected classification layer which is equipped by a global average pooling block and a SoftMax activation function. The entire model is trained in an end-to-end manner using Backpropagation.

### III. THE ROLE OF SPATIAL ATTENTION IN GCN-BASED HUMAN ACTION RECOGNITION

One of the main drawbacks of ST-GCN method which is addressed by more recently proposed methods, such as 2s-AGCN [22], is that it uses a fixed graph structure which is heuristically predefined and represents the natural physical connections between the body joints in a skeleton. The attention mechanism in ST-GCN, by using an element-wise multiplication between the Adjacency matrix and the learnable attention mask, can only highlight or diminish existing connections between the body joints which is not guaranteed to be optimal for action classification task. For some actions such as “touching head”, the connection between the hand and head is important for recognizing the action, while this connection does not exist naturally in the body and, thus, is not considered in the predefined graph structure. Therefore, 2s-AGCN defines the spatial convolution as follows:

$$\mathbf{H}_s^{(l)} = \text{ReLU} \left( \sum_p \left( \hat{\mathbf{A}}_p + \mathbf{M}_p^{(l)} \right) \mathbf{H}^{(l-1)} \mathbf{W}_p^{(l)} \right), \quad (2)$$

where  $\mathbf{M}_p^{(l)}$  is a learnable attention matrix added to the graph in order to both highlight/diminish existing connections between the skeleton joints and also add potentially important connections between the disconnected ones. This matrix is initialized by zeros and it is learned in an end-to-end manner along with the other model’s parameters. In other words, the spatial graph is only initialized by the skeleton’s structure, and it is updated adaptively by an attention matrix whose parameters are learned in the training process. Comparing the attention-based Adjacency matrices of ST-GCN and 2s-AGCN,  $\hat{\mathbf{A}}_p \otimes \mathbf{M}_p^{(l)}$  and  $\hat{\mathbf{A}}_p + \mathbf{M}_p^{(l)}$  respectively, it can be seen that the former is guaranteed to have a structure encoding the natural connections between the human body joints, while the latter corresponds to a matrix encoding a fully-connected graph structure.

### IV. SYMMETRIC SPATIAL ATTENTION FOR GCN-BASED HUMAN ACTION RECOGNITION

We focus on the properties of the attention-based Adjacency matrix  $\hat{\mathbf{A}}_p + \mathbf{M}_p^{(l)}$ , as it has been shown to improve action

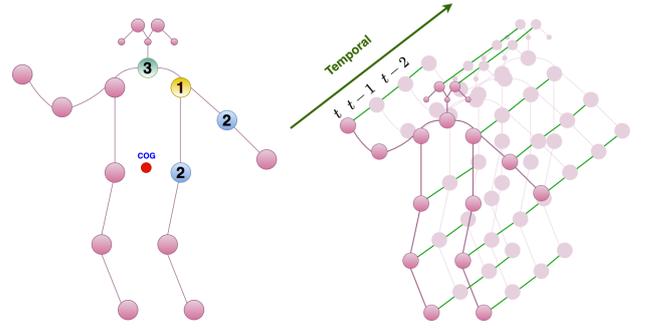


Fig. 1: Illustration of an example Spatio-temporal graph (right), and the neighboring subsets in spatial partitioning process in different colors (left).

classification performance compared to the one given by  $\hat{\mathbf{A}}_p \otimes \mathbf{M}_p^{(l)}$ . It can be shown that our analysis also holds for the latter case. Since the matrix  $\mathbf{M}_p^{(l)}$  is optimized in an end-to-end manner jointly with the parameters of the entire network without imposing any constraints, as detailed in Eq. (2), its final form will be an asymmetric matrix containing both positive and negative values. This form of the attention mask can be qualitatively explained by considering that the attention put by a graph node  $v_i$  to another graph node  $v_j$  can be freely optimized and can differ from the attention put by  $v_j$  to  $v_i$ . However, considering that the nodes of the graph correspond to human body skeleton joints and during action execution their relative positions are symmetric, we would expect that their pair-wise attentions should be the same. In order to enforce this property in the optimization process of the attention-based Adjacency matrix we define the attention mask to be a symmetric matrix, i.e.  $\mathbf{M}_p^{(l)} = \mathbf{L}_p^{(l)} \mathbf{L}_p^{(l)T}$ . Accordingly, we define the spatial convolution as follows:

$$\mathbf{H}_s^{(l)} = \text{ReLU} \left( \sum_p \left( \hat{\mathbf{A}}_p + \mathbf{L}_p^{(l)} \mathbf{L}_p^{(l)T} \right) \mathbf{H}^{(l-1)} \mathbf{W}_p^{(l)} \right). \quad (3)$$

While one can select  $\mathbf{M}_p^{(l)}$  to be of low rank by selecting  $\mathbf{L}_p^{(l)} \in \mathbb{R}^{V \times q}$  with  $q < V$ , we do not set any further restrictions to  $\mathbf{M}_p^{(l)}$ , except of being symmetric. Once the training process ends, the matrix  $\mathbf{M}_p^{(l)}$  is calculated and used for inference, thus, the space and time complexities remain the same as in the case of using Eq. (2). Here we should note that, since the normalized Adjacency matrices  $\hat{\mathbf{A}}_p$  are designed to be asymmetric, the final attention-based Adjacency matrices will be asymmetric too.

### V. SPATIO-TEMPORAL BILINEAR NETWORK

Considering the optimization of the attention-based Adjacency matrices based on the definition of the spatial convolution of ST-GCN in Eqs. (2) and (3), it can be seen that the addition of the normalized Adjacency matrix  $\hat{\mathbf{A}}_p$  to the attention mask serves as an initialization strategy, while the final form of the attention-based Adjacency matrix depends primarily on the learned values of the mask. This means that, after the first few training epochs, the GCN blocks of the ST-GCN using an additive attention mask do not follow the graph

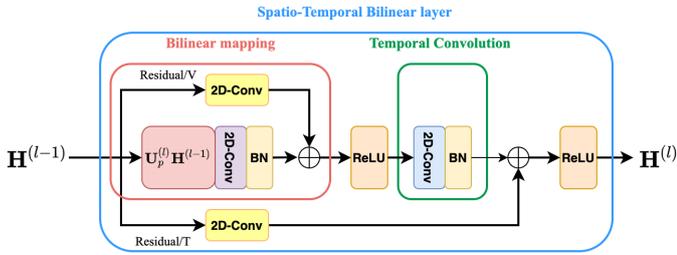


Fig. 2: Illustration of spatio-temporal bilinear layer  $l$  receiving as input  $\mathbf{H}^{(l-1)}$  of size  $C^{(l-1)} \times T^{(l-1)} \times V^{(l-1)}$  providing an output  $\mathbf{H}^{(l)}$  of size  $C^{(l)} \times T^{(l)} \times V^{(l)}$ . 2D-Conv blocks correspond to standard 2D convolutions. The 2D-Conv block in bilinear mapping transforms the data by applying  $C^{(l)}$  convolutions with filters of size  $C^{(l-1)} \times 1 \times 1$  to change the number of channels. The 2D-Conv block in temporal convolution keeps the number of channels unchanged and applies  $C^{(l)}$  filters of size  $C^{(l)} \times k \times 1$  in order to aggregate the features in temporal domain. The residual connections also apply 2D convolutions on the layer’s input to have the same dimension as the layer’s output. Residual/V indicates the residual connection which adds the input to the output of bilinear mapping, and Residual/T denotes the residual connection which adds the layer’s input to the output of temporal convolution. BN and ReLU represent the batch-normalization and ReLU activation function, respectively.

structure anymore, but they are rather equivalent to bilinear layers [28]–[30]. That is, Eqs. (2) and (3) can take the form:

$$\mathbf{H}_s^{(l)} = \text{ReLU} \left( \sum_p \mathbf{U}_p^{(l)} \mathbf{H}^{(l-1)} \mathbf{W}_p^{(l)} \right), \quad (4)$$

where  $\mathbf{U}_p^{(l)} \in \mathbb{R}^{V^{(l)} \times V^{(l-1)}}$  is a learnable matrix which is optimized in an end-to-end manner jointly with the parameters of the entire network. A specific setting of the matrix  $\mathbf{U}_p^{(l)}$  for which  $V^{(l-1)} = V^{(l)} = V$  corresponds to the attention-based Adjacency matrix in Eqs. (2) and (3) receiving as input the representations for the  $V$  human body joints at layer  $l$  and transforming them by applying a data transformation using  $\mathbf{W}_p^{(l)}$  and combining information of neighboring nodes according to the connectivity in  $\mathbf{U}_p^{(l)}$ . However, by using the bilinear layer definition in Eq. (4) one can freely decide on the dimensions of the transformation to be applied using  $\mathbf{U}_p^{(l)}$ . That is, considering that in the first layer of the Spatio-Temporal Bilinear network the matrix  $\mathbf{U}_p^{(1)} \in \mathbb{R}^{V^{(1)} \times V}$  performs a transformation in the mode of the input  $\mathbf{H}^{(0)}$  corresponding to the human body skeleton joints, selection of  $V^{(1)} < V$  will lead to aggregation of joints’ information to create a new set of  $V^{(1)}$  nodes. On the other hand, selection of  $V^{(1)} > V$  will lead to the creation of new nodes to be processed by the second layer. A similar explanation can be given to the selection of the values  $V^{(l)}$  for layer  $l$ . A special case in this setting is the one where  $V^{(l)} = 1$ , leading to the creation of one node encoding information of the entire input skeleton data.

Similar to ST-GCN, a spatio-temporal bilinear layer is

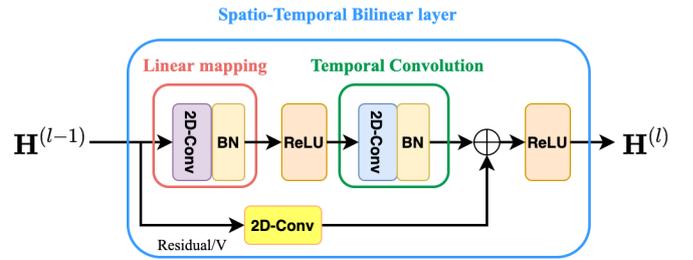


Fig. 3: Illustration of spatio-temporal bilinear layer  $l$  when  $V^{(l-1)} = V^{(l)} = 1$ .

equipped with a batch normalization, residual connections and ReLU activation function. The structure of a bilinear layer is shown in Figure. 2. The layer has two residual connections to stabilize the model by summing up the layer’s input with the output of bilinear mapping and the second one adds the layer’s input to the output of temporal convolution. We can distinguish two cases for the residual connections. When  $V^{(l-1)} = V^{(l)}$ , the number of dimensions of the input tensor  $\mathbf{H}^{(l-1)}$  to layer  $l$  changes by applying the transformation in one of its modes using  $\mathbf{W}_p^{(l)}$ , and the residual connections need to perform a 2D convolution with  $C^{(l)}$  filters of size  $C^{(l-1)} \times 1 \times 1$  on the layer’s input to have the same channel dimension as the layer’s output  $\mathbf{H}^{(l)}$ . When  $V^{(l-1)} \neq V^{(l)}$ , the number of dimensions of the input tensor  $\mathbf{H}^{(l-1)}$  to layer  $l$  changes by applying the transformations in both of its modes using  $\mathbf{W}_p^{(l)}$  and  $\mathbf{U}_p^{(l)}$ , and the residual connections need to perform a 2D convolution with  $C^{(l)} \cdot V^{(l)}$  filters of size  $C^{(l-1)} \times 1 \times V^{(l-1)}$  on the layer’s input. The resulting tensor is then reshaped to a tensor of  $C^{(l)} \times T^{(l)} \times V^{(l)}$  to have the same dimension as the layer’s output  $\mathbf{H}^{(l)}$ .

If at layer  $l - 1$  a value of  $V^{(l-1)} = 1$  is used, the input  $\mathbf{H}^{(l-1)}$  to layer  $l$  becomes of size  $C^{(l-1)} \times T^{(l-1)} \times 1$ . In that case there is no need of using  $\mathbf{U}_p^{(l)} \in \mathbb{R}$  as the scaling factor that would be learned by using it can be absorbed in  $\mathbf{W}_p^{(l)}$ . Thus, the bilinear mapping can be replaced with an equivalent linear mapping. Moreover, in this case there is no need of using a residual connection for the linear mapping block. The architecture of such a spatio-temporal bilinear layer is shown in Figure 3.

Similar to the ST-GCN method, the output of the last spatio-temporal bilinear layer is passed to a global average pooling layer to leading to 256-dimensional features for each sequence and the resulting vector is introduced to a fully connected classification layer which is equipped with a SoftMax activation function. The entire model is trained in an end-to-end manner using Backpropagation to optimize the objective function defined on the output using the classification loss.

## VI. EXPERIMENTS

We conducted two sets of experiments on NTU-RGB+D dataset [10] which is the largest indoor-captured action recognition dataset and it is widely used for evaluating the skeleton-based human action recognition methods. It consists of 56, 880 action video clips from 60 different human action classes and

Attention/Mapping	CS(%)	CV(%)
$\hat{\mathbf{A}}_p + \mathbf{M}_p^{(l)}$	85.29	93.78
$\hat{\mathbf{A}}_p + \mathbf{L}_p^{(l)} \mathbf{L}_p^{(l)T}$	87.14	93.86
$\mathbf{U}_p^{(l)}$	85.71	93.80

TABLE I: Comparison of classification accuracy of spatio-temporal networks equipped with asymmetric and symmetric spatial attentions, and the bilinear mappings on the test set of CV benchmark of NTU-RGB+D dataset. The models are trained using joints data.

#Layers	$\hat{\mathbf{A}}_p + \mathbf{M}_p^{(l)}$	$\hat{\mathbf{A}}_p + \mathbf{L}_p^{(l)} \mathbf{L}_p^{(l)T}$	$\mathbf{U}_p^{(l)}$
1	77.09	77.16	78.39
2	87.53	87.00	88.14
3	89.90	89.93	89.13
4	90.58	90.80	90.51
5	91.96	92.22	92.15
6	92.89	93.15	92.40
7	93.37	93.37	93.51
8	93.51	93.95	93.74
9	93.73	93.76	93.52
10	93.78	93.86	93.80

TABLE II: Comparison of classification accuracy of spatio-temporal networks equipped with asymmetric and symmetric spatial attentions, and the bilinear mappings on the test set of CV benchmark of NTU-RGB+D dataset for different number of layers. The models use the same layer sizes as those in [22] and are trained using joints data.

each clip is captured by 3 cameras from 3 different views. Each action clip is used to extract a sequence of 300 skeletons, each of which is represented by the 3D coordinates of 25 body joints. This leads to each action video clip being represented as a  $3 \times 300 \times 25$  tensor. Two benchmarks are defined in [10], i.e. cross view (CV) and cross-subject (CS), and a train-test split for each benchmark is also provided. We use the same experimental protocols and data splits as in [10]. The CV benchmark contains 37,920 training samples captured by cameras two and three and 18,960 test samples captured by the first camera. In CS benchmark, the training set contains 40,320 samples and test set contains 16,560 samples while the actors in training and test set are different.

The experiments are conducted on PyTorch deep learning framework [31] with 4 GRX 1080-ti GPUs. We used the same experimental settings and network architectures as in 2s-AGCN [22]. The model is trained for 50 epochs with SGD optimizer and cross entropy loss function. The mini-batch size is set to 64 and the learning rate is initialized to 0.1 and it is divided by 10 at epochs 30 and 40.

In the first set of experiments, we compared the performance of spatio-temporal networks using GCN layers equipped with attentions in Eqs. (2) and (3) and using bilinear layer as defined in Eq. (4). The performance of the three models is reported in Table I. As can be seen, the three networks achieved very similar performance. This indicates that there is no idfference in using the predefined graph structure encoded by the matrices  $\hat{\mathbf{A}}_p$  in Eqs. (2) and (3) for initializing the mapping in the bilinear layer (4). To further analyze the effect of the models'

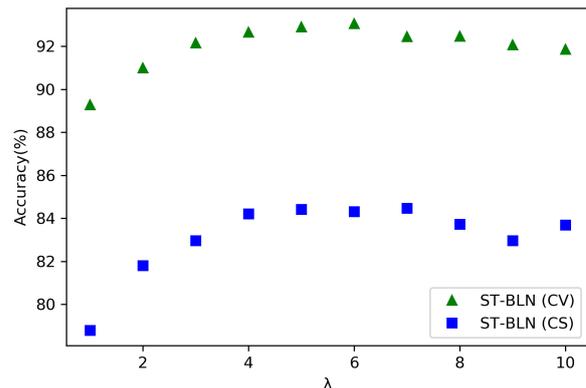


Fig. 4: Performance of ST-BLN model in terms of classification accuracy, when joints' features are aggregated in different layers  $\lambda$  of the network.

size and the differences in performance achieved by the three types of spatio-temporal networks, we conducted multiple experiments using different number of layers and the results are reported in Table II. As can be seen, the performance of the three types of networks for all network sizes are effectively the same. This confirms our observations related to the role of the predefined graph structure used in the spatial attention of spatio-temporal networks for skeleton-based action recognition. We can conclude that the combining information related to the nodes of the human body skeleton graph can be randomly initialized and optimized jointly with the other network parameters, and that the use of predefined skeleton graph structures is not necessary for recognizing the human actions using skeleton data.

For completeness, we also provide in Table III the performance of state-of-the-art methods in skeleton-based human action recognition on the two benchmarks of the NTU-RGB+D dataset. Since most of the state-of-the-art methods, like 2s-AGCN, GCN-NAS, AS-GCN and 2s-TA-GCN, train their model with two or more data streams, we also report the performance achieved by the spation-temporal bilinear network using two streams. That is, the 2s-ST-BLN model is formed by two streams, the first of which receives as input the joints data and the second one receives as input the bone data. The two streams process their inputs and the predicted SoftMax scores of the two streams are fused to obtain the final classification outcome. The results show that ST-BLN and 2s-ST-BLN perform on par with other state-of-the-art methods while they do not require a predefined graph structure to encode the relationships between the joints (and bones) of the human body skeleton. Similar to the GCN-based methods, the proposed method also outperforms the RNN-based and CNN-based methods with a large margin.

To evaluate the need of combining information in the dimension of human body skeleton joints, expressed either as spatial attention of the form in Eq. (2) and as bilinear mapping of the form in Eq. (4), or if it is adequate the employ linear mappings effectively leading to neural layers combining

Method	CS(%)	CV(%)	#Streams
HBRNN [8]	59.1	64.0	5
Deep LSTM [10]	60.7	67.3	1
ST-LSTM [9]	69.2	77.7	1
STA-LSTM [11]	73.4	81.2	1
VA-LSTM [12]	79.2	87.7	1
ARRN-LSTM [13]	80.7	88.8	2
2s-3DCNN [14]	66.8	72.6	2
TCN [15]	74.3	83.1	1
Clips+CNN+MTLN [16]	79.6	84.8	1
Synthesized CNN [17]	80.0	87.2	1
3scale ResNet152 [18]	85.0	92.3	1
CNN+Motion+Trans [19]	83.2	89.3	2
ST-GCN [20]	81.5	88.3	1
DPRL+GCNN [25]	83.5	89.8	1
TA-GCN [26]	87.97	94.2	1
AS-GCN [24]	86.8	94.2	2
2s-AGCN [22]	88.5	95.1	2
2s-TA-GCN [26]	88.5	95.1	2
GCN-NAS [23]	89.4	95.7	2
<b>ST-BLN</b>	<b>85.71</b>	<b>93.80</b>	<b>1</b>
<b>2s-ST-BLN</b>	<b>87.8</b>	<b>95.1</b>	<b>2</b>

TABLE III: Classification accuracy comparison of spatio-temporal bilinear model with state-of-the-art GCN-based methods on the test set of NTU-RGB+D dataset.

a Multilayer Perceptron block with a Temporal Convolution block, we conducted a second set of experiments. In this set of experiments, we used a 10-layer spatio-temporal bilinear network formed by the same data transformation sizes as in the first set of our experiments, but instead of using bilinear mappings with  $V^{(l)} = V$  for all 10 layers of the model, we use  $V^{(l)} = V$ ,  $l = 1, \dots, \lambda - 1$  and  $V^{(l)} = 1$ ,  $l = \lambda, \dots, 10$ . That is, the first  $\lambda - 1$  layers of the model use Spatio-Temporal Bilinear layers as the one illustrated in Figure 2 with values  $V^{(l)} = V$ . At layer  $\lambda$  a Spatio-Temporal Bilinear layer in the form of Figure 2 is used with  $V^{(\lambda)} = 1$ , while the remaining layers take the form of a Spatio-Temporal Bilinear layer as the one illustrated in Figure 3. We applied experiments using both the CV and CS benchmarks of the NTU-RGB+D dataset using the joints data, and the results in terms of classification accuracy and number of FLOPs are indicated in Figures 4 and 5, respectively. As can be seen in Figure 4, on the CV benchmark the best performance is equal to 93.06% and is obtained for a value of  $\lambda = 6$ . On the CS benchmark, the best performance is obtained for a value of  $\lambda = 7$  and is equal to 84.47%. These performance values are on par with the results reported in Table I, achieved by using the Spatio-Temporal Bilinear/GCN layers using  $V^{(l)} = V$ ,  $l = 1, \dots, 10$ . Observing the performance values achieved by the different networks in Figure 4, we can conclude that combining information in the dimension of human body skeleton joints at early layers of the network contributes in the performance achieved by the network, while this is not the case for deeper layers in the network.

By observing the number of Floating Point Operations needed to evaluate an input skeleton sequence for different choices of  $\lambda$  in Figure 5, we can see that one can increase

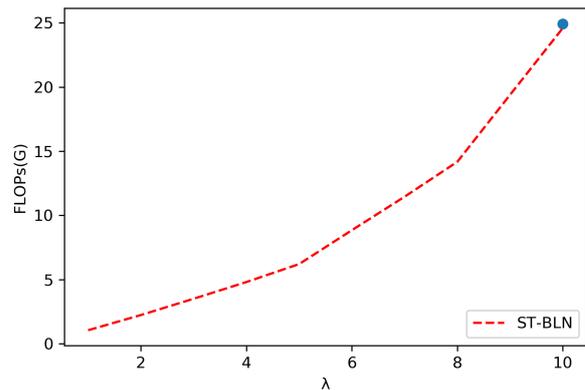


Fig. 5: Floating Point Operations of the ST-BLN model when joints' features are aggregated in different layers  $\lambda$  of the network. The case of a ST-BLN with  $V^{(l)} = V$ ,  $l = 1, \dots, 10$  is illustrated with a dot.

efficiency of the model by using lower values of  $\lambda$ . This is due to that for layers beyond the  $\lambda$ -th layer of the network a smaller number of parameters is needed, as indicated by comparing the two layers illustrated in Figures 2 and 3. Considering that the number of model's parameters at the deeper layers is higher compared to the first layers, it can be seen that a considerable improvement in the model's efficiency can be achieved. Overall, a network using a value of  $\lambda = 6$  ( $\lambda = 7$ ) operates  $\times 2.78$  ( $\times 2.14$ ) faster compared to a network with  $\lambda = 10$ . As expected, the numbers of FLOPs for a network with  $\lambda = 10$  and a ST-BLN with all layers using values  $V^{(l)} = V$  are very similar, i.e. 24.59G and 24.93G, respectively.

## VII. CONCLUSION

In this paper, we studied the properties of the spatial attention used in ST-GCN methods for skeleton-based human action recognition. We made two observations, the first being that the spatial attention used in these methods leads to an asymmetric attention matrix. Our second observation is that the human-expert designed normalized Adjacency matrices used in models with additive attention serve only as an initialization process, and do not really affect parameters of the network for combining information in the dimension of human body skeleton joints used for inference. Based on the first observation, we proposed a symmetric spatial attention that can be directly incorporated in the existing ST-GCN methods without affecting the overall number of computations during inference. Based on the second observation, we proposed the Spatio-Temporal Bilinear Network (ST-BLN) which does not require the use of a predefined Adjacency matrix and allows for more flexible design of the model for skeleton-based human action recognition. Extensive experimental analysis shows that the three models lead to effectively the same performance. Moreover, by exploiting the flexibility provided by the proposed ST-BLN, one can increase the model's efficiency

by a factor of more than  $\times 2$  while preserving the performance levels of the original method.

#### ACKNOWLEDGEMENT

This work was supported by the European Union’s Horizon 2020 Research and Innovation Action Program under Grant 871449 (OpenDR). This publication reflects the authors’ views only. The European Commission is not responsible for any use that may be made of the information it contains.

#### REFERENCES

- [1] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, “Realtime multi-person 2D pose estimation using part affinity fields,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [2] F. Han, B. Reily, W. Hoff, and H. Zhang, “Space-time representation of people based on 3D skeletal data: A review,” *Computer Vision and Image Understanding*, vol. 158, pp. 85–105, 2017.
- [3] A. Franco, A. Magnani, and D. Maio, “A multimodal approach for human activity recognition based on skeleton and rgb data,” *Pattern Recognition Letters*, vol. 131, pp. 293–299, 2020.
- [4] D. C. Luvizon, H. Tabia, and D. Picard, “Learning features combination for human action recognition from skeleton sequences,” *Pattern Recognition Letters*, vol. 99, pp. 13–20, 2017.
- [5] A. Saggese, N. Strisciuglio, M. Vento, and N. Petkov, “Learning skeleton representations for human action recognition,” *Pattern Recognition Letters*, vol. 118, pp. 23–31, 2019.
- [6] M. Li and H. Leung, “Graph-based approach for 3d human skeletal action recognition,” *Pattern Recognition Letters*, vol. 87, pp. 195–202, 2017.
- [7] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, “LSTM: A search space odyssey,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2016.
- [8] Y. Du, W. Wang, and L. Wang, “Hierarchical recurrent neural network for skeleton based action recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.
- [9] J. Liu, A. Shahroudy, D. Xu, and G. Wang, “Spatio-temporal LSTM with trust gates for 3D human action recognition,” in *European Conference on Computer Vision*, 2016.
- [10] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, “NTU RGB+D: A large scale dataset for 3D human activity analysis,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [11] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu, “An end-to-end spatio-temporal attention model for human action recognition from skeleton data,” in *AAAI Conference on Artificial Intelligence*, 2017.
- [12] P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue, and N. Zheng, “View adaptive recurrent neural networks for high performance human action recognition from skeleton data,” in *IEEE International Conference on Computer Vision*, 2017.
- [13] L. Li, W. Zheng, Z. Zhang, Y. Huang, and L. Wang, “Relational network for skeleton-based action recognition,” in *IEEE International Conference on Multimedia & Expo*, 2019.
- [14] H. Liu, J. Tu, and M. Liu, “Two-stream 3d convolutional neural network for skeleton-based action recognition,” *arXiv preprint arXiv:1705.08106*, 2017.
- [15] T. S. Kim and A. Reiter, “Interpretable 3D human action analysis with temporal convolutional networks,” in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017.
- [16] Q. Ke, M. Bennamoun, S. An, F. Sohel, and F. Boussaid, “A new representation of skeleton sequences for 3D action recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [17] M. Liu, H. Liu, and C. Chen, “Enhanced skeleton visualization for view invariant human action recognition,” *Pattern Recognition*, vol. 68, pp. 346–362, 2017.
- [18] B. Li, Y. Dai, X. Cheng, H. Chen, Y. Lin, and M. He, “Skeleton based action recognition using translation-scale invariant image mapping and multi-scale deep CNN,” in *IEEE International Conference on Multimedia & Expo Workshops*, 2017.
- [19] C. Li, Q. Zhong, D. Xie, and S. Pu, “Skeleton-based action recognition with convolutional neural networks,” in *IEEE International Conference on Multimedia & Expo Workshops*, 2017.
- [20] S. Yan, Y. Xiong, and D. Lin, “Spatial temporal graph convolutional networks for skeleton-based action recognition,” in *AAAI Conference on Artificial Intelligence*, 2018.
- [21] L. Shi, Y. Zhang, J. Cheng, and H. Lu, “Skeleton-based action recognition with directed graph neural networks,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [22] L. Shi, Y. Zhang, J. Cheng, and H. Lu, “Two-stream adaptive graph convolutional networks for skeleton-based action recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [23] W. Peng, X. Hong, H. Chen, and G. Zhao, “Learning graph convolutional network for skeleton-based human action recognition by neural searching,” in *AAAI Conference on Artificial Intelligence*, 2020.
- [24] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, and Q. Tian, “Action-structural graph convolutional networks for skeleton-based action recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [25] Y. Tang, Y. Tian, J. Lu, P. Li, and J. Zhou, “Deep progressive reinforcement learning for skeleton-based action recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
- [26] N. Heidari and A. Iosifidis, “Temporal attention-augmented graph convolutional network for efficient skeleton-based human action recognition,” in *International Conference on Pattern Recognition*, 2020.
- [27] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *International Conference on Learning Representations*, 2017.
- [28] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell, “Compact bilinear pooling,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [29] D. T. Tran, A. Iosifidis, J. Kannianen, and M. Gabbouj, “Temporal attention-augmented bilinear network for financial time-series data analysis,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 5, pp. 1407–1418, 2018.
- [30] J.-H. Kim, J. Jun, and B.-T. Zhang, “Bilinear attention networks,” in *Advances in Neural Information Processing Systems*, 2018.
- [31] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” in *Advances in Neural Information Processing Systems Workshops*, 2017.

## **7.10 Improving the performance of lightweight CNNs for binary classification using Quadratic Mutual Information regularization**

The appended paper follows.

# Improving the performance of lightweight CNNs for binary classification using Quadratic Mutual Information regularization

Maria Tzelepi<sup>a,\*</sup>, Anastasios Tefas<sup>a</sup>

<sup>a</sup>*Department of Informatics  
Aristotle University of Thessaloniki  
Thessaloniki, 54124, Greece*

---

## Abstract

In this paper, we propose regularized lightweight deep convolutional neural network models, capable of effectively operating in real-time on-drone for high-resolution video input. Furthermore, we study the impact of hinge loss against the cross entropy loss on the classification performance, mainly in binary classification problems. Finally, we propose a novel regularization method motivated by the Quadratic Mutual Information, in order to improve the generalization ability of the utilized models. Extensive experiments on various binary classification problems involved in autonomous systems are performed, indicating the effectiveness of the proposed models. The experimental evaluation on four datasets indicates that hinge loss is the optimal choice for binary classification problems, considering lightweight deep models. Finally, the effectiveness of the proposed regularizer in enhancing the generalization ability of the proposed models is also validated.

*Keywords:* Hinge Loss, Cross Entropy Loss, Binary Classification Problems, Quadratic Mutual Information, Regularizer, Lightweight Models, Real-time, Convolutional Neural Networks, Deep Learning

---

\*Corresponding author  
*Email addresses:* mtzelepi@csd.auth.gr (Maria Tzelepi), tefas@csd.auth.gr (Anastasios Tefas)

## 1. Introduction

Deep learning algorithms, [1, 2], and principally the deep Convolutional Neural Networks (CNN), [3], have been established among the most effective research directions in a wide range of computer vision tasks, [4], accomplishing outstanding performance over previous shallow models. More specifically, deep CNNs have been successfully applied in image classification [5, 6], object detection [7, 8, 9] and retrieval [10, 11, 12], visual tracking [13, 14], video captioning [15], and pose estimation [16]. Deep CNNs owe their success to the availability of large annotated datasets, and the Graphics Processing Units (GPUs) computational power and affordability. Furthermore, apart from developing successful deep models towards the aforementioned computer vision tasks, another research direction that flourishes during the recent few years while more efficient solutions are still striving to emerge, is the development of lightweight models capable of operating on devices with restricted computational resources such as mobile phones and embedded systems, [17, 18].

During the recent years, we have witnessed the successful introduction of Unmanned Aerial Vehicles (UAV), widely known as *drones*, in the media and entertainment industry. Drones have been applied in a wide spectrum of applications, ranging from entertainment to visual surveillance, rescue within the context of natural disasters [19], and medical emergencies [20], while previous practices in media production are gradually displaced due to their capability of capturing impressive aerial shots or shots of even inaccessible places. A major issue linked with the rise of drones is the demand for developing models for various computer vision tasks, able of both addressing the additional challenges of drone-captured images (such as small object size, unconstrained pose variations, occlusion), and running on-drone, that is with restricted processing power.

Thus, in this work, we propose regularized lightweight deep CNN models for various classification problems involved in autonomous systems, and more specifically, we consider binary classification problems in the context of media coverage of certain sport events (i.e. football match, bicycle race) by drones, allowing real-

time deployment for high resolution images. Furthermore, a key issue related to drone usage and autonomy is the demand for increased safety, since a drone may operate in vicinity of human crowds, and is potentially exposed to environmental hazards or unforeseeable errors that render their emergency landing inevitable.

35 Hence, we also propose lightweight CNN models for crowd detection towards crowd avoidance. Finally, since face detection constitutes a primary step towards recognition of a specific bicyclist, or football player, we also deal with face detection. Summarizing, in this work we train CNN models for bicycle, crowd, face, football player detection in the context of media coverage of certain sport events

40 by drones. Our goal is to produce semantic heatmaps [21] by e.g. predicting for each location within the captured high-resolution scene the crowd presence. That is, models with input of size either  $32 \times 32 \times 3$  or  $64 \times 64 \times 3$  which correspond to the width, height, and number of color channels of the input image, are trained. Then, test images are propagated to the network, and for every window  $32 \times 32$

45 or  $64 \times 64$  respectively the output of the network at the last convolutional layer is computed. An example of a crowd heatmap is provided in Fig. 1. Furthermore, the above procedure is useful in the camera control problem, [22]. That is, the semantic heatmaps for each of the aforementioned classification problems, aim to aid the algorithm for controlling the drone’s camera for cinematography tasks

50 by sending error signals. We should emphasize that the capability of handling high resolution images is extremely important for the application, since objects of interest in drone-captured images are of extremely small size, and thus image resizing, which is used by almost all of the state-of-the-art visual content analysis models (e.g. YOLO [9], SSD [8], etc.), would further shrink the object’s size,

55 rendering the detection impossible.

Subsequently, since we deal with lightweight models which usually perform worse than the more complex models, we aim at enhancing their performance. Thus, one objective of this paper, is to extensively study the impact of hinge loss against the cross entropy loss on the classification performance of binary

60 classification problems. Additionally, we also perform experiments on multi-class datasets.

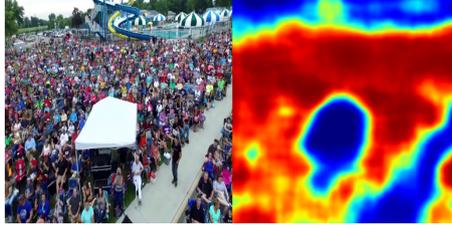


Figure 1: Crowded image and the corresponding predicted heatmap of crowd presence.

Finally, the third objective of this work is to propose a novel regularization method in order to reduce overfitting and improve the generalization ability of the utilized models. This is of considerable importance, in general, in deep learning, since neural networks are prone to overfitting due to their high capacity. In this work, we propose the so-called *Mutual Information (MI) regularizer*. The proposed regularizer is inspired by the Quadratic Mutual Information (QMI) measure [23], which is a variant of the commonly used Mutual Information, an information-theoretic measure of dependence between random variables. That is, apart from the classification loss, we propose to attach an additional optimization criterion based on the QMI. Recently, QMI reformulated to produce a kernel dimensionality reduction method under the Graph Embedding framework [24], while in [25] a Probabilistic Knowledge Transfer method proposed exploiting the QMI. We should note that the proposed regularization method is generic and can be applied in several deep learning architectures for classification purposes.

Over the past years, several regularization schemes have been proposed in order to improve weak generalization ability in neural networks, extended from common regularization methods, like the standard L1/L2 regularization which penalize large weights during the network optimization, to Dropout [26] where for each training sample, a randomly selected subset of the activations is set to zero in each epoch, and Dropconncet [27] that is a generalization of Dropout which instead of activations, sets a randomly selected subset of weights within the network to zero. Other earlier approaches include weight elimination, [28], and Bayesian methods, [29]. From another angle of view, multitask-learning [30] has

85 been employed as a way of enhancing the generalization ability of a model. For instance, techniques developed in semi-supervised learning were introduced in the deep learning domain, in [31]. That is, an unsupervised regularizer is combined with a supervised learner to perform semi-supervised learning. Furthermore, a novel CNN architecture with a SVM classifier at every hidden layer is proposed  
90 in [32]. This companion objective acts as a kind of feature regularization.

Summing up, the objective of this paper is to develop lightweight models for various classification tasks able to run in real time on-drone. We explore ways to enhance the classification performance of the lightweight models, first by investigating the effect of two widely used classification losses, that is the cross entropy  
95 and hinge losses, on the classification performance, and second by proposing a novel regularizer motivated by the QMI criterion.

The main contribution of this work can be summarized as follows:

- We propose regularized lightweight deep CNN models for various classification  
100 problems, capable of running in real-time on-drone.
- We empirically study the impact of hinge loss against cross entropy loss in binary classification problems and we argue that the hinge loss is better for binary classification problems.
- We propose a novel regularizer based on the QMI criterion in order to  
105 enhance the generalization ability of the utilized models.

The remainder of the manuscript is structured as follows. The utilized CNN architectures are described in Section 2. Section 3 presents the two compared loss functions. Subsequently, the proposed MI regularizer is presented in Section 4. The experiments, including the datasets description, the implementation details  
110 and the experimental results, are provided in Section 5. Finally, the conclusions are drawn in Section 6.

## 2. Lightweight CNN Models

In this Section, we provide the descriptions of the utilized architectures. A principal target of the utilized architectures is to permit real-time (that is about  
115 25 frames per second) deployment on-drone for high resolution images. We should emphasize that it is critical for the application to handle high resolution images, since objects in drone-captured images are of extremely small size, and thus image resizing in order to meet real-time deployment limits, would further shrink the object of interest, rendering the detection impossible. Fig. 3 highlights  
120 the demand for high resolution images. That is, an aerial image that contains bicycles (bicycles with bicyclists) is provided, Fig. 3a, and the resulting heatmaps for input of various resolutions, i.e.  $320 \times 240$ ,  $480 \times 360$ ,  $640 \times 480$ ,  $1280 \times 720$ , and  $1920 \times 1080$ , utilizing a proposed model, Figs. 3d-3h. As it shown, as the resolution increases, better performance can be accomplished. Furthermore, in  
125 Figs. 3b and 3c, the predictions for the same input, are provided, utilizing two state-of-the-art detectors which have been trained to detect among other classes, also persons and bicycles, that is YOLO v.2 [9] and Faster R-CNN [7], which operate for input  $608 \times 608$  and  $1000 \times 600$  respectively. As it is shown, both the state-of-the-art detectors perform poorly, while they also operate at much less  
130 than real-time, as it is shown below. It is also noteworthy, that SSD [8] and SSD with MobileNets [17], which operate for input  $300 \times 300$ , as well as for input  $512 \times 512$ , fail to detect any bicycle.

Thus, we utilize two architectures consisting of only five convolutional layers, by discarding the deepest layers and pruning filters of the widely used VGG-  
135 16 model [6]. That is, the first four convolutional layers of the VGG-16 model are used with pruned filters, while the last convolutional layer consists of two channels, each for a class, since we deal with binary classification problems. The first model can run in real-time on-drone for 720p ( $1280 \times 720$ ) resolution image and the second one can run in real-time for 1080p ( $1920 \times 1080$ ) resolution image.  
140 Thus, the models are abbreviated as VGG-720p and VGG-1080p, respectively, based on this attribute. Details on kernel sizes and channels of each layer of the

two architectures can be found in Table 1 and Table 2. The above descriptions of models concern input of training images of size  $32 \times 32$ . For input of size  $64 \times 64$ , using same kernels and channels, we use appropriate stride and pooling  
145 to achieve real-time deployment. Details on the utilized model architectures for both  $32 \times 32$  and  $64 \times 64$  input dimensions are depicted in Fig. 2.

The performance is evaluated on a low-power NVIDIA Jetson TX2 module with 8GB of memory, which is a state of the art GPU used for on-board drone perception. Furthermore, in order to accelerate the deployment speed and achieve  
150 real-time deployment, the TensorRT<sup>1</sup>, deep learning inference optimizer is utilized. TensorRT is a library that optimizes deep learning models providing FP32 (default) and FP16 optimizations for production deployments of various applications. In Table 3, the detection speed in terms of frames per second (fps) is provided for the two architectures and their corresponding image resolution on the NVIDIA  
155 Jetson TX2 module without the utilization of the TensorRT optimizer, with the TensorRT on the default mode, as well as with TensorRT on the FP16 mode. As it shown, TensorRT and in particular the FP16 mode significantly accelerates the proposed models, achieving detection in-real time for high-resolution images.

Note that state-of-the-art detectors run at notably fewer fps on Jetson TX2,  
160 and also for lower resolution input images. For example, SSD [8] runs at 6 fps, for input of size  $300 \times 300$ , SSD with MobileNets [17] runs at 12.4 fps for the same input, and the Faster R-CNN [7] runs at 0.9 fps. Finally, YOLO v.2 [9] runs at 10 fps for input of size  $308 \times 308$ , while it runs at 3.1 fps for input of size  $604 \times 604$  on the Jetson TX2 module. For fair comparisons, we also test the  
165 performance of YOLO v.2 utilizing TensorRT, as we have observed and it is also reported in literature [33, 34] that YOLO is faster than SSD and Faster-RCNN. Thus, utilizing the TensorRT optimizer YOLO v.2 runs at 7.8 fps for input of size  $604 \times 604$ , while further speed up is achieved with the FP16 mode up to 14.4 fps. It is noteworthy that state-of-the-art detectors like YOLO, can not achieve real-  
170 time detection on Jetson TX2, even utilizing the TensorRT optimizer, and also for

---

<sup>1</sup><https://developer.nvidia.com/tensorrt>

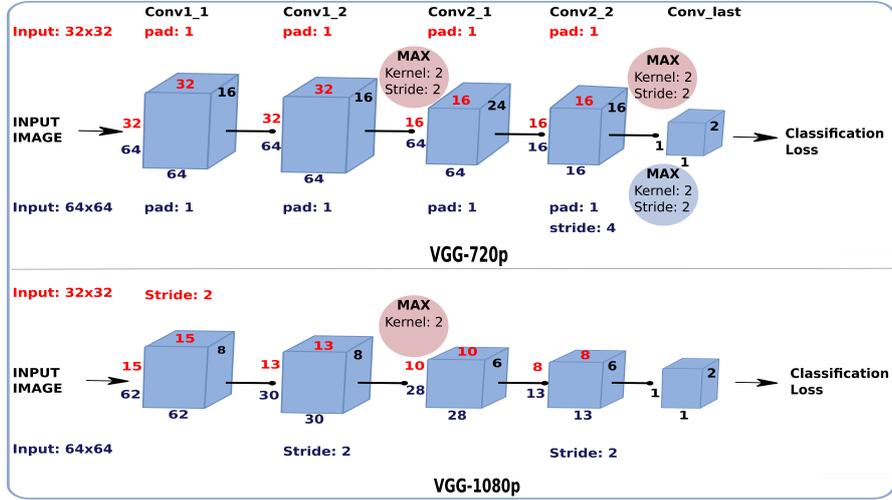


Figure 2: Model architectures: The VGG-720p model is depicted in the upper part of the figure, while the VGG-1080p model is depicted in the lower part of the figure. Details for input of size 32×32 are printed in red for both the model architectures, while details for input of size 64×64 are printed in blue.

considerably lower input resolution. Counterwise, the proposed models allow for real-time deployment utilizing TensorRT even for input resolution 1080p.

### 3. Hinge Loss Versus Cross Entropy Loss

Cross entropy loss and hinge loss functions are probably the most widely  
 175 used loss functions in pattern classification. Support Vector Machines (SVM),  
 [35], which use the hinge loss, constitute up to the present time a vivid research  
 field [36, 37]. SVMs, which are inherently binary classifiers, seek for the optimal  
 hyperplane which distinctly classifies the data samples, that is the hyperplane  
 which maximizes the margin between the two classes. Considering multi-class  
 180 classification problems several works have been proposed for formulating the  
 SVM over multiple classes [38, 39, 40]. Amongst them, the earliest one and  
 probably the most common technique is the one-against-all (or one-against-rest),  
 which builds  $N_c$  one-against-all SVM models where  $N_c$  is the number of classes.



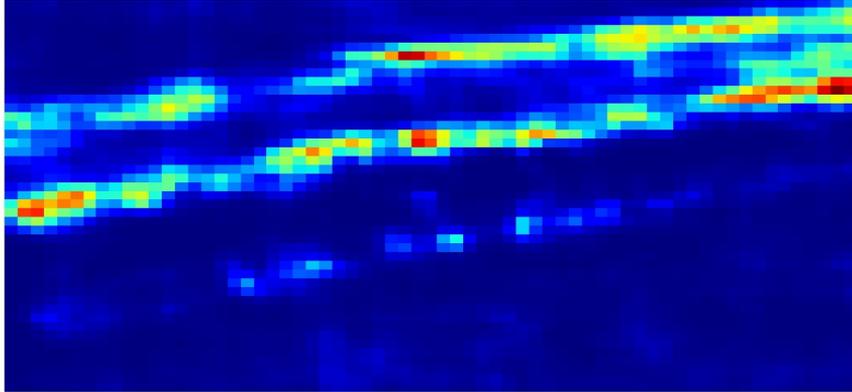
(a) Test image



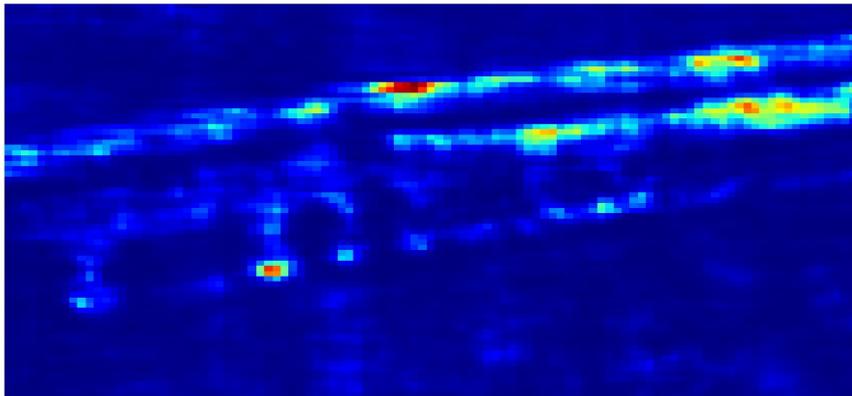
(b) YOLO v.2 Prediction for input of size 604x604



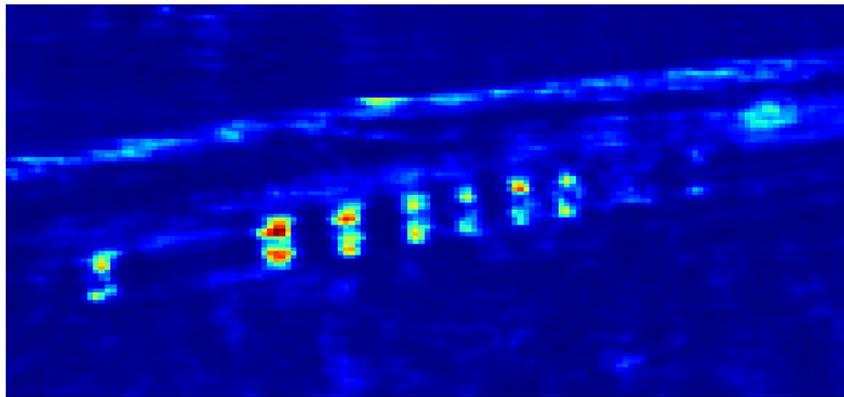
(c) Faster R-CNN Prediction for input of size 1000x600



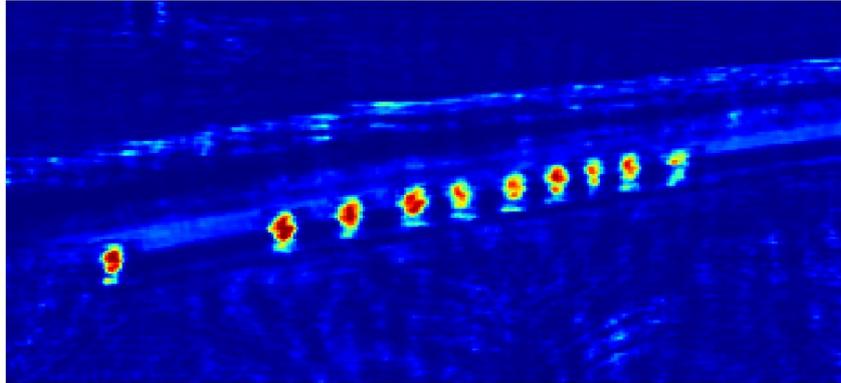
(d) Resulting heatmap for input of size  $320 \times 240$ , utilizing the proposed VGG-1080p model



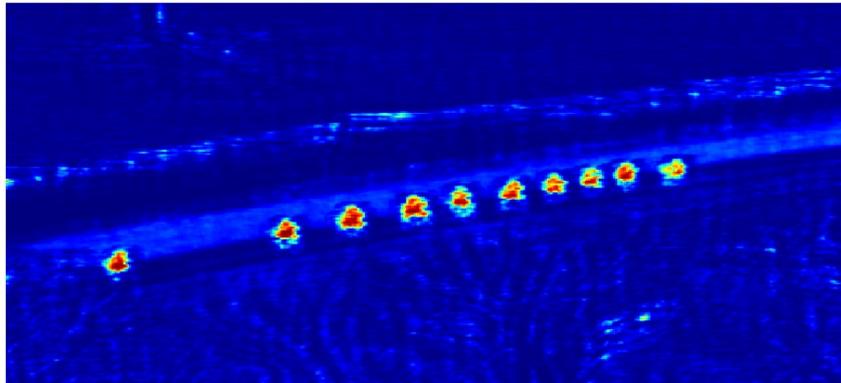
(e) Resulting heatmap for input of size  $480 \times 360$ , utilizing the proposed VGG-1080p model



(f) Resulting heatmap for input of size  $640 \times 480$ , utilizing the proposed VGG-1080p model



(g) Resulting heatmap for input of size  $1280 \times 720$ , utilizing the proposed VGG-1080p model



(h) Resulting heatmap for input of size  $1920 \times 1080$ , utilizing the proposed VGG-1080p model

Figure 3: An aerial high resolution image containing bicycles (3a), predictions utilizing the YOLO v2 and the Faster R-CNN detectors (3b)-(3c), and the resulting heatmaps for various deployment resolutions utilizing the proposed VGG-1080p model trained for bicycle detection (3d)-(3h).

Layer	Kernel	Channels
conv1_1	$3 \times 3$	16
conv1_2	$3 \times 3$	16
conv2_1	$3 \times 3$	24
conv2_2	$3 \times 3$	16
conv_last	$8 \times 8$	2

Table 1: VGG-720p

Layer	Kernel	Channels
conv1_1	$3 \times 3$	8
conv1_2	$3 \times 3$	8
conv2_1	$3 \times 3$	6
conv2_2	$3 \times 3$	6
conv_last	$8 \times 8$	2

Table 2: VGG-1080p

input	Model	Jetson TX2	TensorRT-FP32	TensorRT-FP16
32×32	VGG-720p	10.1	18.1	26.3
32×32	VGG-1080p	12.3	16.9	25.7
64×64	VGG-720p	8.7	16.6	25
64×64	VGG-1080p	8.8	18.5	25.6

Table 3: Speed (fps)

For a set of  $N$  input images  $\mathcal{X} = \{\mathbf{X}_i, i = 1, \dots, N\}$  we consider the corresponding scores with respect to each class,  $\mathbf{y}_i^{last} \in \mathfrak{R}^{N_c \times 1}$ . In the typical case the classification layer is implemented using a fully connected layer - with number of nodes equal to the number of classes - and the output is fed to the loss layer. In our case, since the objective is to develop lightweight models, and hence we propose fully convolutional architectures, instead of a fully connected layer, there is a convolutional layer with number of channels equal to the number of classes and kernel with receptive field equal to the whole input volume.

Then, the hinge loss is defined as:

$$L_{hinge} = \frac{1}{N} \sum_{i=1}^N \sum_{j=1}^{N_c} \max(0, 1 - \delta\{c_i = j\} y_{i,j}^{last}), \quad (1)$$

where  $c_i \in [1, \dots, N_c]$  indicates the correct class among the  $N_c$  classes,  $y_{i,j}^{last}$  indi-

cates the score with respect to the  $j$ -th class for the  $i$ -th image, and

$$\delta\{condition\} = \begin{cases} 1 & , \text{if condition} \\ -1 & , \text{otherwise} \end{cases}$$

In this work, we deal with binary classification problems. That is,  $N_c = 2$ .

Cross entropy loss or softmax classifier, is extensively used in deep learning architectures [5, 6, 41], providing an intuitive output of normalized class probabilities. Instead of computing scores for each class, like the SVM classifier, the softmax classifier computes the scores for each class, and then applies the softmax function [42] to transform them to a vector of values between zero and one that sum to one, in order to be interpreted as class probabilities. Finally, the classification process is realized using the cross entropy loss function.

The cross entropy loss is defined as:

$$L_{cross\_entropy} = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^{N_c} l_{i,j} \log(p_{i,j}), \quad (2)$$

where  $N_c$  is the number of classes,  $l_{i,j} \in \{0, 1\}$  is a binary indicator that takes the value 1 if the the sample  $i$  belongs to class  $j$ , and  $p_{i,j}$  is the predicted softmax probability the sample  $i$  to belong to class  $j$ . For a two-class classification problem the cross entropy loss can be calculated as:

$$L_{cross\_entropy\_binary} = -\frac{1}{N} \sum_{i=1}^N (l_i \log(p_i) + (1 - l_i) \log(1 - p_i)) \quad (3)$$

To the best of our knowledge, there is no other previous work extensively investigating the impact of hinge loss against cross entropy loss on the classification accuracy, with special emphasis to binary classification problems. Surveying the relevant literature, we observe that the cross entropy loss is widely used in deep CNNs for dealing with multi-class classification problems [5, 6, 41].

On the other hand, in [43] the author first proposes to replace the softmax loss layer (i.e. cross entropy loss), with a linear SVM layer. Particularly, the L2-SVM objective [44] is utilized instead of the standard hinge loss. Experimental results on MNIST-10 and CIFAR-10 datasets show that for some deep neural models, the linear SVM layer is beneficial over the softmax loss one.

In [45] the authors provide comparisons among various classification losses, including the cross entropy and hinge losses, for multi-class classification problems. The authors conclude that depending on the application of the deep model, losses other than cross entropy loss are preferable. Subsequently, in [32], which is  
220 a work with a different goal where a new CNN architecture with a SVM classifier at each hidden layer is proposed, we also observe that a CNN with SVM loss layer outperforms the CNN with softmax loss layer in the MNIST-10 dataset.

Finally, studying the work presented in [46], and particularly in the Feature Analysis section where an analysis of the discriminative information of each  
225 layer is provided, apart from the stated observations on the importance of the model's depth, some potentially interesting remarks arise. That is, we first observe that in a dataset with comparatively few classes the SVM classifier outperforms the softmax classifier, while in a similar dataset with much more classes, the softmax classifier performs better. Furthermore, we see that in the first dataset  
230 of fewer classes, the difference between the SVM classifier over the softmax is notably bigger in the less deep layer. Thus, this also enhances our motivation to investigate the efficiency of hinge loss as compared to the cross entropy loss, since a major objective of this work is to provide lightweight models with improved performance.

In order to obtain real-time performance someone has to severely decrease  
235 the number of layers and the number of filters. Thus, the resulting lightweight models are weaker than the heavier ones in terms of performance. In order to improve their performance we should exploit the available losses and possible regularizers that fit better to the specific tasks, which is binary classification with  
240 limited computational resources.

#### 4. The Proposed MI Regularizer

In this paper, we propose a novel regularizer motivated by the Quadratic Mutual Information [23]. Apart from the classification loss, we propose a regularization loss derived from the so-called information potentials of the QMI. Thus, in

245 this Section, we first introduce the Mutual Information and its quadratic variant,  
and then we present the proposed MI regularizer.

We assume a random variable  $Y$  representing the image representations of the feature space generated by a specific deep neural layer. We also assume a discrete-value variable  $C$  that represents the class labels. For each feature  
250 representation  $\mathbf{y}$  there is a class label  $c$ . The MI measures dependence between random variables, first introduced by Shannon, [47]. That is, the MI measures how much the uncertainty for the class label  $c$  is reduced by observing the feature vector  $\mathbf{y}$ . Let  $p(c)$  be the probability of observing the class label  $c$ , and  $p(\mathbf{y}, c)$  the probability density function of the corresponding joint distribution.

255 The MI between the two random variables is defined as:

$$MI(Y, C) = \sum_c \int_{\mathbf{y}} p(\mathbf{y}, c) \log \frac{p(\mathbf{y}, c)}{p(\mathbf{y})P(c)} d\mathbf{y}, \quad (4)$$

where  $P(c) = \int_{\mathbf{y}} p(\mathbf{y}, c) d\mathbf{y}$ . MI can also be interpreted as a Kullback-Leibler divergence between the joint probability density  $p(\mathbf{y}, c)$  and the product of marginal probabilities  $p(\mathbf{y})$  and  $P(c)$ .

260 QMI is derived by replacing the Kullback-Leibler divergence by the quadratic divergence measure [23]. That is:

$$QMI(Y, C) = \sum_c \int_{\mathbf{y}} (p(\mathbf{y}, c) - p(\mathbf{y})P(c))^2 d\mathbf{y}. \quad (5)$$

And thus, by expanding eq. (5) we arrive at the following equation:

$$QMI(Y, C) = \sum_c \int_{\mathbf{y}} p(\mathbf{y}, c)^2 d\mathbf{y} + \sum_c \int_{\mathbf{y}} p(\mathbf{y})^2 P(c)^2 d\mathbf{y} - 2 \sum_c \int_{\mathbf{y}} p(\mathbf{y}, c) p(\mathbf{y}) P(c) d\mathbf{y}. \quad (6)$$

The quantities appearing in eq. (6), are called *information potentials* and they are defined as follows:  $V_{IN} = \sum_c \int_{\mathbf{y}} p(\mathbf{y}, c)^2 d\mathbf{y}$ ,  $V_{ALL} = \sum_c \int_{\mathbf{y}} p(\mathbf{y})^2 P(c)^2 d\mathbf{y}$ ,  $V_{BTW} =$   
265  $\sum_c \int_{\mathbf{y}} p(\mathbf{y}, c) p(\mathbf{y}) P(c) d\mathbf{y}$ , and thus, the quadratic mutual information between the data samples and the corresponding class labels can be expressed as follows in terms of the information potentials:

$$QMI = V_{IN} + V_{ALL} - 2V_{BTW}. \quad (7)$$

If we assume that there are  $N_c$  different classes, each of them consisting of  $J_p$  samples, the class prior probability for the  $c_p$  class is given as:  $P(c_p) = \frac{J_p}{N}$ , where  $N$  corresponds to the total number of samples. Kernel Density Estimation [48] can be used to estimate the joint density probability:  $p(\mathbf{y}, c_p) = \frac{1}{N} \sum_{j=1}^{J_p} K(\mathbf{y}, \mathbf{y}_{pj}; \sigma^2)$ , for a symmetric kernel  $K$ , with width  $\sigma$ , where we use the notation  $\mathbf{y}_{pj}$  to refer to the  $j$ -th sample of the  $p$ -th class, as well as the probability density of  $Y$  as  $p(\mathbf{y}) = \sum_{p=1}^{J_p} p(\mathbf{y}, c_p) = \frac{1}{N} \sum_{j=1}^N K(\mathbf{y}, \mathbf{y}_j; \sigma^2)$ .

Thus, eq. (7) is formulated as follows:

$$V_{IN} = \frac{1}{N^2} \sum_{p=1}^{N_c} \sum_{k=1}^{J_p} \sum_{l=1}^{J_p} K(\mathbf{y}_{pk}, \mathbf{y}_{pl}; 2\sigma^2), \quad (8)$$

$$V_{ALL} = \frac{1}{N^2} \left( \sum_{p=1}^{N_c} \left( \frac{J_p}{N} \right)^2 \right) \sum_{k=1}^N \sum_{l=1}^N K(\mathbf{y}_k, \mathbf{y}_l; 2\sigma^2), \quad (9)$$

$$V_{BTW} = \frac{1}{N^2} \sum_{p=1}^{N_c} \frac{J_p}{N} \sum_{j=1}^{J_p} \sum_{k=1}^N K(\mathbf{y}_{pj}, \mathbf{y}_k; 2\sigma^2). \quad (10)$$

The kernel function  $K(\mathbf{y}_i, \mathbf{y}_j; \sigma^2)$  expresses the similarity between two samples  $i$  and  $j$ . There are several choices for the kernel function, [48]. For example, in [23] the Gaussian kernel is used, while in [25] the authors utilize a cosine similarity based kernel to avoid defining the width, in order to ensure that a meaningful probability estimation is obtained, since finetuning the width of the kernel is not a straightforward task, [49]. In our experiments, we use as kernel metric a Euclidean based similarity, which also absolves us from defining the width of the kernel. Given two vectors  $\mathbf{y}_i, \mathbf{y}_j$ , the Gaussian kernel is defined as:

$$K_G = \exp\left(-\frac{\|\mathbf{y}_i - \mathbf{y}_j\|_2^2}{\sigma}\right). \quad (11)$$

And the Euclidean-based similarity is defined as:

$$K_{ED} = \frac{1}{1 + \|\mathbf{y}_i - \mathbf{y}_j\|_2^2}. \quad (12)$$

The pairwise interactions described above between the samples can be interpreted as follows:

- $V_{IN}$  expresses the interactions between pairs of samples inside each class
- $V_{ALL}$  expresses the interactions between all pairs of samples, regardless of the class membership
- 290 •  $V_{BTW}$  expresses the interactions between samples of each class against all other samples

Thus, motivated by the QMI, in this work we propose a novel regularizer in order to enhance the generalization ability of a deep model. That is, apart from the optimization criterion defined by the hinge loss function which aims at separating the samples belonging to different classes, we propose an additional optimization criterion utilizing the information potential defined in eq. (7). We assume that the hinge loss preserves the  $V_{BTW}$  information potential which aims to separate samples belonging to different classes. Then, our objective is to maximize pairwise interactions between the samples described by the  $V_{IN} + V_{ALL}$  quantities. 295 The derived joint optimization criterion defines an additional loss function, which is attached to the penultimate convolutional layer (that is the last convolutional layer, before the one utilized for the classification task) and acts as regularizer to the main classification objective.

$$L_{MI} = -(V_{IN} + V_{ALL}), \quad (13)$$

where:

$$V_{IN} = \frac{1}{N^2} \sum_{p=1}^{N_c} \sum_{k=1}^{J_p} \sum_{l=1}^{J_p} K_{ED}(\mathbf{y}_{pk}, \mathbf{y}_{pl}), \quad (14)$$

305 and

$$V_{ALL} = \frac{1}{N^2} \left( \sum_{p=1}^{N_c} \left( \frac{J_p}{N} \right)^2 \right) \sum_{k=1}^N \sum_{l=1}^N K_{ED}(\mathbf{y}_k, \mathbf{y}_l). \quad (15)$$

Considering binary classification problems the above optimization criteria can be formulated as follows:

$$V_{IN} = \frac{1}{N^2} \sum_{k=1}^{J_1} \sum_{l=1}^{J_1} K_{ED}(\mathbf{y}_{1k}, \mathbf{y}_{1l}) + \frac{1}{N^2} \sum_{k=1}^{J_2} \sum_{l=1}^{J_2} K_{ED}(\mathbf{y}_{2k}, \mathbf{y}_{2l}), \quad (16)$$

and

$$V_{ALL} = \frac{1}{N^2} \left( \frac{J_1^2 + J_2^2}{N^2} \right) \sum_{k=1}^N \sum_{l=1}^N K_{ED}(\mathbf{y}_k, \mathbf{y}_l), \quad (17)$$

310 The total loss for the network training is defined as:

$$L_{total} = L_{Hinge} + \eta L_{MI}, \quad (18)$$

where the parameter  $\eta \in [0, 1]$  controls the relative importance of  $L_{MI}$ . We solve the above optimization problem using gradient descent. We should note that the proposed regularizer can be applied for the whole dataset, as well as in terms of mini-batch training. In our experiments we utilize the mini-batch mode. We  
 315 should finally note that in the regularized training we utilize the hinge loss layer since, as we show, it performs steadily better than the cross entropy one in binary classification problems, however the cross entropy loss could also be utilized.

## 5. Experiments

In this Section, we present the experiments conducted in order to evaluate  
 320 the impact of hinge loss against cross entropy loss in the classification performance, as well as the effectiveness of the proposed regularizer in improving the classification performance. To this aim, we first performed experiments on six datasets, four two-class datasets constructed for various classification problems involved in the context of media coverage of specific sport events by drones, and  
 325 two multiple-class datasets. Subsequently, in order to evaluate the performance of the MI regularizer, we performed experiments on the aforementioned two-class datasets. The descriptions of the utilized datasets are presented in the following subsections. Two post-hoc Bonferroni tests conducted in order to evaluate the statistical significance of the obtained results. Finally, qualitative results are  
 330 provided utilizing real world drone images for evaluating the performance of the proposed models trained with the MI regularizer. Throughout this work, we use Test Accuracy (Classification Accuracy) to evaluate the proposed method. Each experiment is repeated five times and we report the mean value and the standard deviation, considering the maximum value of Test Accuracy for each experiment.

335 The probabilistic factor is the random weight initialization. We also provide the  
curves of mean Test Accuracy.

### 5.1. Face

The dataset contains 70,000 train images of faces and equal number of train  
images of non-faces, and a test set of 7,468 images. Images of faces have been  
340 randomly selected from the AFLW [50], MTFI [51], and WIDER FACE [52]  
datasets. Input images are of size  $32 \times 32$ . Sample images of the constructed *Face*  
dataset are presented in Fig. 4.



Figure 4: Sample images of the Face dataset.

### 5.2. Football Player

The Football Player dataset, [21] consists of 98,000 train images that contain  
345 football players and non-football players, and a test set of 10,000 images. Input  
images are of size  $32 \times 32$ . Sample images of the *Football Player* dataset are  
illustrated in Fig. 5.



Figure 5: Sample images of the Football Player dataset.

### 5.3. Crowd-Drone

The dataset contains 40,000 train images of crowded scenes and non-crowded  
350 scenes, and 11,550 test images. Input images are of size  $64 \times 64$ . Sample images

of the constructed *Crowd-Drone* dataset are presented in Fig. 6.



Figure 6: Sample images of the Crowd-Drone dataset.

#### 5.4. Bicycles

The Bicycles dataset, [21] contains 51,200 equally distributed train images of bicycles (bicycle with bicyclist) and non-bicycles, and a test set of 10,000 images. Input images are of size  $64 \times 64$ . Sample images of the constructed *Bicycle* dataset are presented in Fig. 7.



Figure 7: Sample images of the Bicycles dataset.

#### 5.5. Street View House Numbers

The Street View House Numbers (SVHN) dataset, [53], obtained from house numbers in Google Street View images. It contains 73,257 train images and 26,032 test images, divided into 10 classes, 1 for each digit from 0 to 9. Input images are of size  $32 \times 32$  and sample images are provided in Fig. 8



Figure 8: Sample images of the SVHN dataset.

### 5.6. *Cifar-10*

The *Cifar-10* dataset, [54], consists of 60,000 images of size  $32 \times 32$  divided into 10 classes with 6,000 images per class. 50,000 images are used as the train set and 10,000 images as the test set. Sample images of the *Cifar-10* dataset are provided in Fig. 9



Figure 9: Sample images of the *Cifar-10* dataset.

### 5.7. *Implementation Details*

All the experiments conducted using the Caffe Deep Learning framework [55]. We use the mini-batch gradient descent for the networks' training. That is, an update is performed for every mini-batch of  $N_b$  training samples. The learning rate is set to  $10^{-3}$  and drops to  $10^{-4}$  gradually, and the batch size is set to 256. The momentum is 0.9. All the models are trained on an NVIDIA GeForce GTX 1080 with 8GB of GPU memory for 100 epochs, and can run in real-time when deployed on an NVIDIA Jetson TX2. Regarding the parameter which controls the importance of the regularization term of common regularizers, like L1 and L2, is usually set to 0.0005. In this work, the parameter  $\eta$  in (18) which controls the relative importance of the proposed regularizer's loss, is set to 0.001, since we have seen that in most cases provides best performance. However we should

note the proposed regularizer improves the performance for different values of  $\eta$ , too. For example, in Table 4, we representatively provide the experimental results for various values of  $\eta$ , in the case of the Bicycles dataset, utilizing the VGG-720p model. As we can see, the MI regularizer operates improvingly in any case. The comparisons among hinge loss against cross entropy loss, as well as only hinge loss against hinge loss & MI regularizer performed with the exact same training settings. Note that we have set the learning rate to  $10^{-3}$  (with drop policy) since it appears to be generally the most appropriate. However, experiments also performed with various values of learning rate (LR). In Table 5, we provide some indicative results in the Face dataset in terms of Test Accuracy, where the superiority of hinge loss over cross entropy loss is verified for various learning rates, while in Fig. 10 we present the corresponding mean Test Accuracy over the 100 epochs of training. Best results are printed in bold.

Training Approach	$\eta$	Test Accuracy
Only Hinge Loss	-	0.9785 $\pm$ 0.0021
Hinge Loss & MI Regularizer	1	0.9814 $\pm$ 0.0015
Hinge Loss & MI Regularizer	0.1	0.9827 $\pm$ 0.0018
Hinge Loss & MI Regularizer	0.01	0.9836 $\pm$ 0.002
Hinge Loss & MI Regularizer	0.001	<b>0.9884 <math>\pm</math> 0.0011</b>

Table 4: Bicycle Dataset - VGG-720p: Impact of parameter  $\eta$  in eq. (18)

LR	Cross Entropy Loss	Hinge Loss
$LR = 10^{-3}$ - drop	0.9126 $\pm$ 0.0021	<b>0.9273 <math>\pm</math> 0.0036</b>
$LR = 10^{-4}$ - fixed	0.896 $\pm$ 0.0025	<b>0.9197 <math>\pm</math> 0.0046</b>
$LR = 10^{-4}$ - drop	0.8755 $\pm$ 0.004	<b>0.9054 <math>\pm</math> 0.0014</b>

Table 5: Face Dataset - VGG-720p - Test Accuracy for various learning rates

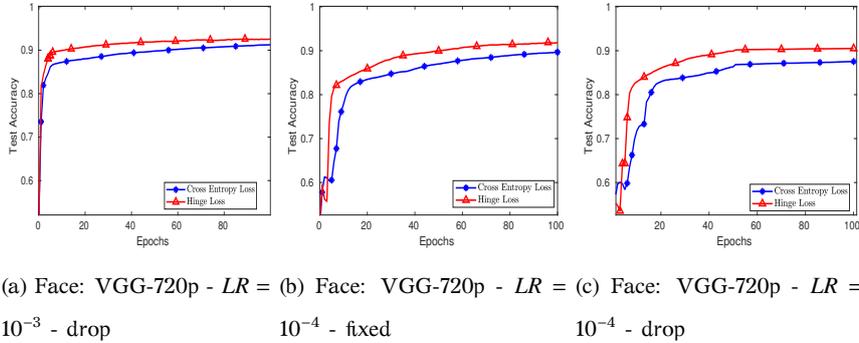


Figure 10: Face Dataset - VGG-720p - Various learning rates

### 5.8. Experimental Results

In the first set of experiments, we evaluate the classification performance of hinge loss against cross entropy loss. In Fig. 11 we provide the comparison of the mean Test Accuracy of the hinge loss against cross entropy loss, utilizing both the proposed models for all the two-class datasets, that is the Face, Football Player, Crowd-Drone, and Bicycles datasets. Furthermore, in Tables 8-11 we present the mean value and the standard deviation of the Test Accuracy, for the two losses under consideration. As we can observe the hinge loss is steadily superior over the cross entropy loss.

In Tables 6 and 7 we provide the corresponding evaluation results on the SVHN and Cifar-10 datasets, for the proposed VGG-720p model. We should note that the VGG-1080p architecture could not achieve remarkable classification performance on the aforementioned multi-class datasets. As we can observe, the cross entropy loss achieves marginally superior performance in the SVHN dataset, whilst the hinge loss outperforms the cross entropy one in the Cifar-10 dataset. Thus, we could observe that the hinge loss is undoubtedly the optimal choice for binary classification problems, considering lightweight deep models, however this is not a safe claim in multi-class classification problems.

In the second set of experiments, we evaluate the proposed MI regularizer. In Tables 8-11 we present the mean value and the standard deviation of the Test Accuracy, for the considered training approaches, that is utilizing only cross entropy

loss, only hinge loss, and hinge loss with the proposed MI regularizer, utilizing both the proposed models. Correspondingly, in Figs. 12-15 we illustrate the curves of mean Test Accuracy of the only hinge loss training against hinge loss & MI regularized training. We can see in the demonstrated results, that the proposed MI regularizer remarkably enhances the classification performance for both the proposed model architectures on all the utilized datasets. In Table 12 we provide representative comparisons against the common L1 and L2 regularizers on the Bicycles dataset, utilizing the VGG-720p model. As we can see, the L2 regularizer marginally improves the performance and the L1 one harms the performance, while the proposed MI regularizer achieves considerably better performance. Finally, it is noteworthy that we have tested the performance of the proposed MI regularizer on additional non-real-time models, where its effectiveness is further validated. However, we do not include these experiments, since a principal target of this work is to provide real-time models.

Training Approach	VGG-720p	Training Approach	VGG-720p
Cross Entropy Loss	<b>0.8987 ± 0.0019</b>	Cross Entropy Loss	0.5801 ± 0.0161
Hinge Loss	0.8972 ± 0.0057	Hinge Loss	<b>0.5919 ± 0.016</b>

Table 6: SVHN-10 Dataset - Test Accuracy

Table 7: CIFAR-10 Dataset - Test Accuracy

In the third set of experiments, we conducted two post-hoc Bonferroni tests [56], first for ranking the hinge loss and cross entropy loss for binary classification problems and evaluating the statistical significance of the obtained results, and second for ranking the proposed regularization method and the only hinge loss training and evaluating the statistical significance of the obtained results. The

Training Approach	VGG-720p	VGG-1080p
Only Cross Entropy Loss	0.9126 ± 0.0021	0.8738 ± 0.0052
Only Hinge Loss	0.9273 ± 0.0036	0.8841 ± 0.004
Hinge Loss & MI Regularizer	<b>0.9292 ± 0.0048</b>	<b>0.8896 ± 0.0007</b>

Table 8: Face Dataset - Test Accuracy

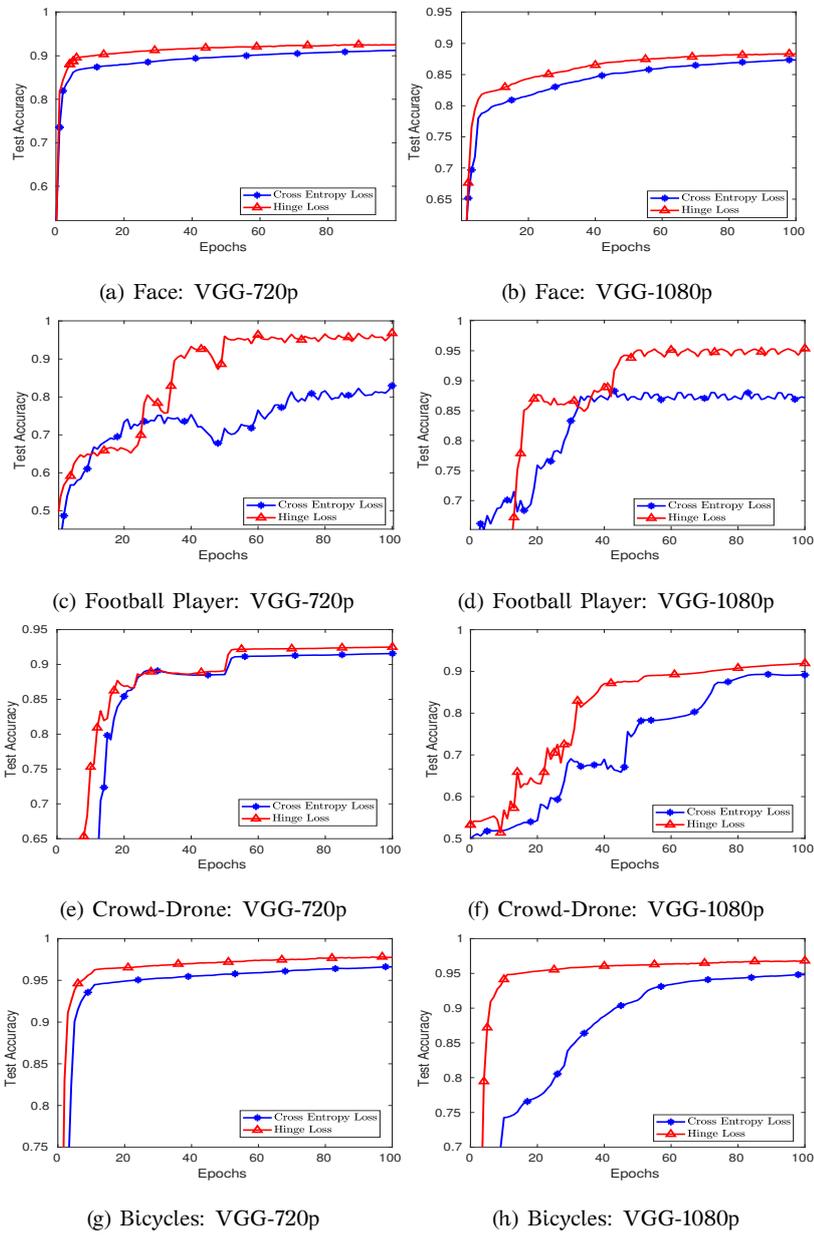


Figure 11: Cross Entropy Loss VS Hinge Loss

<b>Training Approach</b>	<b>VGG-720p</b>	<b>VGG-1080p</b>
Only Cross Entropy Loss	0.9183 $\pm$ 0.0307	0.8897 $\pm$ 0.0747
Only Hinge Loss	0.9680 $\pm$ 0.0080	0.9568 $\pm$ 0.01
Hinge Loss & MI Regularizer	<b>0.9813 <math>\pm</math> 0.0027</b>	<b>0.9744 <math>\pm</math> 0.01</b>

Table 9: Football Player Dataset - Test Accuracy

<b>Training Approach</b>	<b>VGG-720p</b>	<b>VGG-1080p</b>
Only Cross Entropy Loss	0.9157 $\pm$ 0.0058	0.9030 $\pm$ 0.014
Only Hinge Loss	0.9334 $\pm$ 0.01	0.9194 $\pm$ 0.0082
Hinge Loss & MI Regularizer	<b>0.9371 <math>\pm</math> 0.0011</b>	<b>0.9303 <math>\pm</math> 0.0076</b>

Table 10: Crowd-Drone Dataset - Test Accuracy

<b>Training Approach</b>	<b>VGG-720p</b>	<b>VGG-1080p</b>
Only Cross Entropy Loss	0.9664 $\pm$ 0.001	0.9484 $\pm$ 0.0023
Only Hinge Loss	0.9785 $\pm$ 0.0021	0.9684 $\pm$ 0.0037
Hinge Loss & MI Regularizer	<b>0.9884 <math>\pm</math> 0.0011</b>	<b>0.9696 <math>\pm</math> 0.0018</b>

Table 11: Bicycle Dataset - Test Accuracy

<b>Training Approach</b>	<b>Test Accuracy</b>
Only Hinge Loss	0.9785 $\pm$ 0.0021
Hinge Loss & MI Regularizer	<b>0.9884 <math>\pm</math> 0.0011</b>
Hinge Loss & L1 Regularizer	0.9719 $\pm$ 0.0027
Hinge Loss & L2 Regularizer	0.9797 $\pm$ 0.001

Table 12: Bicycles Dataset - VGG-720p: Comparison against the common L1 and L2 regularizers

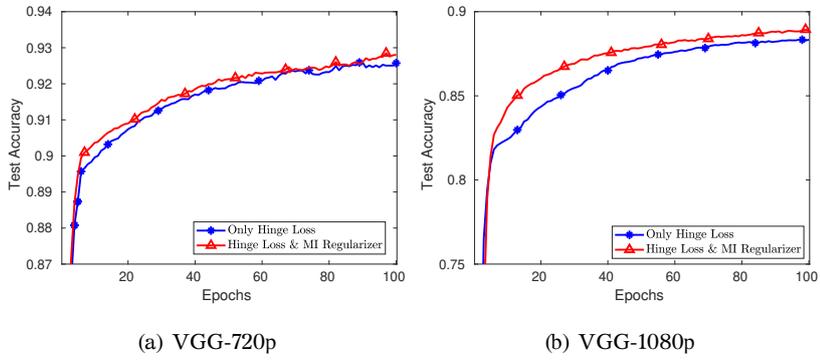


Figure 12: Face Dataset: MI Regularizer

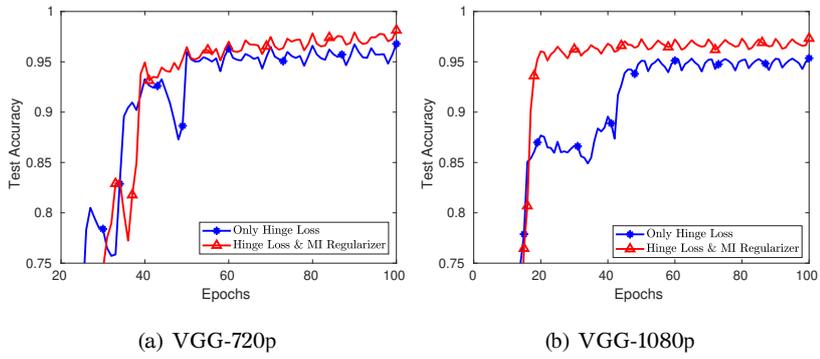


Figure 13: Football Player Dataset: MI Regularizer

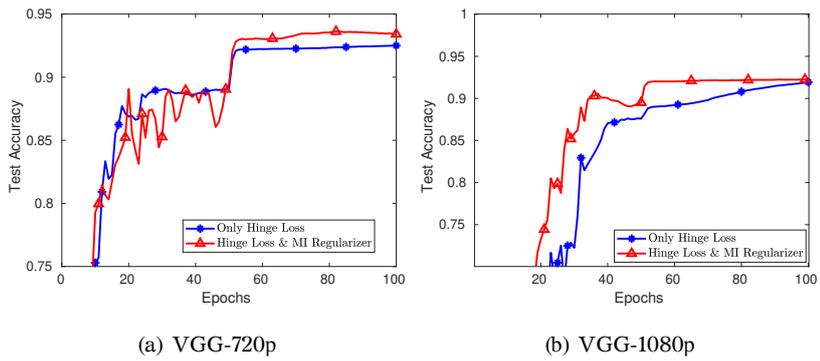


Figure 14: Crowd-Drone Dataset: MI Regularizer

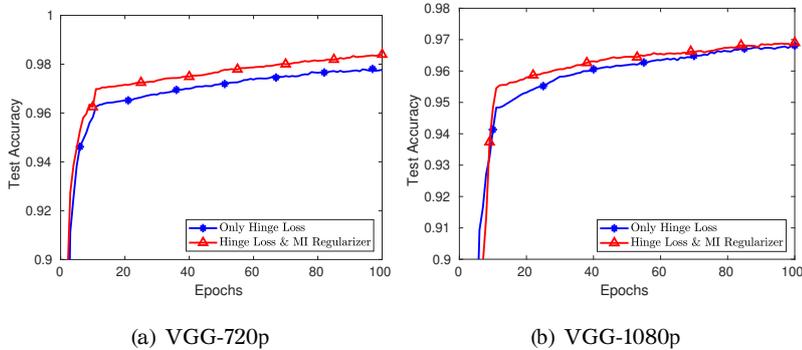


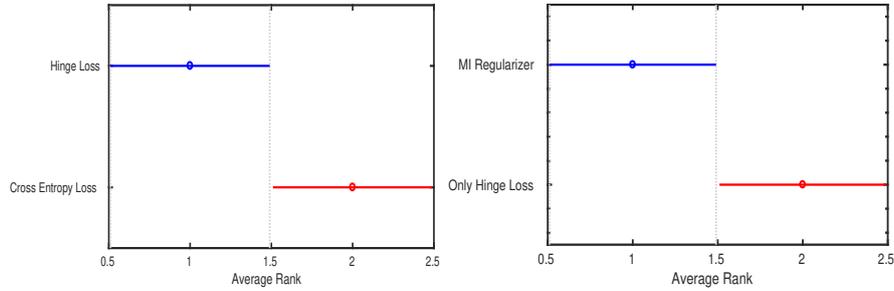
Figure 15: Bicycles Dataset: MI Regularizer

performance of two methods is significantly different, if the corresponding average ranks over the datasets differ by at least the critical difference:

$$CD = q_\alpha \sqrt{\frac{m(m+1)}{6D}}, \quad (19)$$

where  $m$  is the number of methods compared,  $D$  is the number of datasets and  
 435 critical values  $q_\alpha$  can be found in [56]. In our comparisons we set  $\alpha = 0.05$ . The  
 number of datasets is four in the performed tests. The compared methods are  
 two, that is the training using the hinge loss is compared with a control method  
 which is the training with cross entropy loss, and second the proposed regularizer  
 is compared with a control method which is the only hinge loss training approach.  
 440 The ranking results are illustrated in Figs. 16a and 16b, respectively. The vertical  
 axis depicts the two methods, while the horizontal axis depicts the performance  
 ranking. The circles indicate the mean rank and the intervals around them indi-  
 cate the confidence interval as this is determined by the  $CD$  value. Overlapping  
 445 intervals between two methods indicate that there is not a statistically significant  
 difference between the corresponding ranks, while non-overlapping intervals indi-  
 cate that the compared methods are significantly different. As we can observe, the  
 hinge loss is significantly different against cross entropy loss for binary classifica-  
 tion problems, as well as the proposed regularizer is significantly different against  
 the only hinge loss training approach. We should note that we representatively  
 450 present the performance utilizing the VGG-720p architecture, however identical

performance is achieved utilizing the VGG-1080p architecture.



(a) Cross Entropy Loss vs Hinge Loss for Binary Classification      (b) MI regularizer vs Only Hinge Loss

Figure 16: Post-Hoc Bonferroni Tests

Finally, in the forth set of experiments, we compute heatmaps of the object's presence for the classification problems under consideration, so as to provide some qualitative results for evaluating the effectiveness of the proposed real-time models trained with the MI regularizer. Thus, considering high resolution images captured by drones, the heatmaps considering the tasks of bicycle, football player, and crowd detection utilizing the proposed VGG-1080p models are computed. Evaluation results provided in Fig. 17 indicate the efficiency of the proposed models.

## 6. Conclusions

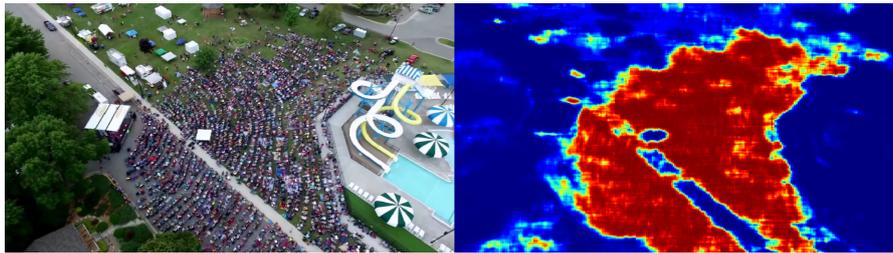
In this paper, we proposed regularized lightweight CNN models for addressing various classification tasks (e.g. crowd detection, bicycle detection, etc.) able to run in real-time on-drone. Second, in order to explore ways to enhance the classification performance of the lightweight models, we extensively investigated the impact of two widely used classification losses, that is the cross entropy and hinge losses, on the classification performance, considering binary classification problems. Third, we proposed a novel regularizer motivated by the QMI, the so-called MI regularizer. The performance was evaluated on four datasets.



(a) Bicycle Detection



(b) Football Player Detection



(c) Crowd Detection

Figure 17: Heatmaps on real world drone images for specific detection problems utilizing the corresponding proposed models.

The evaluation results indicate that hinge loss is undoubtedly the optimal choice  
 470 for binary classification problems, considering lightweight deep models. Further-  
 more, the effectiveness of the proposed regularizer in enhancing the generalization  
 ability of the proposed models is also validated. Finally, even the proposed mod-  
 els can achieve significant performance in the considered two-class classification  
 problems, it can be observed in the experimental results that there is a drop per-  
 475 formance in more complex problems (i.e Cifar-10 dataset). Thus, future work will

consist on developing more efficient models, capable of efficiently addressing, in terms of both speed and accuracy, more complicated problems under the considered computation and memory constraints. A first step towards this goal is to investigate if we can incorporate established methodologies (e.g. skip connections  
480 [41]) to our work.

### Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR). This publication reflects the authors' views only. The European Commission is  
485 not responsible for any use that may be made of the information it contains.

- [1] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, M. S. Lew, Deep learning for visual understanding: A review, *Neurocomputing* 187 (2016) 27–48.
- [2] J. Schmidhuber, Deep learning in neural networks: An overview, *Neural networks* 61 (2015) 85–117.
- 490 [3] B. B. Le Cun, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, Handwritten digit recognition with a back-propagation network, in: *Advances in neural information processing systems* 2, Morgan Kaufmann Publishers Inc., 1990, pp. 396–404.
- [4] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang,  
495 G. Wang, J. Cai, et al., Recent advances in convolutional neural networks, *Pattern Recognition* 77 (2018) 354–377.
- [5] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: *Advances in neural information processing systems*, 2012, pp. 1097–1105.
- 500 [6] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556.

- [7] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, in: *Advances in neural information processing systems*, 2015, pp. 91–99.
- 505 [8] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg, Ssd: Single shot multibox detector, in: *European conference on computer vision*, Springer, 2016, pp. 21–37.
- [9] J. Redmon, A. Farhadi, Yolo9000: better, faster, stronger, arXiv preprint arXiv:1612.08242.
- 510 [10] S. Pang, J. Zhu, J. Wang, V. Ordonez, J. Xue, Building discriminative cnn image representations for object retrieval using the replicator equation, *Pattern Recognition*.
- [11] M. Tzelepi, A. Tefas, Deep convolutional learning for content based image retrieval, *Neurocomputing* 275 (2018) 2467–2478.
- 515 [12] M. Tzelepi, A. Tefas, Deep convolutional image retrieval: A general framework, *Signal Processing: Image Communication* 63 (2018) 30–43.
- [13] P. Li, D. Wang, L. Wang, H. Lu, Deep visual tracking: Review and experimental comparison, *Pattern Recognition* 76 (2018) 323–338.
- [14] B. Chen, P. Li, C. Sun, D. Wang, G. Yang, H. Lu, Multi attention module  
520 for visual tracking, *Pattern Recognition*.
- [15] E. Daskalakis, M. Tzelepi, A. Tefas, Learning deep spatiotemporal features for video captioning, *Pattern Recognition Letters* 116 (2018) 143–149.
- [16] A. Toshev, C. Szegedy, Deeppose: Human pose estimation via deep neural networks, in: *Proceedings of the IEEE Conference on Computer Vision and  
525 Pattern Recognition*, 2014, pp. 1653–1660.
- [17] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, arXiv preprint arXiv:1704.04861.

- [18] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, K. Keutzer,  
530 Squeezenet: Alexnet-level accuracy with 50x fewer parameters and  $< 0.5$  mb  
model size, arXiv preprint arXiv:1602.07360.
- [19] L. Apvrille, T. Tanzi, J.-L. Dugelay, Autonomous drones for assisting rescue  
services within the context of natural disasters, in: General Assembly and  
Scientific Symposium (URSI GASS), 2014 XXXIth URSI, IEEE, 2014, pp. 1–4.
- 535 [20] A. Claesson, D. Fredman, L. Svensson, M. Ringh, J. Hollenberg, P. Nordberg,  
M. Rosenqvist, T. Djarv, S. Osterberg, J. Lennartsson, et al., Unmanned aerial  
vehicles (drones) in out-of-hospital-cardiac-arrest, *Scandinavian journal of  
trauma, resuscitation and emergency medicine* 24 (1) (2016) 124.
- [21] M. Tzelepi, A. Tefas, Graph embedded convolutional neural networks in hu-  
540 man crowd detection for drone flight safety, *IEEE Transactions on Emerging  
Topics in Computational Intelligence*.
- [22] N. Passalis, A. Tefas, I. Pitas, Efficient camera control using 2d visual infor-  
mation for unmanned aerial vehicle-based cinematography, in: *Circuits and  
Systems (ISCAS)*, 2018 IEEE International Symposium on, IEEE, 2018, pp.  
545 1–5.
- [23] K. Torkkola, Feature extraction by non-parametric mutual information max-  
imization, *Journal of machine learning research* 3 (Mar) (2003) 1415–1438.
- [24] D. Bouzas, N. Arvanitopoulos, A. Tefas, Graph embedded nonparametric mu-  
tual information for supervised dimensionality reduction, *IEEE transactions  
550 on neural networks and learning systems* 26 (5) (2015) 951–963.
- [25] N. Passalis, A. Tefas, Learning deep representations with probabilistic knowl-  
edge transfer, in: *The European Conference on Computer Vision (ECCV)*,  
2018.
- [26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov,  
555 Dropout: a simple way to prevent neural networks from overfitting, *The  
Journal of Machine Learning Research* 15 (1) (2014) 1929–1958.

- [27] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, R. Fergus, Regularization of neural networks using dropconnect, in: International Conference on Machine Learning, 2013, pp. 1058–1066.
- 560 [28] A. S. Weigend, D. E. Rumelhart, B. A. Huberman, Generalization by weight-elimination with application to forecasting, in: Advances in neural information processing systems, 1991, pp. 875–882.
- [29] D. J. MacKay, Probable networks and plausible predictions—a review of practical bayesian methods for supervised neural networks, *Network: Computation in Neural Systems* 6 (3) (1995) 469–505.
- 565 [30] R. Caruana, Multitask learning, *Machine learning* 28 (1) (1997) 41–75.
- [31] J. Weston, F. Ratle, R. Collobert, Deep learning via semi-supervised embedding, in: Proceedings of the 25th international conference on Machine learning, ACM, 2008, pp. 1168–1175.
- 570 [32] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, Z. Tu, Deeply-supervised nets, in: Artificial Intelligence and Statistics, 2015, pp. 562–570.
- [33] S. Zhang, L. Wen, X. Bian, Z. Lei, S. Z. Li, Single-shot refinement neural network for object detection, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 4203–4212.
- 575 [34] P. Nousi, E. Patsiouras, A. Tefas, I. Pitas, Convolutional neural networks for visual information analysis with limited computing resources, in: 2018 25th IEEE International Conference on Image Processing (ICIP), IEEE, 2018, pp. 321–325.
- [35] V. N. Vapnik, V. Vapnik, *Statistical learning theory*, Vol. 1, Wiley New York, 1998.
- 580 [36] C. Tzelepis, V. Mezaris, I. Patras, Linear maximum margin classifier for learning from uncertain data, *IEEE transactions on pattern analysis and machine intelligence* 40 (12) (2018) 2948–2962.

- [37] V. Mygdalis, A. Tefas, I. Pitas, Exploiting multiplex data relationships in support vector machines, *Pattern Recognition* 85 (2019) 70–77.
- [38] J. Weston, C. Watkins, Multi-class support vector machines, Tech. rep., Cite-seer (1998).
- [39] C.-W. Hsu, C.-J. Lin, A comparison of methods for multiclass support vector machines, *IEEE transactions on Neural Networks* 13 (2) (2002) 415–425.
- [40] U. Dogan, T. Glasmachers, C. Igel, A unified view on multi-class support vector classification, *Journal of Machine Learning Research* 17 (45) (2016) 1–32.  
URL <http://jmlr.org/papers/v17/11-229.html>
- [41] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [42] J. S. Bridle, Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition, in: *Neurocomputing*, Springer, 1990, pp. 227–236.
- [43] Y. Tang, Deep learning using linear support vector machines, arXiv preprint arXiv:1306.0239.
- [44] Y.-J. Lee, O. L. Mangasarian, Ssvm: A smooth support vector machine for classification, *Computational optimization and Applications* 20 (1) (2001) 5–22.
- [45] K. Janocha, W. M. Czarnecki, On loss functions for deep neural networks in classification, arXiv preprint arXiv:1702.05659.
- [46] M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: *European conference on computer vision*, Springer, 2014, pp. 818–833.

- 610 [47] C. E. Shannon, A mathematical theory of communication, *ACM SIGMOBILE mobile computing and communications review* 5 (1) (2001) 3–55.
- [48] D. W. Scott, *Multivariate density estimation: theory, practice, and visualization*, John Wiley & Sons, 2015.
- [49] S.-T. Chiu, Bandwidth selection for kernel density estimation, *The Annals of Statistics* (1991) 1883–1905.
- 615 [50] M. Koestinger, P. Wohlhart, P. M. Roth, H. Bischof, Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization, in: *Computer Vision Workshops (ICCV Workshops)*, 2011 IEEE International Conference on, IEEE, 2011, pp. 2144–2151.
- 620 [51] Z. Zhang, P. Luo, C. C. Loy, X. Tang, Facial landmark detection by deep multi-task learning, in: *European Conference on Computer Vision*, Springer, 2014, pp. 94–108.
- [52] S. Yang, P. Luo, C.-C. Loy, X. Tang, Wider face: A face detection benchmark, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5525–5533.
- 625 [53] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Y. Ng, Reading digits in natural images with unsupervised feature learning, in: *NIPS workshop on deep learning and unsupervised feature learning*, Vol. 2011, 2011, p. 5.
- [54] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images.
- 630 [55] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: Convolutional architecture for fast feature embedding, in: *Proceedings of the 22nd ACM international conference on Multimedia*, ACM, 2014, pp. 675–678.
- 635 [56] J. Demšar, Statistical comparisons of classifiers over multiple data sets, *Journal of Machine learning research* 7 (Jan) (2006) 1–30.

## **7.11 Quadratic mutual information regularization in real-time deep CNN models**

The appended paper follows.

# QUADRATIC MUTUAL INFORMATION REGULARIZATION IN REAL-TIME DEEP CNN MODELS

*Maria Tzelepi and Anastasios Tefas*

Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece

## ABSTRACT

In this paper, regularized lightweight deep convolutional neural network models, capable of effectively operating in real-time on devices with restricted computational power for high-resolution video input are proposed. Furthermore, a novel regularization method motivated by the Quadratic Mutual Information, in order to improve the generalization ability of the utilized models is proposed. Extensive experiments on various binary classification problems involved in autonomous systems are performed, indicating the effectiveness of the proposed models as well as of the proposed regularizer.

**Index Terms**— Quadratic Mutual Information, Regularizer, Lightweight Models, Real-time, Deep Learning.

## 1. INTRODUCTION

Deep Learning (DL) models, [1], and especially deep Convolutional Neural Networks (CNN) have been established among the most efficient research directions in a wide range of computer vision tasks, eclipsing previous shallow algorithms, [2]. However, state-of-the-art DL models are usually computation-heavy, obstructing their application on autonomous systems. This has directed the research towards the development of lightweight models capable of running on devices with restricted computational resources such as mobile phones and embedded systems, [3, 4].

Thus, in this paper, we propose lightweight deep CNN models allowing real-time deployment for high resolution images for specific binary classification problems involved in autonomous robots applications. Specifically, we consider the media coverage of certain events by UAVs (also known as drones). We deal with face, bicycle, and football player detection, as well as crowd detection towards human crowd avoidance. Our goal is to provide semantic heatmaps, by predicting for each location within the captured high-resolution scene the considered object’s presence. That is, we train models with RGB input of size either  $32 \times 32$  or  $64 \times 64$ , and then test images are introduced to the network, and, through sliding window process, for every window  $32 \times 32$  or  $64 \times 64$  respectively, we compute the output of the network at the last convolutional layer. An example of a football player heatmap

is provided in Fig.1.



**Fig. 1:** An image containing football players and the corresponding predicted heatmap of football player presence.

Surveying the relevant literature we can see that several works have emerged towards designing lightweight models. For example, an approach that proposes to replace  $3 \times 3$  convolutions with  $1 \times 1$  convolutions to create a very small network capable of reducing  $50\times$  the number of parameters while obtaining high accuracy is proposed in [4]. However, there is no other work in the recent literature proposing real-time models capable of running on devices with limited computational resources for high resolution input. Additionally, in [5] where a computation-efficient CNN model is proposed for mobile devices with limited computing power, it is demonstrated that in order to accomplish real-time deployment, someone has to reduce the input frame resolution to  $224 \times 224$ .

Furthermore, since we deal with lightweight models that usually have inferior performance compared to the more complex ones, we focus on enhancing their performance. That is, the second goal of this work is to propose a novel regularization method in order to circumvent over-fitting and enhance the generalization ability of the proposed real-time models. Generally, this constitutes a major issue in deep learning algorithms, since neural networks are prone to over-fitting due to their high capacity. During the past years, several regularization schemes have been proposed in order to prevent overfitting in neural networks, e.g. common regularization methods, like  $\mathcal{L}1/\mathcal{L}2$  regularization, and Dropout [6]. Besides, multitask-learning [7] has been proposed as a way to improve the generalization ability of a model. For example, in [8] the authors introduced techniques developed in semi-

supervised learning in the deep learning domain, whilst in [9], a novel CNN architecture with an SVM classifier at every hidden layer is proposed. This companion objective acts as a kind of feature regularization.

In this work, the so-called *Mutual Information (MI) regularizer* is proposed. The proposed regularizer is inspired by the Quadratic Mutual Information (QMI) measure [10], which is a variant of the commonly used Mutual Information, an information-theoretic measure of dependence between random variables. That is, apart from the classification loss, we propose to attach an additional optimization criterion based on the QMI. Recently, QMI reformulated to produce a kernel dimensionality reduction method under the Graph Embedding framework [11], while in [12] a Probabilistic Knowledge Transfer method proposed exploiting the QMI. It is noteworthy that the proposed regularizer is generic and can be applied in several deep learning architectures for classification purposes.

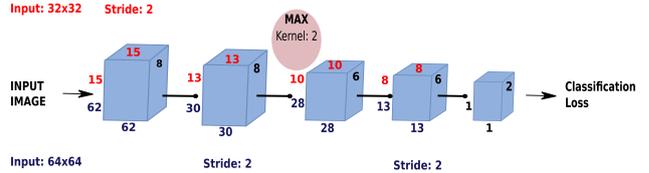
The remainder of the manuscript is structured as follows. The utilized CNN architectures are described in Section 2. The proposed MI regularizer is presented in Section 3. The experiments conducted to validate the proposed method are provided in Section 4. Finally, the conclusions are drawn in Section 5.

## 2. REAL-TIME CNN MODELS

In this paper, our goal is to propose effective deep models for various binary classification problems, which allow real-time deployment (about 25 frames per second) on-drone for high resolution images. It should be emphasized that it is of utmost importance for the application to handle high resolution images, since objects in drone-captured images are extremely small, and thus image resizing in order to reach real-time deployment limits, would further shrink the object of interest, rendering the detection infeasible. An example that highlights the demand for high resolution images is provided in Fig. 3. That is, an aerial image that contains bicycles (bicycles with bicyclists) is provided in Fig. 3a, and the resulting heatmaps for input of two different resolutions, utilizing the proposed model are provided in Figs. 3b-3c. As it can be observed, as the resolution increases, better performance can be achieved.

The objective of this work is two-fold: a) to propose real-time architectures that can be deployed on-drone, and b) to improve the state-of-the-art performance using MI regularization. Thus, we propose a model consisting of only five convolutional layers, by discarding the deepest layers and pruning filters of the widely used VGG-16 model [13]. That is, we use the first four convolutional layers of the VGG-16 model with pruned filters, while the last convolutional layer consists of two channels, each for a class, since we deal with binary classification problems. The proposed model runs in real-time on-drone on test images of 1080p (1920×1080) resolution, utilizing the sliding window process, as previously mentioned.

The model is abbreviated as as VGG-1080p based on this attribute. Since we deal with datasets of  $32 \times 32$  and  $64 \times 64$  input dimensions, we propose two variant models. The models use same kernels and channels, and real-time deployment is achieved with appropriate stride and pooling, as it is shown in Fig. 2. The evaluation results on the deployment speed for the proposed models are provided in the Experiments Section.



**Fig. 2:** VGG-1080p Architecture: Details for input of size  $32 \times 32$  are printed in red, while details for input of size  $64 \times 64$  are printed in blue.

## 3. THE PROPOSED MI REGULARIZER

In this paper, a novel regularizer motivated by the Quadratic Mutual Information [10] is proposed. Apart from the classification loss, we propose a regularization loss derived from the so-called information potentials of the QMI. Thus, in this Section, we first introduce the Mutual Information and its quadratic variant, and then we present the proposed MI regularizer.

We assume a random variable  $Y$  representing the image representations of the feature space generated by a specific deep neural layer. We also assume a discrete-value variable  $C$  that represents the class labels. For each feature representation  $\mathbf{y}$  there is a class label  $c$ . The MI measures dependence between random variables, first introduced by Shannon, [14]. That is, the MI measures how much the uncertainty for the class label  $c$  is reduced by observing the feature vector  $\mathbf{y}$ . Let  $p(c)$  be the probability of observing the class label  $c$ , and  $p(\mathbf{y}, c)$  the probability density function of the corresponding joint distribution.

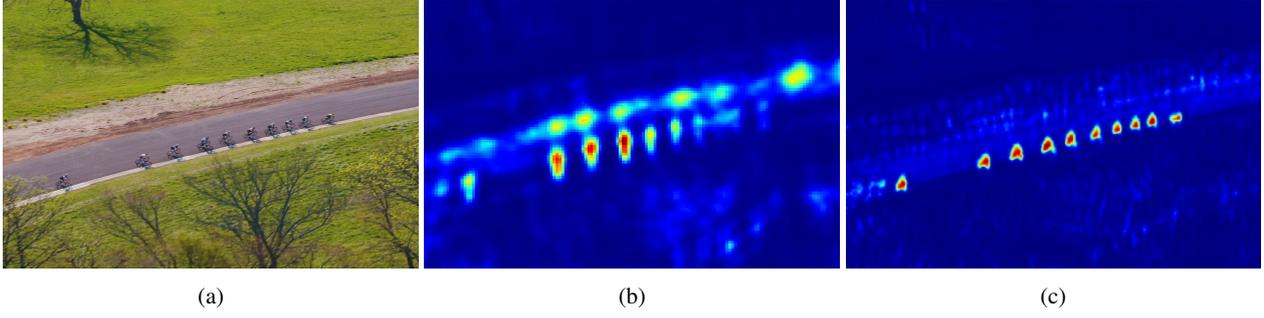
The MI between the two random variables is defined as:

$$MI(Y, C) = \sum_c \int_{\mathbf{y}} p(\mathbf{y}, c) \log \frac{p(\mathbf{y}, c)}{p(\mathbf{y})P(c)} d\mathbf{y}, \quad (1)$$

where  $P(c) = \int_{\mathbf{y}} p(\mathbf{y}, c) d\mathbf{y}$ . MI can also be interpreted as a Kullback-Leibler divergence between the joint probability density  $p(\mathbf{y}, c)$  and the product of marginal probabilities  $p(\mathbf{y})$  and  $P(c)$ .

QMI is derived by replacing the Kullback-Leibler divergence by the quadratic divergence measure [10]. That is:

$$QMI(Y, C) = \sum_c \int_{\mathbf{y}} (p(\mathbf{y}, c) - p(\mathbf{y})P(c))^2 d\mathbf{y}. \quad (2)$$



**Fig. 3:** An aerial high resolution image containing bicycles (3a), and the resulting heatmaps for input of size  $640 \times 480$  (3b) and for input of size  $1920 \times 1080$  (3c) utilizing the proposed VGG-1080p model trained for bicycle detection.

And thus, by expanding eq. (2) we arrive at the following equation:

$$QMI(Y, C) = \sum_c \int_{\mathbf{y}} p(\mathbf{y}, c)^2 d\mathbf{y} + \sum_c \int_{\mathbf{y}} p(\mathbf{y})^2 P(c)^2 d\mathbf{y} - 2 \sum_c \int_{\mathbf{y}} p(\mathbf{y}, c) p(\mathbf{y}) P(c) d\mathbf{y}. \quad (3)$$

The quantities appearing in eq. (3), are called *information potentials* and they are defined as follows:  $V_{IN} = \sum_c \int_{\mathbf{y}} p(\mathbf{y}, c)^2 d\mathbf{y}$ ,  $V_{ALL} = \sum_c \int_{\mathbf{y}} p(\mathbf{y})^2 P(c)^2 d\mathbf{y}$ ,  $V_{BTW} = \sum_c \int_{\mathbf{y}} p(\mathbf{y}, c) p(\mathbf{y}) P(c) d\mathbf{y}$ , and thus, the quadratic mutual information between the data samples and the corresponding class labels can be expressed as follows in terms of the information potentials:

$$QMI = V_{IN} + V_{ALL} - 2V_{BTW}. \quad (4)$$

If we assume that there are  $N_c$  different classes, each of them consisting of  $J_p$  samples, the class prior probability for the  $c_p$  class is given as:  $P(c_p) = \frac{J_p}{N}$ , where  $N$  corresponds to the total number of samples. Kernel Density Estimation [15] can be used to estimate the joint density probability:  $p(\mathbf{y}, c_p) = \frac{1}{N} \sum_{j=1}^{J_p} K(\mathbf{y}, \mathbf{y}_{pj}; \sigma^2)$ , for a symmetric kernel  $K$ , with width  $\sigma$ , where we use the notation  $\mathbf{y}_{pj}$  to refer to the  $j$ -th sample of the  $p$ -th class, as well as the probability density of  $Y$  as  $p(\mathbf{y}) = \sum_{p=1}^{J_p} p(\mathbf{y}, c_p) = \frac{1}{N} \sum_{j=1}^N K(\mathbf{y}, \mathbf{y}_j; \sigma^2)$ .

Thus, eq. (4) is formulated as follows:

$$V_{IN} = \frac{1}{N^2} \sum_{p=1}^{N_c} \sum_{k=1}^{J_p} \sum_{l=1}^{J_p} K(\mathbf{y}_{pk}, \mathbf{y}_{pl}; 2\sigma^2), \quad (5)$$

$$V_{ALL} = \frac{1}{N^2} \left( \sum_{p=1}^{N_c} \left( \frac{J_p}{N} \right)^2 \right) \sum_{k=1}^N \sum_{l=1}^N K(\mathbf{y}_k, \mathbf{y}_l; 2\sigma^2), \quad (6)$$

$$V_{BTW} = \frac{1}{N^2} \sum_{p=1}^{N_c} \frac{J_p}{N} \sum_{j=1}^{J_p} \sum_{k=1}^N K(\mathbf{y}_{pj}, \mathbf{y}_k; 2\sigma^2). \quad (7)$$

The kernel function  $K(\mathbf{y}_i, \mathbf{y}_j; \sigma^2)$  expresses the similarity between two samples  $i$  and  $j$ . There are several choices for the kernel function, [15]. For example, in [10] the Gaussian kernel is used, while in [12] the authors utilize a cosine similarity based kernel to avoid defining the width, in order to ensure that a meaningful probability estimation is obtained, since finetuning the width of the kernel is not a straightforward task, [16]. In our experiments, we use as kernel metric a Euclidean based similarity, defined as  $K_{ED} = \frac{1}{1 + \|\mathbf{y}_i - \mathbf{y}_j\|_2^2}$ , which also absolves us from defining the width of the kernel.

The pairwise interactions described above between the samples can be interpreted as follows:  $V_{IN}$  expresses the interactions between pairs of samples inside each class.  $V_{ALL}$  expresses the interactions between all pairs of samples, regardless of the class membership. Finally,  $V_{BTW}$  expresses the interactions between samples of each class against all other samples.

Thus, motivated by the QMI, in this work we propose a novel regularizer in order to enhance the generalization ability of a deep model. That is, apart from the optimization criterion defined by the hinge loss function which aims at separating the samples belonging to different classes, we propose an additional optimization criterion utilizing the information potential defined in eq. (4). We assume that the hinge loss preserves the  $V_{BTW}$  information potential which aims to separate samples belonging to different classes. Then, our objective is to maximize pairwise interactions between the samples described by the  $V_{IN} + V_{ALL}$  quantities. The derived joint optimization criterion defines an additional loss function, which is attached to the penultimate convolutional layer (that is the last convolutional layer, before the one utilized for the classification task) and acts as regularizer to the main classification objective.

$$J_{MI} = -(V_{IN} + V_{ALL}), \quad (8)$$

where:

$$V_{IN} = \frac{1}{N^2} \sum_{p=1}^{N_c} \sum_{k=1}^{J_p} \sum_{l=1}^{J_p} K_{ED}(\mathbf{y}_{pk}, \mathbf{y}_{pl}), \quad (9)$$

and

$$V_{ALL} = \frac{1}{N^2} \left( \sum_{p=1}^{N_c} \left( \frac{J_p}{N} \right)^2 \right) \sum_{k=1}^N \sum_{l=1}^N K_{ED}(\mathbf{y}_k, \mathbf{y}_l). \quad (10)$$

Considering binary classification problems the above optimization criteria can be formulated as follows:

$$V_{IN} = \frac{1}{N^2} \sum_{k=1}^{J_1} \sum_{l=1}^{J_1} K_{ED}(\mathbf{y}_{1k}, \mathbf{y}_{1l}) + \frac{1}{N^2} \sum_{k=1}^{J_2} \sum_{l=1}^{J_2} K_{ED}(\mathbf{y}_{2k}, \mathbf{y}_{2l}), \quad (11)$$

and

$$V_{ALL} = \frac{1}{N^2} \left( \frac{J_1^2 + J_2^2}{N^2} \right) \sum_{k=1}^N \sum_{l=1}^N K_{ED}(\mathbf{y}_k, \mathbf{y}_l), \quad (12)$$

The total loss for the network training is defined as:

$$J_{total} = J_{class} + \eta J_{MI}, \quad (13)$$

where  $J_{class}$  stands for the classification loss, and the parameter  $\eta \in [0, 1]$  controls the relative importance of  $J_{MI}$ . We solve the above optimization problem using gradient descent. We should note that the proposed regularizer can be applied for the whole dataset, as well as in terms of mini-batch training. In our experiments we utilize the mini-batch mode. We should finally note that in the regularized training we utilize the hinge loss since, as we have experimentally observed, it performs better than the cross entropy one in binary classification problems, however the cross entropy loss could also be utilized.

## 4. EXPERIMENTS

In this Section, we present the experiments conducted in order to evaluate the proposed models regarding the deployment speed as well as the proposed regularization method. We evaluate the detection speed in terms of frames per second (fps), while we use test accuracy to evaluate the proposed regularizer, since we deal with balanced datasets. Each experiment is executed five times, and the mean value and the standard deviation are reported, considering the maximum value of the test accuracy for each experiment. The probabilistic factor is the random weight initialization.

### 4.1. Datasets

In order to evaluate the performance of the proposed regularizer we perform experiments on four datasets constructed for Football Player, Face, Bicycles, and Crowd detection. The Football Player dataset, [17], consists of 98,000 train and 10,000 test images of size  $32 \times 32$  that contain equal number

of football players and non-football players. The face dataset contains 70,000 train and 7,468 test images of size  $32 \times 32$  that contain equal number of face and non-face images. Images of faces have been randomly selected from the AFLW [18], MFL [19], and WIDER FACE [20] datasets. The Bicycles dataset, [17], contains 51,200 equally distributed train images of bicycles and non-bicycles, and correspondingly a test set of 10,000 images. Input images are of size  $64 \times 64$ . Finally, the Crowd-Drone dataset, [17], contains 40,000 UAV-captured train images of equal number of crowded scenes and non-crowded scenes, and 11,550 equally distributed crowded and non-crowded test images. Input images are of size  $64 \times 64$ .

### 4.2. Implementation Details

All the experiments conducted using the Caffe Deep Learning framework. We use the mini-batch gradient descent for the networks' training. That is, an update is performed for every mini-batch of  $N_b$  training samples. The learning rate is set to  $10^{-3}$  and drops to  $10^{-4}$  gradually, and the batch size is set to 256. The momentum is 0.9. All the models are trained on an NVIDIA GeForce GTX 1080 with 8GB of GPU memory for 100 epochs, and can run in real-time when deployed on an NVIDIA Jetson TX2. In this work, the parameter  $\eta$  in (13) which controls the relative importance of the proposed regularizer's loss, is set to 0.001, since we have seen that in most cases provides best performance. Best results are printed in bold.

### 4.3. Experimental Results

First, the evaluation results of the proposed models regarding the deployment speed are provided. The performance is tested on a low-power NVIDIA Jetson TX2 module with 8GB of memory, which is a state of the art GPU used for on-board UAV perception. Additionally, in order to accelerate the deployment speed and achieve real-time deployment, TensorRT<sup>1</sup> deep learning inference optimizer is utilized. TensorRT is a library that optimizes deep learning models providing FP32 (default) and FP16 optimizations for production deployments of various applications. In Table 1 we provide the detection speed in terms of fps for the two proposed models on the NVIDIA Jetson TX2 module without the utilization of the TensorRT optimizer, with the TensorRT on the default mode, and finally with TensorRT on the FP16 mode. As we can see TensorRT and in particular the FP16 mode significantly accelerates the proposed models, achieving detection in-real time for high-resolution images. To gain some intuition about the deployment speed, we note that state-of-the-art detectors run at notably fewer FPS on Jetson TX2, and also for lower resolution input images. For example, YOLO v.2

<sup>1</sup><https://developer.nvidia.com/tensorrt>

Input	Jetson TX2	TensorRT-FP32	TensorRT-FP16
32×32	12.3	16.9	25.7
64×64	8.8	18.5	25.6

Table 1: VGG-1080p: Speed (fps)

Dataset	Only Hinge Loss	MI Regularizer
Football Player	0.9568 ± 0.0100	<b>0.9744 ± 0.0100</b>
Face	0.8841 ± 0.0040	<b>0.8896 ± 0.0007</b>
Bicycles	0.9684 ± 0.0037	<b>0.9696 ± 0.0018</b>
Crowd - Drone	0.9194 ± 0.0082	<b>0.9303 ± 0.0076</b>

Table 2: Test Accuracy

[21] runs at 3.1 fps for input of size  $604 \times 604$ , while utilizing TensorRT (FP32) runs at 7.8 fps, and further speed up is achieved with the FP16 mode up to 14.4 fps, which remains far away from real-time even for lower input resolution. Finally, we should highlight that the deployment speed regards all the models, that is with and without the proposed regularizer, since the regularizer does not affect the deployment speed.

In Table 2 we present the mean value and the standard deviation of the test accuracy, for the considered training approaches, that is utilizing only hinge loss, and hinge loss with the proposed MI regularizer. Correspondingly, in Fig. 4 we illustrate the curves of mean test accuracy of the only hinge loss training against hinge loss & MI regularized training. We can see in the demonstrated results, that the proposed MI regularizer remarkably enhances the classification performance on all the utilized datasets.

Finally, we conducted a post-hoc Bonferroni test [22], for ranking the proposed regularization method and the only hinge loss training and evaluating the statistical significance of the obtained results. The performance of two methods is significantly different, if the corresponding average ranks over the datasets differ by at least the critical difference:

$$CD = q_\alpha \sqrt{\frac{m(m+1)}{6D}}, \quad (14)$$

where  $m$  is the number of methods compared,  $D$  is the number of datasets and critical values  $q_\alpha$  can be found in [22]. In our comparisons we set  $\alpha = 0.05$ . The number of datasets is four in the performed tests. The compared methods are two, that is the proposed regularizer is compared with a control method which is the only hinge loss training approach. The ranking results are illustrated in Fig. 5. The vertical axis depicts the two methods, while the horizontal axis depicts the performance ranking. The circles indicate the mean rank and the intervals around them indicate the confidence interval as this is determined by the  $CD$  value. Overlapping intervals between two methods indicate that there is not a statistically significant difference between the corresponding ranks, while non-

overlapping intervals indicate that the compared methods are significantly different. As we can observe, the proposed regularizer is significantly different against the only hinge loss training approach.

## 5. CONCLUSIONS

In this paper, we proposed regularized lightweight CNN models for addressing various binary classification tasks able to run in real-time on-drone. Furthermore, we proposed a novel regularizer motivated by the QMI, the so-called MI regularizer. The performance was evaluated on four datasets. The evaluation results validate the effectiveness of the proposed regularizer in enhancing the generalization ability of the proposed models.

## Acknowledgment

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR). This publication reflects the authors’ views only. The European Commission is not responsible for any use that may be made of the information it contains.

## 6. REFERENCES

- [1] Li Deng, “A tutorial survey of architectures, algorithms, and applications for deep learning,” *APSIPA Transactions on Signal and Information Processing*, vol. 3, pp. e2, 2014.
- [2] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al., “Recent advances in convolutional neural networks,” *Pattern Recognition*, vol. 77, pp. 354–377, 2018.
- [3] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.
- [4] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 0.5 mb model size,” *arXiv preprint arXiv:1602.07360*, 2016.
- [5] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun, “Shufflenet: An extremely efficient convolutional neural network for mobile devices,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6848–6856.

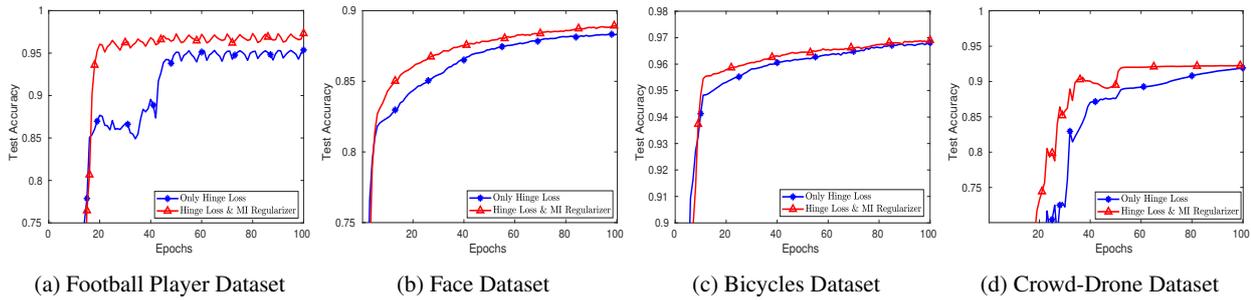


Fig. 4: MI Regularizer

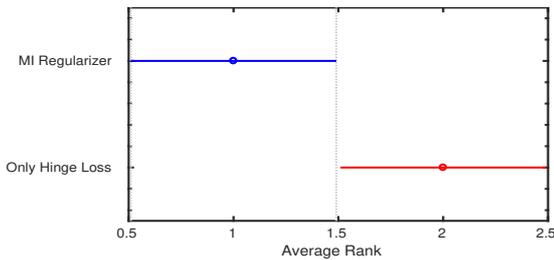


Fig. 5: Post-Hoc Bonferroni Test

- [6] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [7] Rich Caruana, “Multitask learning,” *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.
- [8] Jason Weston, Frédéric Ratle, and Ronan Collobert, “Deep learning via semi-supervised embedding,” in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1168–1175.
- [9] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu, “Deeply-supervised nets,” in *Artificial Intelligence and Statistics*, 2015, pp. 562–570.
- [10] Kari Torkkola, “Feature extraction by non-parametric mutual information maximization,” *Journal of machine learning research*, vol. 3, no. Mar, pp. 1415–1438, 2003.
- [11] Dimitrios Bouzas, Nikolaos Arvanitopoulos, and Anastasios Tefas, “Graph embedded nonparametric mutual information for supervised dimensionality reduction,” *IEEE transactions on neural networks and learning systems*, vol. 26, no. 5, pp. 951–963, 2015.
- [12] Nikolaos Passalis and Anastasios Tefas, “Learning deep representations with probabilistic knowledge transfer,” in *The European Conference on Computer Vision (ECCV)*, September 2018.
- [13] Karen Simonyan and Andrew Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [14] Claude Elwood Shannon, “A mathematical theory of communication,” *ACM SIGMOBILE mobile computing and communications review*, vol. 5, no. 1, pp. 3–55, 2001.
- [15] David W Scott, *Multivariate density estimation: theory, practice, and visualization*, John Wiley & Sons, 2015.
- [16] Shean-Tsong Chiu, “Bandwidth selection for kernel density estimation,” *The Annals of Statistics*, pp. 1883–1905, 1991.
- [17] Maria Tzelepi and Anastasios Tefas, “Graph embedded convolutional neural networks in human crowd detection for drone flight safety,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2019.
- [18] Martin Koestinger, Paul Wohlhart, Peter M Roth, and Horst Bischof, “Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization,” in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2144–2151.
- [19] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang, “Facial landmark detection by deep multi-task learning,” in *European Conference on Computer Vision*. Springer, 2014, pp. 94–108.
- [20] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang, “Wider face: A face detection benchmark,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5525–5533.
- [21] Joseph Redmon and Ali Farhadi, “Yolo9000: better, faster, stronger,” *arXiv preprint arXiv:1612.08242*, 2016.
- [22] Janez Demšar, “Statistical comparisons of classifiers over multiple data sets,” *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.

## **7.12 Multilinear Compressive Learning**

The appended paper follows.

# Multilinear Compressive Learning

Dat Thanh Tran\*, Mehmet Yamaç\*, Aysen Degerli\*, Moncef Gabbouj\*, Alexandros Iosifidis†

\*Department of Computing Sciences, Tampere University, Tampere, Finland

†Department of Engineering, Aarhus University, Aarhus, Denmark

Email:{thanh.tran,mehmet.yamac, aysen.degerli, moncef.gabbouj}@tuni.fi, alexandros.iosifidis@eng.au.dk

**Abstract**—Compressive Learning is an emerging topic that combines signal acquisition via compressive sensing and machine learning to perform inference tasks directly on a small number of measurements. Many data modalities naturally have a multi-dimensional or tensorial format, with each dimension or tensor mode representing different features such as the spatial and temporal information in video sequences or the spatial and spectral information in hyperspectral images. However, in existing compressive learning frameworks, the compressive sensing component utilizes either random or learned linear projection on the vectorized signal to perform signal acquisition, thus discarding the multi-dimensional structure of the signals. In this paper, we propose Multilinear Compressive Learning, a framework that takes into account the tensorial nature of multi-dimensional signals in the acquisition step and builds the subsequent inference model on the structurally sensed measurements. Our theoretical complexity analysis shows that the proposed framework is more efficient compared to its vector-based counterpart in both memory and computation requirement. With extensive experiments, we also empirically show that our Multilinear Compressive Learning framework outperforms the vector-based framework in object classification and face recognition tasks, and scales favorably when the dimensionalities of the original signals increase, making it highly efficient for high-dimensional multi-dimensional signals.

## I. INTRODUCTION

The classical sample-based signal acquisition and manipulation approach usually involve separate steps of signal sensing, compression, storing or transmitting, then the reconstruction. This approach requires the signal to be sampled above the Nyquist rate in order to ensure high-fidelity reconstruction. Since the existence of spatial-multiplexing cameras, over the past decade, Compressive Sensing (CS) [1] has become an efficient and a prominent approach for signal acquisition at sub-Nyquist rates, combining the sensing and compression step at the hardware level. This is due to the assumption that the signal often possesses specific structures that exhibit sparse or compressible representation in some basis, thus, can be sensed at a lower rate than the Nyquist rate but still allows almost perfect reconstruction [2], [3]. In fact, many data modalities that we operate on are often sparse or compressible. For example, smooth signals are compressible in the Fourier domain or subsequent frames in a video are piecewise smooth, thus compressible in a wavelet domain. With the efficient realization at the hardware level such as the popular Single Pixel Camera, CS becomes an efficient signal acquisition framework, however, making the signal manipulation an intim-

idating task. Indeed, over the past decade, since reversing the signal to its original domain is often considered the necessary step for signal manipulation, a significant amount of works have been dedicated to signal reconstruction, giving certain insights and theoretical guarantees for the successful recovery of the signal from compressively sensed measurements [2], [1], [3].

While signal recovery plays a major role in some sensing applications such as image acquisition for visual purposes, there are many scenarios in which the primary objective is the detection of certain patterns or inferring some properties in the acquired signal. For example, in many radar applications, one is often interested in anomaly patterns in the measurements, rather than signal recovery. Moreover, in certain applications [4], [5], signal reconstruction is undesirable since the step can potentially disclose private information, leading to the infringement of data protection legislation. These scenarios naturally led to the emergence of Compressive Learning (CL) concept [6], [7], [8], [9] in which the inference system is built on top of the compressively sensed measurements without the explicit reconstruction step. While the amount of literature in CL is rather insignificant compared to signal reconstruction in CS, different attempts have been made to modify the sensing component in accordance with the learning task [10], [11], to extract discriminative features [7], [12] from the randomly sensed measurements or to jointly optimize the sensing matrix [13], [14] and the subsequent inference system. Improvements to different components of CL pipeline have been proposed, however, existing frameworks utilize the same compressive acquisition step that performs a linear projection of the vectorized data, thereby operating on the vector-based measurements and thus losing the tensorial structure in the measurements of multi-dimensional data.

In fact, many data modalities naturally possess the tensorial format such as color images, videos or multivariate time-series. The multi-dimensional representation naturally reflects the semantic differences inherent in different dimensions or tensor modes. For example, the spatial and temporal dimensions in a video or the spatial and the spectral dimensions in hyperspectral images represent two different concepts, having different properties. Thus by exploiting this natural form of the signals and considering the semantic differences between different dimensions, many tensor-based signal processing, and learning algorithms have shown its superiority over the vector-based

approach, which simply operates on the vectorized data [15], [16], [17], [18], [19], [20], [21]. Indeed, tensor representation and its associated mathematical operations and properties have found various applications in the Machine Learning community. For example, in multivariate time-series analysis, the multilinear projection was utilized in [18], [22] to model the dependencies between data points along the feature and temporal dimension separately. Several multilinear regression [23], [24] or discriminant models [25], [26] have been developed to replace their linear counterparts, with improved performance. In neural network literature, multilinear techniques have been employed to compress pre-trained networks [27], [28], [29], or to construct novel neural network architectures [19], [30], [22].

It is worth noting that CS plays an important role in many applications that involves high-dimensional tensor signals because the standard point-based signal acquisition is both memory and computationally intensive. Representative examples include Hyperspectral Compressive Imaging (HCI), Synthetic Aperture Radar (SAR) imaging, Magnetic Resonance Imaging (MRI) or Computer Tomography (CT). Therefore, the tensor-based approach has also found its place in CS, also known as Multi-dimensional Compressive Sensing (MCS) [31], which replaces the linear sensing and reconstruction model with multilinear one. Similar to vector-based CS, thereupon simply referred to as CS, the majority of efforts in MCS are dedicated to constructing multilinear models that induce sparse representation along each tensor mode with respect to a set of bases. For example, the adoption of sparse Tucker representation and the Kronecker sensing scheme in MRI allows computationally efficient signal recovery with very low Peak Signal to Noise Ratio (PSNR) [31], [32]. In addition, the availability of optical implementations of separable sensing operators such as [33] naturally enables MCS, significantly reducing the amount of data collection and reconstruction cost.

While multilinear models have been successfully applied in Compressive Sensing and Machine Learning, to the best of our knowledge, we have not seen their utilization in Compressive Learning, which is the joint framework combining CS and ML. In this paper, in order to leverage the multi-dimensional structure in many data modalities, we propose Multilinear Compressive Learning framework, which adopts a multilinear sensing operator and a neural network classifier that is designed to utilize the multi-dimensional structure-preserving compressed measurements. The contribution of this paper is as follows:

- We propose Multilinear Compressive Learning (MCL), a novel CL framework that consists of a multilinear sensing module, a multilinear feature synthesis component, both taking into account the multi-dimensional property of the signals, and a task-specific neural network. The multilinear sensing module compressively senses along each separate mode of the original tensor signal, producing structurally encoded measurements. Similarly, the feature synthesis component performs the feature learning steps separately along each mode of the compressed measure-

ments, producing inputs to the subsequent task-specific neural network which has the structure depending on the inference problem.

- We show both theoretically and empirically that the proposed MCL framework is highly cost-effective in terms of memory and computational complexity. In addition, theoretical analysis and experimental results also indicate that our framework scales well when the dimensionalities of the original signal increases, making it highly efficient for high-dimensional tensor signals.
- We conduct extensive experiments in object classification and face recognition tasks to validate the performance of our framework in comparison with its vector-based counterpart. Besides, the effect of different components and hyperparameters in the proposed framework were also empirically analyzed.
- We publicly provide our implementation of the experiments reported in this paper to facilitate future research. By following our detailed instructions on how to set up the software environment, all experiment results can be reproduced in just one line of code.<sup>1</sup>

The remainder of the paper is organized as follows: in Section 2, we review the background information in Compressive Sensing, Multi-dimensional Compressive Sensing and Compressive Learning. In Section 3, the detailed description of the proposed Multilinear Compressive Learning framework is given. Complexity analysis and comparison with the vector-based framework are also given in Section 3. In Section 4, we provide details of our experiment protocols and quantitative analysis of different experiment configurations. Section 5 concludes our work with possible future research directions.

## II. RELATED WORK

### A. Notation

In this paper, we denote scalar values by either lower-case or upper-case characters ( $x, y, X, Y \dots$ ), vectors by lower-case bold-face characters ( $\mathbf{x}, \mathbf{y}, \dots$ ), matrices by upper-case or Greek bold-face characters ( $\mathbf{A}, \mathbf{B}, \mathbf{\Phi}, \dots$ ) and tensor as calligraphic capitals ( $\mathcal{X}, \mathcal{Y}, \dots$ ). A tensor with  $K$  modes and dimension  $I_k$  in the mode- $k$  is represented as  $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_K}$ . The entry in the  $i_k$ th index in mode- $k$  for  $k = 1, \dots, K$  is denoted as  $\mathcal{X}_{i_1, i_2, \dots, i_K}$ . In addition,  $vec(\mathcal{X})$  denotes the vectorization operation that rearranges elements in  $\mathcal{X}$  to the vector representation.

*Definition 1 (The Kronecker Product):* The Kronecker product between two matrices  $\mathbf{A} \in \mathbb{R}^{M \times N}$  and  $\mathbf{B} \in \mathbb{R}^{P \times Q}$  is denoted as  $\mathbf{A} \otimes \mathbf{B}$  having dimension  $MP \times NQ$ , is defined by:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} \mathbf{A}_{11}\mathbf{B} & \dots & \mathbf{A}_{1N}\mathbf{B} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{M1}\mathbf{B} & \dots & \mathbf{A}_{MN}\mathbf{B} \end{bmatrix} \quad (1)$$

*Definition 2 (Mode- $n$  Product):* The mode- $k$  product between a tensor  $\mathcal{X} = [x_{i_1, \dots, i_N}] \in \mathbb{R}^{I_1 \times \dots \times I_N}$  and a

<sup>1</sup><https://github.com/vieboy/MultilinearCompressiveLearningFramework>

matrix  $\mathbf{W} \in \mathbb{R}^{J_n \times I_n}$  is another tensor of size  $I_1 \times \dots \times J_n \times \dots \times I_N$  and denoted by  $\mathcal{X} \times_n \mathbf{W}$ . The element of  $\mathcal{X} \times_n \mathbf{W}$  is defined as  $[\mathcal{X} \times_n \mathbf{W}]_{i_1, \dots, i_{n-1}, j_n, i_{n+1}, \dots, i_N} = \sum_{i_n=1}^{I_n} [\mathcal{X}]_{i_1, \dots, i_{n-1}, i_n, \dots, i_N} [\mathbf{W}]_{j_n, i_n}$ .

The following relationship between the Kronecker product and  $n$ -mode product is the cornerstone in MCS:

$$\mathcal{Y} = \mathcal{X} \times_1 \mathbf{W}_1 \times \dots \times_N \mathbf{W}_N \quad (2)$$

can be written as

$$\mathbf{y} = (\mathbf{W}_1 \otimes \dots \otimes \mathbf{W}_N) \mathbf{x} \quad (3)$$

where  $\mathbf{y} = \text{vec}(\mathcal{Y})$  and  $\mathbf{x} = \text{vec}(\mathcal{X})$

### B. Compressive Sensing

Compressive Sensing (CS) [1] is a signal acquisition and manipulation paradigm that performs simultaneous sensing and compression on the hardware level, leading to large reduction in computation cost and the number of measurements. The signal  $\mathbf{y}$  working under CS is assumed to have a sparse or compressible representation  $\mathbf{x}$  in some basis or dictionary  $\Psi \in \mathbb{R}^{I \times I}$ , that is:

$$\mathbf{y} = \Psi \mathbf{x} \quad \text{with} \quad \|\mathbf{x}\|_0 \leq K \quad \text{and} \quad K \ll I \quad (4)$$

where  $\|\mathbf{x}\|_0$  denotes the number of non-zero entries in  $\mathbf{x}$ . While the dictionary presented in Eq. (4) is complete, i.e., the number of columns in  $\Psi$  is equal to the signal dimension  $I$ , we should note that signal models with over-complete dictionaries can also work, i.e.,  $\Psi \in \mathbb{R}^{J \times I}$  with some modifications [34].

With the assumption on the sparsity, CS performs the linear sensing step using the sensing operator  $\Phi \in \mathbb{R}^{M \times I}$ , acquiring a small number of measurements  $\mathbf{z} \in \mathbb{R}^M$  with  $M < I$ , from analog signal  $\mathbf{y}$ :

$$\mathbf{z} = \Phi \mathbf{y} \quad (5)$$

Eq. (5) represents both the sensing and compression step that can be efficiently implemented at the sensor level. Thus, what we obtain from CS sensors is a limited number of measurements  $\mathbf{z}$  that is used for other processing steps. By combining Eq. (4, and 5), the CS model is usually expressed as:

$$\mathbf{z} = \Phi \Psi \mathbf{x} \quad \text{with} \quad \|\mathbf{x}\|_0 \leq K \quad \text{and} \quad K \ll I \quad (6)$$

In some applications, we are interested in recovering the signal  $\mathbf{y}$  from  $\mathbf{z}$ . This involves developing theoretical properties and algorithms to determine the sensing operator  $\Phi$ , the dictionary or basis  $\Psi$ , and the number of nonzero coefficients  $K$  in order to ensure that the reconstruction is unique, and of high-fidelity [2], [35], [3]. The reconstruction of  $\mathbf{y}$  is often posed as finding the sparsest solution of the under-determined linear system [36], particularly:

$$\arg \min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{s.t.} \quad \|\mathbf{z} - \Phi \Psi \mathbf{x}\|_2 \leq \epsilon \quad (7)$$

where  $\epsilon$  is a small constant specifying the amount of residual error allowed in the approximation. A large body of research has been dedicated to solve the problem in Eq. (7) and its variants with two main approaches: *basis pursuit* (BP) which transforms Eq. (7) to a convex one to be solved by linear programming [37] or second-order cones programs [2], and *matching pursuit* (MP), a class of greedy algorithms, which iteratively refines the solution to the sparsest [38], [39]. Both BP and MP algorithms are computationally intensive when the number of elements in  $\mathbf{y}$  is big, especially in the case of multi-dimensional signals.

### C. Multi-dimensional Compressive Sensing

Given a multi-dimensional signal  $\mathcal{Y} \in \mathbb{R}^{I_1 \times \dots \times I_N}$ , a direct application of the sparse representation in Eq. (4) requires vectorizing  $\mathbf{y} = \text{vec}(\mathcal{Y})$  and the calculations on  $\Phi \Psi \in \mathbb{R}^{M \times (I_1 \dots I_N)}$ , which is a very big matrix with the number of elements scales exponentially with  $N$ . Instead of assuming  $\text{vec}(\mathcal{Y})$  is sparse in some basis or dictionary, MCS adopts a sparse Tucker model [40] as follows:

$$\mathcal{Y} = \mathcal{X} \times_1 \Psi_1 \times \dots \times_N \Psi_N \quad (8)$$

which assumes that the signal  $\mathcal{Y}$  is sparse with respect to a set of bases or dictionaries  $\Psi_n, n = 1, \dots, N$ . Since in some cases, the sensing step can be taken in a multilinear way, i.e., by using a set of linear operators along each mode separately, also known as separable sensing operators:

$$\mathcal{Z} = \mathcal{Y} \times_1 \Phi_1 \times \dots \times_N \Phi_N \quad (9)$$

that allows us to obtain the measurements  $\mathcal{Z}$  with retained multi-dimensional structure. From Eq. (2, 3, 8 and 9), the MCS model is often expressed as:

$$\mathbf{z} = (\mathbf{B}_1 \otimes \dots \otimes \mathbf{B}_N) \mathbf{x} \quad \text{with} \quad \|\mathbf{x}\|_0 \leq K \quad (10)$$

where  $\mathbf{z} = \text{vec}(\mathcal{Z})$ , and  $\mathbf{B}_n = \Phi_n \Psi_n$  ( $n = 1, \dots, N$ ). The formulation in Eq. (10) is also known as Kronecker CS [41].

Since MCS can be expressed in the vector form, the existing algorithms and theoretical bounds for vector-based CS have also been extended for MCS. Representative examples include Kronecker OMP and its tensor block-sparsity extension [42] that improves the computation significantly. It is worth noting that by adopting a multilinear structure, MCS operates with a set of smaller sensing and dictionaries, requiring much lower memory and computation compared to the vectorization approach [31].

### D. Compressive Learning

The idea of learning directly from the compressed measurements dates back to the early work of [7] in which the authors proposed a framework termed *compressive classification* which introduces the concept of *smashed filters* and operates directly on the compressive measurements without reconstruction as the first proxy step. The result in [7] was subsequently strengthened in [43] showing that when sufficiently large

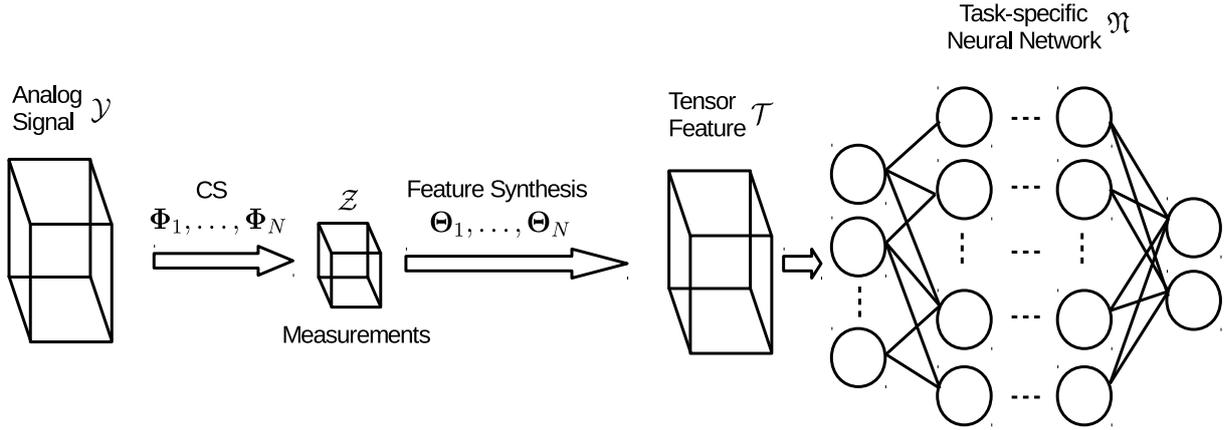


Fig. 1. Illustration of the proposed Multilinear Compressive Learning framework

random sensing matrix is used, it can capture the structure of the data manifold. Later, further extensions that extract discriminative features from compressive measurements for activity recognition [44], [45] or face recognition [12] have also been proposed.

The concept of CL was introduced in [6], which provides theoretical analysis illustrating that learning machines can be built directly in the compressed domain. Particularly, given certain conditions of the sensing matrix  $\Phi$ , the performance of a linear Support Vector Machine (SVM) trained on compressed measurements is as good as the best linear threshold classifier trained on the original signal  $\mathbf{y}$ . Later, for compressive learning of signals described by a Gaussian Mixture Model, asymptotic behavior of the upper-bound [9] and its extension [11] to learn the sensing matrix were also derived.

The idea of jointly optimizing the sensing matrix with the classifier was also adopted in [10] in which the authors proposed an adaptive version of *feature-specific imaging* system to learn an optimal sensing matrix based on past measurements. With the advances in computing hardware and stochastic optimization techniques, end-to-end CL system was proposed in [13], and several follow-up extensions and applications [46], [47], [48], indicating the superior performance when simultaneously optimizing the sensing component and the classifier via task-specific data. Our work is closely related to the end-to-end CL system in [13] in that we also optimize the CL system via stochastic optimization in an end-to-end manner. Different from [13], our proposed framework efficiently utilizes the tensor structure inherent in many types of signals, thus outperforming the approach in [13] in both inference performance and computational efficiency.

### III. MULTILINEAR COMPRESSIVE LEARNING FRAMEWORK

In this Section, we first give our description of the proposed Multilinear Compressive Learning (MCL) framework that op-

erates directly on the tensor representation of the signals. Then, the initialization scheme and optimization procedures of the proposed framework is discussed. Lastly, theoretical analysis of the framework's complexity in comparison with its vector-based counterpart is provided.

#### A. Motivation

In order to model the multi-dimensional structure in the signal of interest, we assume that the discriminative structure in  $\mathcal{Y} \in \mathbb{R}^{I_1 \times \dots \times I_N}$  can be captured in a lower-dimensional multilinear subspace  $\mathfrak{F} \subset \mathbb{R}^{J_1 \times \dots \times J_N}$  of  $\mathbb{R}^{I_1 \times \dots \times I_N}$  with ( $J_n < I_n, \forall n = 1, \dots, N$ ):

$$\mathcal{Y} = \bar{\mathcal{X}} \times_1 \bar{\Psi}_1 \times \dots \times_N \bar{\Psi}_N \quad (11)$$

where  $\bar{\Psi}_n \in \mathbb{R}^{I_n \times J_n}, \forall n = 1, \dots, N$  denotes the factor matrices and  $\bar{\mathcal{X}} \in \mathbb{R}^{J_1 \times \dots \times J_N}$  is the signal representation in this multilinear subspace.

Here we should note that although Eq. (11) in our framework and Eq. (8) in MCS look similar in its mathematical form, the assumption and motivation are different. The objective in MCS is to reconstruct the signal  $\mathcal{Y}$  by assuming the existence of the set of sparsifying dictionaries or bases  $\Psi_n$  and optimizing  $\Psi_n$  to induce the sparsest  $\mathcal{X}$ . Since our objective is to learn a classification or regression model, we make no assumption or constraint on the sparsity of  $\bar{\mathcal{X}}$  but assume that the factorization in Eq. (11) can lead to a tensor subspace  $\mathfrak{F}$  in which the representation  $\bar{\mathcal{X}}$  is discriminative or meaningful for the learning problem.

As mentioned in the previous Section, in some applications, the measurements can be taken in a multilinear fashion, with different linear sensing operators operating along different tensor modes, i.e., separable sensing operators, we obtain the measurements  $\mathcal{Z}$  from the following sensing equation:

$$\mathcal{Z} = \mathcal{Y} \times_1 \Phi_1 \times \dots \times_N \Phi_N \quad (12)$$

where  $\Phi_n \in \mathbb{R}^{M_n \times I_n}$  ( $n = 1, \dots, N$ ) represent the sensing matrices of those linear operators.

In cases where the measurements of the multi-dimensional signals are taken in a vector-based fashion, i.e., the following sensing model:

$$\mathbf{z} = \Phi \text{vec}(\mathcal{Y}) \quad (13)$$

with a single sensing operator  $\Phi \in \mathbb{R}^{M \times I_1 \times \dots \times I_N}$ , we can still enforce a structure-preserving sensing operation similar to the multilinear sensing scheme in Eq. (12) by setting:

$$\Phi = \Phi_1 \otimes \dots \otimes \Phi_N \quad (14)$$

to obtain  $\mathcal{Z}$  in Eq. (12) from  $\mathbf{z}$  in Eq. (13).

Combining Eq. (11 and 12), we can express our measurements  $\mathcal{Z}$  as:

$$\mathcal{Z} = \bar{\mathcal{X}} \times_1 (\Phi_1 \bar{\Psi}_1) \times \dots \times_N (\Phi_N \bar{\Psi}_N) \quad (15)$$

By setting the sensing matrices  $\Phi_n$  to be pseudo-inverse of  $\bar{\Psi}_n$  for all  $n = 1, \dots, N$ , we obtain the measurements  $\mathcal{Z}$  that lie in the discriminative-induced tensor subspace  $\mathfrak{F}$  mentioned previously.

### B. Design

Figure 1 illustrates our proposed MCL framework which consists of the following components:

- CS component: the data acquisition step of the multi-dimensional signals is done via separable linear sensing operators  $\Phi_n, n = 1, \dots, N$ . As mentioned previously, in cases where the actual hardware implementation only allows vector-based sensing scheme, Eq. (14) allows the simulation of this multilinear sensing step. This component produces measurements  $\mathcal{Z}$  with encoded tensor structure, having the same number of tensor modes ( $N$ ) as the original signal.
- Feature Synthesis (FS) component: from  $\mathcal{Z}$ , this step performs feature extraction along  $N$  modes of the measurements  $\mathcal{Z}$  with the set of learnable matrices  $\Theta_n$ . Since the measurements typically have many fewer elements compared to the original signal  $\mathcal{Y}$ , the FS component expands the dimensions of  $\mathcal{Z}$ , allowing better separability between the sensed signals from different classes in a higher multi-dimensional space that is found through optimization. While the sensing step performs linear interpolations for computational efficiency, the FS component can be either multilinear or nonlinear transformations. A typical nonlinear transformation step is to perform zero-thresholding, i.e., ReLU, on  $\mathcal{Z}$  before multiplying with  $\Theta_n, n = 1 \dots, N$ , i.e.,  $\text{ReLU}(\mathcal{Z}) \times_1 \Theta_1 \times \dots \times_N \Theta_N$ . In applications which require the transmission of  $\mathcal{Z}$  to be analyzed, this simple thresholding step can, before transmission, increase the compression rate by sparsifying the encoded signal and discarding the sign bits. While nonlinearity is often considered beneficial for neural networks, adding the thresholding step as described above

further restricts the information retained in a limited number of measurements  $\mathcal{Z}$ , thus, adversely affects the inference system. In the Experiments Section, we provide empirical analysis on the effect of nonlinearity towards the inference tasks at different measurement rates. Here we should note that while our FS component resembles the reprojection step in the vector-based framework [13], our FS and CS components have different weights ( $\Theta_n$  and  $\Phi_n, n = 1, \dots, N$ ) and the dimensionality of the tensor feature  $\mathcal{T}$  produced by FS component is task-dependent, and is not constrained to that of the original signal.

- Task-specific Neural Network  $\mathfrak{N}$ : from the tensor representation  $\mathcal{T}$  produced by FS step, a neural network with task-dependent architecture is built on top to generate the regression or classification outputs. For example, when analyzing visual data, the  $\mathfrak{N}$  can be a Convolutional Neural Network (CNN) in case of static images or a Convolutional Recurrent Neural Network in case of videos. In CS applications that involve distributed arrays of sensors that continuously collect data, specific architectures for time-series analysis such as Long-Short Term Memory Network should be considered for  $\mathfrak{N}$ . Here we should note that the size of  $\mathcal{T}$  is also task-dependent and should match with the neural network component. For example, in object detection and localization task, it is desirable to keep the spatial aspect ratio of  $\mathcal{T}$  similar to  $\mathcal{Y}$  to allow precise localization.

### C. Optimization

In our proposed MCL framework, we aim to optimize all three components, i.e.,  $\Phi_n$ ,  $\Theta_n$  and  $\mathfrak{N}$ , with respect to the inference task. A simple and straightforward approach is to consider all components in this framework as a single computation graph, then randomly initialize the weights according to some popular initialization scheme [49], [50] and perform stochastic gradient descent on this graph with respect to the loss function defined by the learning task. However, this approach does not take into account any existing domain knowledge of each component that we have.

As mentioned in Section III.A, with the assumption of the existence of a tensor subspace  $\mathfrak{F}$  and the factorization in Eq. (11), the sensing matrix  $\Phi_n$  in the CS component can be initialized equal to the pseudo-inverse of  $\bar{\Psi}_n$  for all  $n = 1, \dots, N$  to obtain initial  $\mathcal{Z}$  that are discriminative or meaningful. There have been several algorithms proposed to learn the factorization in Eq. (11) with respect to different criteria such as the multi-class discriminant [25], class-specific discriminant [26], max-margin [51] or Tucker Decomposition with non-negative constraint [52].

In a general setting, we propose to apply Higher Order Singular Value Decomposition (HOSVD) [40] and initialize  $\Phi_n$  with the left singular vectors that correspond to the largest singular values in mode  $n$ . The sensing matrices are then adjusted together with other components during the stochastic optimization process. This initialization scheme resembles the

TABLE I  
COMPLEXITY OF THE PROPOSED MCL FRAMEWORK AND VECTOR-BASED FRAMEWORK [13]

	Our	Vector [13]
Memory	$O(2 \sum_{n=1}^N I_n * M_n)$	$O(\prod_{n=1}^N I_n * M_n)$
Computation	$O(\sum_{n=1}^N (\prod_{p=1}^n M_p * \prod_{k=n}^N I_k) + \sum_{n=1}^N (\prod_{p=1}^n I_p * \prod_{k=n}^N M_k))$	$O(2 \prod_{n=1}^N I_n * M_n)$

one proposed for vector-based CL framework which utilizes Principal Component Analysis (PCA). In a general case where one has no prior knowledge on the structure of  $\mathfrak{F}$ , a transformation that retains the most energy in the signal such as PCA or HOSVD is a popular choice when reducing dimensionalities of the signal. While for higher-order data, HOSVD only provides a quasi-optimal condition for data reconstruction in the least-square sense [53], since our objective is to make inferences, this initialization scheme works well as indicated in our Experiments Section.

With the aforementioned initialization scheme of CS component for a general setting, it is natural to also initialize  $\Theta_n$  in FS component with the right singular vectors corresponding to the largest singular values in mode  $n$  of the training data. With this initialization of  $\Theta_n$ , during the initial forward steps in stochastic gradient descent, the FS component produces an approximate version of  $\mathcal{Y}$ , and in cases where a classifier  $\mathcal{C}$  pre-trained on  $\mathcal{Y}$  or its approximated version  $\hat{\mathcal{Y}}$  exists, the weights of neural network  $\mathfrak{N}$  can be initialized with that of  $\mathcal{C}$ . It is worth noting that the reprojection step in the vector-based framework in [13] shares the weights with the sensing matrices, performing inexplicit signal reconstruction while we have different sensing  $\Phi_n, n = 1, \dots, N$  and feature extraction  $\Theta_n, n = 1, \dots, N$  weights. Since the vector-based framework involves large sensing and reprojection matrices, from the optimization point of view, enforcing shared weights might be essential in their framework to reduce overfitting as indicated by their empirical results.

After performing the aforementioned initialization steps, all three components in our MCL framework are optimized using Stochastic Gradient Descent method. It is worth noting that above initialization scheme for CS and FS component is proposed in a generic setting, which can serve as a good starting point. In cases where certain properties of the tensor subspace  $\mathfrak{F}$  or the tensor feature  $\mathcal{T}$  are known to improve the learning task, one might adopt a different initialization strategy for CS and FS components to induce such properties.

#### D. Complexity Analysis

Since the complexity of the neural network component  $\mathfrak{N}$  varies with the choice of the architecture, we will estimate the theoretical complexity for the CS and FS component and make comparison with the vector-based framework [13]. Let  $\mathbb{R}^{I_1 \times \dots \times I_N}$  and  $\mathbb{R}^{M_1 \times \dots \times M_N}$  denote the dimensionality of the original signal  $\mathcal{Y}$  and its measurements  $\mathcal{Z}$ , respectively. In addition, to compare with the vector-based framework, we also assume that the dimensionality of the feature  $\mathcal{T}$

is also  $\mathbb{R}^{I_1 \times \dots \times I_N}$ . Thus,  $\Phi_n$  belongs to  $\mathbb{R}^{M_n \times I_n}$  and  $\Theta_n$  belongs to  $\mathbb{R}^{I_n \times M_n}$  for  $n = 1, \dots, N$  in our CS and FS component, while in [13], the sensing matrix  $\Phi$  and the reconstruction matrix  $\Phi^T$  belong to  $\mathbb{R}^{(I_1 * \dots * I_N) \times (M_1 * \dots * M_N)}$  and  $\mathbb{R}^{(M_1 * \dots * M_N) \times (I_1 * \dots * I_N)}$ , respectively.

It is clear that the memory complexity of CS and FS component in our MCL framework is  $O(2 \sum_{n=1}^N I_n * M_n)$ , and that of the vector-based framework is  $O(\prod_{n=1}^N I_n * M_n)$ . To see the huge difference between the two frameworks, let us consider 3D MRI image of size  $I_1 \times I_2 \times I_3 = 256 \times 256 \times 64$  with the sampling ratio 70%, i.e.,  $M_1 \times M_2 \times M_3 = 214 \times 214 \times 64$ , the memory complexity in our framework is  $O(256 * 214 + 256 * 214 + 64 * 64) \approx O(10^5)$  while that of the vector-based framework is  $O(256 * 214 * 256 * 214 * 64 * 64) = O(10^{13})$ .

Regarding computational complexity of our framework, the CS component performs  $\mathcal{Z} = \mathcal{Y} \times_1 \Phi_1 \times \dots \times_N \Phi_N$  having complexity of  $O(\sum_{n=1}^N (\prod_{p=1}^n M_p * \prod_{k=n}^N I_k))$ , and the FS component performs  $\mathcal{T} = \mathcal{Z} \times_1 \Theta_1 \times \dots \times_N \Theta_N$  having complexity of  $O(\sum_{n=1}^N (\prod_{p=1}^n I_p * \prod_{k=n}^N M_k))$ . For the vector-based framework, the sensing step computes  $\mathbf{z} = \Phi \text{vec}(\mathcal{Y})$  and reprojection step computes  $\Phi^T \mathbf{z}$ , resulting in total complexity of  $O(2 \prod_{n=1}^N I_n * M_n)$ . With the same 3D MRI example as in the previous paragraph, the total computational complexity of our framework is  $O(10^9)$  while that of the vector-based framework is  $O(10^{13})$ .

Table I summarizes the complexity of the two frameworks. It is worth noting that by taking into account the multi-dimensional structure of the signal, the proposed framework has both memory and computational complexity several orders of magnitudes lower than its vector-based counterpart.

## IV. EXPERIMENTS

In this section, we provide a detailed description of our empirical analysis of the proposed MCL framework. We start by describing the datasets and the experiments' protocols that have been used. In the standard set of experiments, we analyze the performance of MCL in comparison with the vector-based framework proposed in [13], [54]. We further investigate the effect of different components in our framework in the Ablation Study Subsection.

#### A. Datasets and Experiment Protocol

We have conducted experiments on the object classification and face recognition tasks on the following datasets:

- CIFAR-10 and CIFAR-100: CIFAR dataset [55] is a color (RGB) image dataset for evaluating object recognition task. The dataset consists of 50K images for training and

10K images for testing with resolution  $32 \times 32$  pixels. CIFAR-10 refers to the 10-class objection recognition task in which each individual image has a single class label coming from 10 different categories. Likewise, CIFAR-100 refers to a more fine-grained classification task with each image having a label coming from 100 different categories. In our experiment, from the training set of CIFAR-10 and CIFAR-100, we randomly selected 5K images for validation purpose and only trained the algorithms on 45K images.

- CelebA: CelebA [56] is a large-scale face attributes dataset with more than 200K images at different resolutions from more than 10K identities. In our experiment, we used a subset of 100 identities in this dataset which corresponds to 7063, 2373, and 2400 samples for training, validation, and testing, respectively. In order to evaluate the scalability of our proposed framework, we resized the original images to different set of resolutions, including:  $32 \times 32$ ,  $48 \times 48$ ,  $64 \times 64$ , and  $80 \times 80$  pixels, which are subsequently denoted as CelebA-32, CelebA-48, CelebA-64, and CelebA-80, respectively.

In our experiments, two types of network architecture have been employed for the neural network component  $\mathfrak{N}$ : the AllCNN architecture [57] and the ResNet architecture [58]. AllCNN is a simple 9-layer feed-forward architecture which has no max-pooling (pooling is done via convolution with stride more than 1) and no fully-connected layer. ResNet is a 110-layer CNN with residual connections. The exact topologies of AllCNN and ResNet in our experiment can be found in our publicly available implementation<sup>2</sup>.

Since all of the datasets contain RGB images, we followed the implementation proposed in [54] for the vector-based framework, which is an extension of [13], which has 3 different sensing matrices for each of the color channel, and the corresponding reprojection matrices are enforced to share weights with the sensing matrices. The sensing matrices in MCL were initialized with the HOSVD decomposition on the training sets while the sensing matrices in the vector-based framework were initialized with PCA decomposition on the training set. Likewise, the bases obtained from HOSVD and PCA were also used to initialize the FS component in our framework and the reprojection matrices in the vector-based framework. In addition, we also trained the neural network component  $\mathfrak{N}$  on uncompressed data with respect to the learning tasks and initialized the classifier in each framework with these pre-trained networks’ weights. After the initialization step, both frameworks were trained in an end-to-end manner.

All algorithms were trained with ADAM optimizer [59] with the following learning rate the schedule  $\{10^{-3}, 10^{-4}, 10^{-5}\}$ , changing at epoch 80 and 120. Each algorithm was trained for 160 epochs in total. Weight decay coefficient was set to 0.0001 to regularize all the trainable weights in all experiments. We performed no data preprocessing step, except scaling

TABLE II  
DIFFERENT CONFIGURATIONS OF MEASUREMENTS BETWEEN VECTOR-BASED FRAMEWORK AND OUR FRAMEWORK. \* *Measurement Rate* IS CALCULATED WITH RESPECT TO THE ORIGINAL SIGNAL OF SIZE  $32 \times 32 \times 3$

Type	Configuration	#measurements	Measurement Rate
vector [54]	$256 \times 3$	768	0.250
MCL (our)	$20 \times 19 \times 2$	760	0.247
MCL (our)	$28 \times 27 \times 1$	756	0.246
vector [54]	$102 \times 3$	306	0.100
MCL (our)	$14 \times 11 \times 2$	308	0.100
MCL (our)	$18 \times 17 \times 1$	306	0.100
vector [54]	$18 \times 3$	54	0.018
MCL (our)	$9 \times 6 \times 1$	54	0.018
MCL (our)	$6 \times 9 \times 1$	54	0.018

all the pixel values to  $[0, 1]$ . In addition, data augmentation was employed by random flipping on the horizontal axis and image shifting within 10% of the spatial dimensions. In all experiments, the final model weights which are used to measure the performance on the test sets, are obtained from the epoch which has the highest validation accuracy.

For each experiment configuration, we performed 3 runs and the mean and standard deviation of test accuracy are reported.

### B. Comparison with the vector-based framework

In order to compare with the vector-based framework in [13], [54], we performed experiments on 3 datasets: CIFAR-10, CIFAR-100, and CelebA-32. To compare the performances at different measurement rates, we employed three different measurement values  $Z$  for the vector-based framework:  $256 \times 3 = 768$ ,  $102 \times 3 = 306$ , and  $18 \times 3 = 54$ . Here  $\times 3$  indicates that the vector-based framework has 3 different sensing matrices for each color channel. Since we cannot always select the size of the measurements  $Z$  in MCL to match the number of measurements in the vector-based framework, we try to find the configurations of  $Z$  that closely match with the vector-based ones. In addition, with a target number of measurements, there can be more than one configuration of  $Z$  that yields a similar number of measurements. For each measurement value (768, 102, 54) in the vector-based framework, we evaluated two different values of  $Z$ , particularly, the following sizes of  $Z$  were used:  $20 \times 19 \times 2 = 760$ ,  $28 \times 27 \times 1 = 756$ ,  $14 \times 11 \times 2 = 308$ ,  $18 \times 17 \times 1 = 306$ ,  $6 \times 9 \times 1 = 54$  and  $9 \times 6 \times 1 = 54$ . The measurement configurations are summarized in Table II.

In order to effectively compare the CS and FS component in MCL with those in [54], two different neural network architectures with different capacities have been used. Table III and IV show the accuracy on the test set with AllCNN and ResNet architecture, respectively. The second row of each table shows the performance of the base classifier on the uncompressed data, which we term as *Oracle*.

It is clear that our proposed framework outperforms the vector-based framework in all compression rates and datasets

<sup>2</sup><https://github.com/vieboy/MultilinearCompressiveLearningFramework>

with both AllCNN and ResNet architecture, except for CIFAR-100 dataset at the lowest measurement rate (0.018). The performance gaps between the proposed MCL framework and the vector-based one are huge, with more than 10% differences for the CIFAR datasets at measurement rates 0.25 and 0.10. In case of CelebA-32 dataset and at measurement rate 0.246 (configuration  $28 \times 27 \times 1$ ), the inference systems learned by our proposed framework even slightly outperform the Oracle setting for both AllCNN and ResNet architecture.

Although the capacities of AllCNN and ResNet architecture are different, their performances on the uncompressed data are roughly similar. Regarding the effect of two different base classifiers in the two Compressive Learning pipelines, it is clear that the optimal configurations of our framework at each measurement rate are consistent between the two classifiers, i.e., the bold patterns from both Table III and IV are similar. When switching from AllCNN to ResNet, the vector-based framework observes performance drop at the highest measurement rate (0.25), but increases in lower rates (0.1 and 0.018). For our framework when switching from AllCNN to ResNet, the test accuracies stay approximately similar or improve.

Table V shows the empirical complexity of both frameworks with respect to different measurement configurations, excluding the base classifiers. Since all three datasets employed in this experiment have the same input size and the size of the feature tensor  $\mathcal{T}$  in MCL was set similar to the original input size, the complexities of CS and FS components in all three datasets are similar. It is clear that our proposed MCL framework has much lower memory and computational complexity compared to the vector-based counterpart. *In our proposed framework, even operating at the highest measurement rate 0.247, the CS and FS components require only 2.5K parameters and 5K FLOPs, which are approximately 20 times fewer than that of the vector-based framework operating at the lowest measurement rate 0.018.* Interestingly, the optimal configuration at each measurement rate obtained in our framework also has lower or similar complexity than the other configuration.

In Figure 2, we visualize the features obtained from the reprojection step and the FS component in the proposed framework, respectively. It is worth noting that the sensing matrices and the reprojection matrices (in case of the vector-based framework) or  $\Theta_n$  (in FS component of MCL framework) were initialized with PCA and HOSVD. In addition, the base network classifiers were also initialized with the ones trained on the original data. Thus, it is intuitive to expect the features obtained from both frameworks to be visually interpretable for human, despite no explicit reconstruction objective was incorporated during the training phase. Indeed, from Figure 2, we can see that with the highest number of measurements, the feature images obtained from both frameworks look very similar to the original images. Particularly, the ones synthesized by the vector-based framework look visually closer to the original images than those obtained from our MCL framework. Since the sensing and reprojection steps in

TABLE III  
TEST ACCURACY WITH ALLCNN ARCHITECTURE AS THE BASE CLASSIFIER

Configuration	CIFAR-10	CIFAR-100	CelebA-32
Oracle	92.33	72.25	92.58
$256 \times 3$ [54]	81.36 $\pm$ 00.00	55.32 $\pm$ 00.00	89.25 $\pm$ 00.00
$20 \times 19 \times 2$ (our)	<b>89.35</b> $\pm$ 00.26	<b>65.97</b> $\pm$ 00.19	92.36 $\pm$ 00.07
$28 \times 27 \times 1$ (our)	88.56 $\pm$ 00.14	62.82 $\pm$ 00.09	<b>92.74</b> $\pm$ 00.31
$102 \times 3$ [54]	65.14 $\pm$ 02.37	44.03 $\pm$ 03.60	67.04 $\pm$ 02.36
$14 \times 11 \times 2$ (our)	<b>84.15</b> $\pm$ 00.55	<b>59.77</b> $\pm$ 00.12	87.01 $\pm$ 00.99
$18 \times 17 \times 1$ (our)	83.17 $\pm$ 00.32	54.96 $\pm$ 00.17	<b>91.26</b> $\pm$ 00.13
$18 \times 3$ [54]	61.38 $\pm$ 00.05	<b>37.78</b> $\pm$ 00.08	63.76 $\pm$ 00.22
$9 \times 6 \times 1$ (our)	<b>64.45</b> $\pm$ 00.39	34.74 $\pm$ 00.29	<b>68.49</b> $\pm$ 00.28
$6 \times 9 \times 1$ (our)	64.28 $\pm$ 00.35	35.16 $\pm$ 00.16	65.92 $\pm$ 00.77

TABLE IV  
TEST ACCURACY WITH RESNET ARCHITECTURE AS THE BASE CLASSIFIER

Configuration	CIFAR-10	CIFAR-100	CelebA-32
Oracle	92.47	72.38	93.08
$256 \times 3$ [54]	78.56 $\pm$ 00.00	53.03 $\pm$ 00.00	87.29 $\pm$ 00.00
$20 \times 19 \times 2$ (our)	<b>89.22</b> $\pm$ 00.27	<b>67.21</b> $\pm$ 00.18	92.00 $\pm$ 00.48
$28 \times 27 \times 1$ (our)	88.24 $\pm$ 00.15	63.37 $\pm$ 00.44	<b>93.54</b> $\pm$ 00.32
$102 \times 3$ [54]	67.65 $\pm$ 02.99	47.90 $\pm$ 01.22	76.32 $\pm$ 02.35
$14 \times 11 \times 2$ (our)	<b>84.74</b> $\pm$ 00.16	<b>60.30</b> $\pm$ 00.21	88.50 $\pm$ 00.26
$18 \times 17 \times 1$ (our)	83.31 $\pm$ 00.21	55.51 $\pm$ 00.08	<b>90.82</b> $\pm$ 00.14
$18 \times 3$ [54]	61.96 $\pm$ 00.17	<b>41.03</b> $\pm$ 00.22	67.29 $\pm$ 00.44
$9 \times 6 \times 1$ (our)	<b>64.14</b> $\pm$ 00.24	33.67 $\pm$ 00.17	<b>69.90</b> $\pm$ 00.41
$6 \times 9 \times 1$ (our)	64.07 $\pm$ 00.16	32.40 $\pm$ 01.62	67.39 $\pm$ 00.34

the vector-based framework share the same weight matrices during the optimization procedure, the whole pipeline is more constrained to reconstruct the images at the reprojection step.

When the number of measurements drops to approximately 10% of the original signal, the reverse scenario happens: the feature images (in configuration  $14 \times 11 \times 2$ ,  $28 \times 27 \times 1$ ) obtained from our framework retain more facial features compared to those from the vector-based framework ( $102 \times 3$ ), especially in the  $28 \times 27 \times 1$  configuration. This is due to the fact that most of the facial information in particular, and natural images in general lie on the spatial dimensions, i.e., height and width. Besides, when the dimension of the third mode of the measurement  $\mathcal{Z}$  is set to 1 (as in configuration  $28 \times 27 \times 1$ ,  $18 \times 17 \times 1$ ), after the optimization procedure, our proposed framework effectively discards the color information which is less relevant to the facial recognition task, and retains more lightness details, thus, performs better than the configurations with the 3-mode dimension set to 2 (in configuration  $20 \times 19 \times 2$ ,  $14 \times 11 \times 2$ ).

With the above observations from the empirical analysis, it is clear that structure-preserving Compressive Sensing and Feature Synthesis components in our proposed MCL framework can better capture essential information inherent in the multi-dimensional signal for the learning tasks, compared with the vector-based framework.

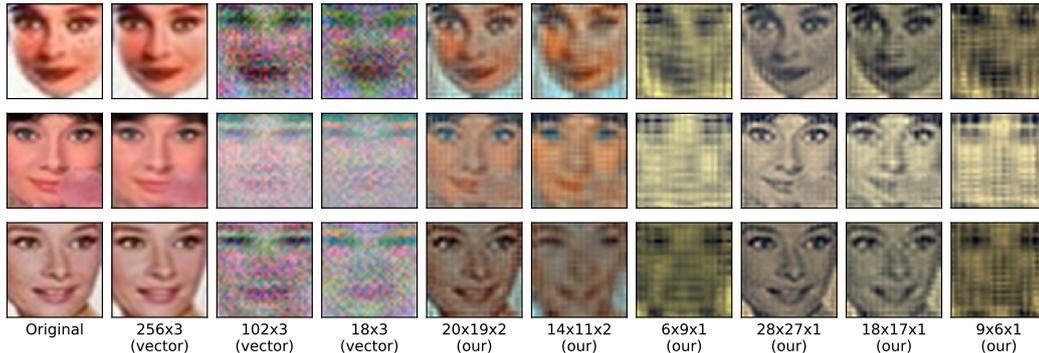


Fig. 2. Illustration of the feature images (inputs to ResNet) synthesized by the proposed framework and the vector-based counterpart. The original images come from the test set of CelebA-32.

TABLE V  
COMPLEXITY OF THE PROPOSED FRAMEWORK AND THE VECTOR-BASED COUNTERPART, EXCLUDING THE BASE CLASSIFIER COMPONENT

Configuration	#Parameters	#FLOPs
$256 \times 3$ [54]	786K	1573K
$20 \times 19 \times 2$ (our)	<b>2.5K</b>	<b>5K</b>
$28 \times 27 \times 1$ (our)	3.5K	7K
$102 \times 3$ [54]	313K	627K
$14 \times 11 \times 2$ (our)	<b>1.6K</b>	<b>3.2K</b>
$18 \times 17 \times 1$ (our)	2.2K	4.5K
$18 \times 3$ [54]	55K	111K
$9 \times 6 \times 1$ (our)	<b>1.0K</b>	<b>1.9K</b>
$6 \times 9 \times 1$ (our)	<b>1.0K</b>	<b>1.9K</b>

### C. Ablation Study

In this subsection, we provide the empirical analysis on the effect of different components in MCL framework. These factors include the effect of the popular nonlinear thresholding step discussed in Section III.B; the choice of having shared or separate weights in CS and FS component; the initialization step discussed in Section III.C; the scalability of the proposed framework when the original dimensionalities of the signal increase. Since the total number of experiment settings when combining all of the aforementioned factors is huge, and the results involved multiple factors are difficult to interpret, we analyze these factors in a progressive manner.

1) *Linearity versus Nonlinearity and Shared versus Separate Weights*: Firstly, the choice of linearity or nonlinearity and the choice of shared or separate weights in CS and FS component are analyzed together since the two factors are closely related. In this setting, the CS and FS components are initialized by HOSVD decomposition as described in Section III.C. The neural network classifier  $\mathfrak{N}$  has the AllCNN architecture with the weights initialized from the corresponding pre-trained network on the original data. Table VI shows the test accuracies on CIFAR-10, CIFAR-100 and CelebA-

32 at different measurements. It is clear that most of the highest test accuracies are obtained without the thresholding step and with separate weights in CS and FS component, i.e., most bold-face numbers appear in the lower quarter on the left side of Table VI. Comparing between linearity and nonlinearity option, it is obvious that the nonlinearity effect of ReLU adversely affect the performances, especially when the number of measurements decreases. The reason might be that applying ReLU to the compressed measurements restricts the information to be represented in the positive subspace only, thus further losing the representation power in the compressed measurements when only a limited number of measurements allowed.

In the linearity setting, while the performance differences between shared and separate weights in some configurations are small, here we should note that allowing non-shared weights can be beneficial in cases where we know that certain features should be synthesized in the FS component in order to make inferences.

2) *Effect of The Initialization Step*: From the observation obtained from the above analysis on the effect of linearity and separate weights, we investigated the effect of the initialization step discussed in Section III.C. All setups were trained with a multilinear FS component having separate weights from CS component. From Table VII, we can easily observe that by initializing the CS and FS components with HOSVD, the performances of the learning systems increase significantly. When CS and FS components are initialized with HOSVD, utilizing a pre-trained network further improves the inference performance of the systems, especially in the low measurement rate regime. Thus, the initialization strategy proposed in Section III.C is beneficial in a general setting for the learning tasks.

3) *Scalability*: Finally, the scalability of the proposed framework is validated in different resolutions of the CelebA dataset. All of the previous experiments were demonstrated with CelebA-32 dataset, which we assume that there are only 3072 elements in the original signal. To investigate

TABLE VI

TEST ACCURACY WITH RESPECT TO THE CHOICE OF LINEARITY OR NONLINEARITY IN CONJUNCTION WITH THE CHOICE OF SHARED OR SEPARATE WEIGHTS IN CS AND FS COMPONENT. THE **BOLD** NUMBERS DENOTE THE BEST TEST ACCURACY (AMONG 4 COMBINATIONS OF *LINEARITY* VERSUS *NONLINEARITY* AND *SHARED* VERSUS *SEPARATE*) IN THE SAME DATASET WITH THE SAME CONFIGURATION

	Configuration	LINEARITY			NONLINEARITY		
		CIFAR-10	CIFAR-100	CelebA-32	CIFAR-10	CIFAR-100	CelebA-32
SHARED	$20 \times 19 \times 2$	$89.25 \pm 00.39$	$66.00 \pm 00.26$	$92.24 \pm 00.21$	$89.16 \pm 00.14$	<b><math>66.11 \pm 00.10</math></b>	$91.69 \pm 00.46$
	$28 \times 27 \times 1$	$88.44 \pm 00.04$	$62.62 \pm 00.30$	$92.42 \pm 00.64$	$88.32 \pm 00.20$	$61.99 \pm 01.24$	$92.61 \pm 00.61$
	$14 \times 11 \times 2$	<b><math>84.37 \pm 00.35</math></b>	$59.75 \pm 00.23$	<b><math>88.00 \pm 00.38</math></b>	$83.84 \pm 00.21$	$58.01 \pm 00.80$	$84.75 \pm 01.37$
	$18 \times 17 \times 1$	<b><math>83.73 \pm 00.10</math></b>	$54.88 \pm 00.23$	$91.22 \pm 00.43$	$83.57 \pm 00.30$	$54.47 \pm 00.39$	$90.85 \pm 00.60$
	$6 \times 9 \times 1$	$64.27 \pm 00.28$	$33.17 \pm 00.20$	$64.47 \pm 00.61$	$62.18 \pm 01.37$	$33.04 \pm 00.08$	$49.46 \pm 00.95$
	$9 \times 6 \times 1$	$64.43 \pm 00.25$	$32.82 \pm 00.39$	$68.25 \pm 00.09$	$62.34 \pm 00.16$	$33.08 \pm 00.16$	$59.29 \pm 01.83$
SEPARATE	$20 \times 19 \times 2$	<b><math>89.35 \pm 00.26</math></b>	$65.97 \pm 00.19$	$92.36 \pm 00.07$	$88.19 \pm 01.00$	$64.28 \pm 02.29$	$91.79 \pm 00.21$
	$28 \times 27 \times 1$	<b><math>88.56 \pm 00.14</math></b>	<b><math>62.82 \pm 00.09</math></b>	<b><math>92.74 \pm 00.31</math></b>	$88.14 \pm 00.34$	$62.15 \pm 00.04$	$92.49 \pm 00.43$
	$14 \times 11 \times 2$	$84.15 \pm 00.55$	<b><math>59.77 \pm 00.12</math></b>	$87.01 \pm 00.99$	$82.16 \pm 02.61$	$59.15 \pm 00.16$	$84.64 \pm 00.57$
	$18 \times 17 \times 1$	$83.17 \pm 00.32$	<b><math>54.96 \pm 00.17</math></b>	<b><math>91.26 \pm 00.14</math></b>	$82.90 \pm 00.19$	$53.90 \pm 00.41$	$90.10 \pm 00.69$
	$6 \times 9 \times 1$	<b><math>64.28 \pm 00.35</math></b>	<b><math>35.16 \pm 00.16</math></b>	<b><math>65.92 \pm 00.77</math></b>	$61.09 \pm 00.20$	$32.15 \pm 00.49$	$50.68 \pm 01.00$
	$9 \times 6 \times 1$	<b><math>64.45 \pm 00.39</math></b>	$34.74 \pm 00.29$	$68.49 \pm 00.28$	$62.18 \pm 00.25$	$32.68 \pm 00.81$	$61.54 \pm 01.40$

TABLE VII

TEST ACCURACY WITH RESPECT TO THE INITIALIZATION OF CS & FS COMPONENT AND THE BASE CLASSIFIER (ALLCNN). THE **BOLD** NUMBERS DENOTE THE BEST TEST ACCURACY (AMONG 4 COMBINATIONS OF *PRECOMPUTE CLASSIFIER* VERSUS *RANDOM CLASSIFIER* AND *PRECOMPUTE CS & FS* VERSUS *RANDOM CS & FS*) IN THE SAME DATASET WITH THE SAME CONFIGURATION

	Configuration	PRECOMPUTE CS & FS			RANDOM CS & FS		
		CIFAR-10	CIFAR-100	CelebA-32	CIFAR-10	CIFAR-100	CelebA-32
PRECOMPUTE CLASSIFIER	$28 \times 27 \times 1$	$88.56 \pm 00.14$	<b><math>62.82 \pm 00.09</math></b>	<b><math>92.74 \pm 00.31</math></b>	$71.47 \pm 00.62$	$38.59 \pm 02.60$	$68.65 \pm 01.56$
	$18 \times 17 \times 1$	$83.17 \pm 00.32$	<b><math>54.96 \pm 00.17</math></b>	<b><math>91.26 \pm 00.14</math></b>	$69.46 \pm 01.16$	$38.65 \pm 00.28$	$65.65 \pm 00.70$
	$9 \times 6 \times 1$	<b><math>64.45 \pm 00.39</math></b>	<b><math>34.74 \pm 00.29</math></b>	<b><math>68.49 \pm 00.28</math></b>	$59.88 \pm 00.17$	$29.03 \pm 01.69$	$60.43 \pm 02.29$
RANDOM CLASSIFIER	$28 \times 27 \times 1$	<b><math>88.71 \pm 00.05</math></b>	$60.36 \pm 00.62$	$85.56 \pm 00.84$	$71.37 \pm 01.24$	$38.98 \pm 01.40$	$67.18 \pm 00.32$
	$18 \times 17 \times 1$	<b><math>83.82 \pm 00.28</math></b>	$49.41 \pm 04.07$	$84.71 \pm 01.39$	$68.93 \pm 01.16$	$37.84 \pm 03.06$	$64.12 \pm 02.06$
	$9 \times 6 \times 1$	$63.99 \pm 00.11$	$33.66 \pm 00.66$	$65.62 \pm 02.71$	$59.85 \pm 00.67$	$30.90 \pm 00.17$	$56.61 \pm 02.27$

the scalability, we pose the following question: *What if the original dimensions of the signal are higher than  $32 \times 32 \times 3$ , with the same numbers of measurements presented in Table II, can we still learn to recognize facial images with feasible costs?* To answer this question, we trained our framework on CelebA-32, CelebA-48, CelebA-64 and CelebA-80 and recorded the test accuracies, the number of parameters and the number of FLOPs at different number of measurements, which are shown in Table VIII. It is clear that at each measurement configuration, when the original signal resolution increases, the measurement rate drops at a similar rate, however, without any adverse effect on the inference performance. Particularly, if we look into the last column of Table VIII, with a sampling rate of only 4%, the proposed framework achieves 93% accuracy, which is only 2% lower compared to that of the base classifier trained on the original data. Here we should note that most of the images in CelebA dataset have higher resolution than  $80 \times 80 \times 3$  pixel, therefore, 4 different versions of CelebA ( $32 \times 32 \times 3$ ,  $48 \times 48 \times 3$ ,  $64 \times 64 \times 3$ ,  $80 \times 80 \times 3$ ) in our experiments indeed contain increasing levels of data fidelity. From the performance statistics, we can observe that the performance of our framework is characterized by the

number of measurements, rather than the measurement rates or compression rates.

Due to the memory limitation when training the vector-based framework at higher resolutions, we could not perform the same set of experiments for the vector-based framework. However, to compare the scalability in terms of computation and memory between the two frameworks, we measured the number of FLOPs and parameters in the vector-based framework, excluding the base classifier and visualize the results on Figure 3. It is worth noting that on the y-axis is the log scale and as the dimensions of the original signal increase, the complexity of the vector-based framework increases by an order of magnitude while our proposed MCL framework scales favorably in both memory and computation.

## V. CONCLUSIONS

In this paper, we proposed Multilinear Compressive Learning, an efficient framework to tackle the Compressive Learning task that operates on multi-dimensional signals. The proposed framework takes into account the tensorial nature of the multi-dimensional signals and performs the compressive sensing as well as the feature extraction step along different modes of

TABLE VIII  
 TEST PERFORMANCE & COMPLEXITY OF THE PROPOSED FRAMEWORK AT DIFFERENT RESOLUTIONS OF THE ORIGINAL CELEBA DATASET, WITH ALLCNN AS THE BASE CLASSIFIER

Configuration	ACCURACY				MEASUREMENT RATE			
	CelebA-32	CelebA-48	CelebA-64	CelebA-80	CelebA-32	CelebA-48	CelebA-64	CelebA-80
Oracle	92.58	93.37	94.75	95.04	1.0	1.0	1.0	1.0
$20 \times 19 \times 2$	$92.36 \pm 00.07$	$91.24 \pm 00.23$	$92.62 \pm 00.36$	$93.01 \pm 00.32$	0.247	0.110	0.062	0.040
$28 \times 27 \times 1$	$92.74 \pm 00.31$	$92.43 \pm 00.19$	$93.39 \pm 00.52$	$93.42 \pm 00.37$	0.246	0.109	0.062	0.039
$14 \times 11 \times 2$	$87.01 \pm 00.99$	$87.00 \pm 00.87$	$87.75 \pm 00.27$	$88.67 \pm 00.26$	0.100	0.045	0.025	0.016
$18 \times 17 \times 1$	$91.26 \pm 00.14$	$90.17 \pm 00.30$	$91.36 \pm 00.43$	$91.89 \pm 00.46$	0.1	0.044	0.025	0.016
$6 \times 9 \times 1$	$65.92 \pm 00.77$	$66.56 \pm 00.05$	$66.69 \pm 00.46$	$66.03 \pm 00.37$	0.018	0.008	0.004	0.003
$9 \times 6 \times 1$	$68.49 \pm 00.28$	$68.69 \pm 00.96$	$67.75 \pm 01.62$	$67.31 \pm 00.49$	0.018	0.008	0.004	0.003

Configuration	#FLOP				#PARAMETER			
	CelebA-32	CelebA-48	CelebA-64	CelebA-80	CelebA-32	CelebA-48	CelebA-64	CelebA-80
$20 \times 19 \times 2$	50K	7.5K	10.0K	12.5K	2.5K	3.8K	5.0K	6.3K
$28 \times 27 \times 1$	7.0K	10.6K	14.1K	17.6K	3.5K	5.3K	7.0K	8.8K
$14 \times 11 \times 2$	3.2K	4.8K	6.4K	8.0K	1.6K	2.4K	3.2K	4.0K
$18 \times 17 \times 1$	4.5K	6.7K	9.0K	11.2K	2.2K	3.4K	4.9K	5.6K
$6 \times 9 \times 1$	1.9K	2.9K	3.9K	4.8K	1.0K	1.4K	1.9K	2.4K
$9 \times 6 \times 1$	1.9K	2.9K	3.9K	4.8K	1.0K	1.4K	1.9K	2.4K

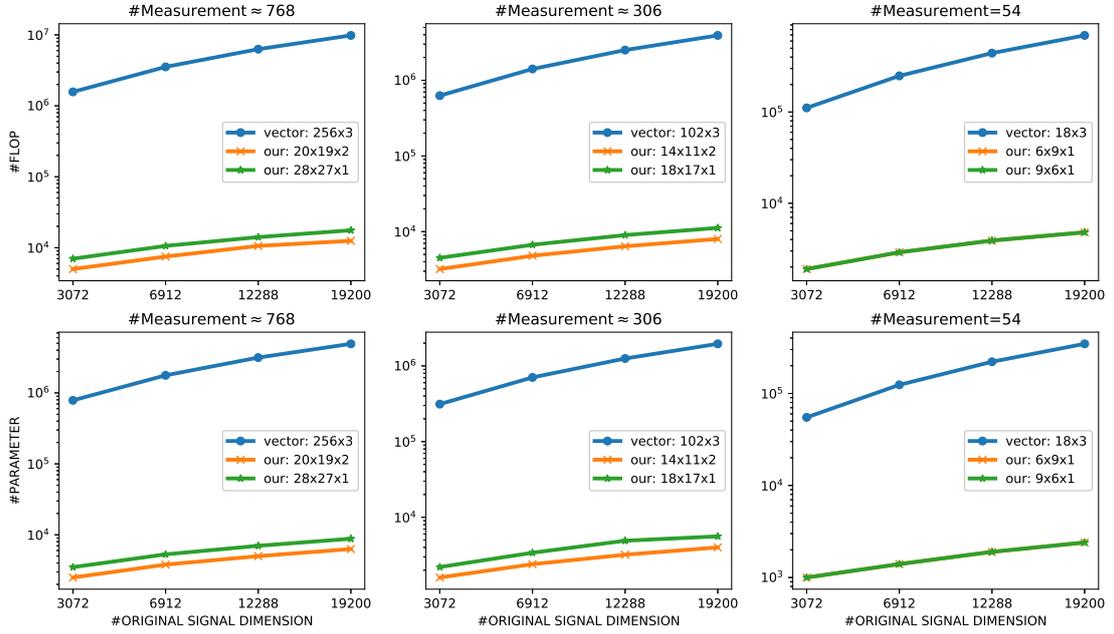


Fig. 3. #FLOP and #PARAMETER versus the original dimensionalities of the signal, measured in the proposed framework and the vector-based framework, excluding the base classifier. The x-axis represents the original dimension of the input signal. The y-axis on the first row represents the number of FLOPs in log scale while the y-axis on the second row represents the number of parameters

the original data, thus being able to retain and synthesize essential information on a multilinear subspace for the learning task. We show theoretically and empirically that the proposed framework outperforms its vector-based counterpart in both inference performance and computational efficiency. Extensive ablation study has been conducted to investigate the effect of different components in the proposed framework, giving insights into the importance of different design choices.

## VI. ACKNOWLEDGEMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR). This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

## REFERENCES

- [1] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling [a sensing/sampling paradigm that goes against the common knowledge in data acquisition]," *IEEE signal processing magazine*, vol. 25, no. 2, pp. 21–30, 2008.
- [2] E. J. Candès, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 59, no. 8, pp. 1207–1223, 2006.
- [3] D. L. Donoho *et al.*, "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.
- [4] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 19–38, IEEE, 2017.
- [5] E. Hesamifard, H. Takabi, and M. Ghasemi, "Cryptodl: Deep neural networks over encrypted data," *arXiv preprint arXiv:1711.05189*, 2017.
- [6] R. Calderbank and S. Jafarpour, "Finding needles in compressed haystacks," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3441–3444, IEEE, 2012.
- [7] M. A. Davenport, M. F. Duarte, M. B. Wakin, J. N. Laska, D. Takhar, K. F. Kelly, and R. G. Baraniuk, "The smashed filter for compressive classification and target recognition," in *Computational Imaging V*, vol. 6498, p. 64980H, International Society for Optics and Photonics, 2007.
- [8] M. A. Davenport, P. Boufounos, M. B. Wakin, R. G. Baraniuk, *et al.*, "Signal processing with compressive measurements," *J. Sel. Topics Signal Processing*, vol. 4, no. 2, pp. 445–460, 2010.
- [9] H. Reboredo, F. Renna, R. Calderbank, and M. R. Rodrigues, "Compressive classification," in *2013 IEEE International Symposium on Information Theory*, pp. 674–678, IEEE, 2013.
- [10] P. K. Baheti and M. A. Neifeld, "Adaptive feature-specific imaging: a face recognition example," *Applied optics*, vol. 47, no. 10, pp. B21–B31, 2008.
- [11] H. Reboredo, F. Renna, R. Calderbank, and M. R. Rodrigues, "Projections designs for compressive classification," in *2013 IEEE Global Conference on Signal and Information Processing*, pp. 1029–1032, IEEE, 2013.
- [12] S. Lohit, K. Kulkarni, P. Turaga, J. Wang, and A. C. Sankaranarayanan, "Reconstruction-free inference on compressive measurements," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 16–24, 2015.
- [13] A. Adler, M. Elad, and M. Zibulevsky, "Compressed learning: A deep neural network approach," *arXiv preprint arXiv:1610.09615*, 2016.
- [14] S. Lohit, K. Kulkarni, and P. Turaga, "Direct inference on compressive measurements using convolutional neural networks," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 1913–1917, IEEE, 2016.
- [15] D. Nion and N. D. Sidiropoulos, "Tensor algebra and multidimensional harmonic retrieval in signal processing for mimo radar," *IEEE Transactions on Signal Processing*, vol. 58, no. 11, pp. 5693–5705, 2010.
- [16] F. Miwakeichi, E. Martinez-Montes, P. A. Valdés-Sosa, N. Nishiyama, H. Mizuhara, and Y. Yamaguchi, "Decomposing eeg data into space–time–frequency components using parallel factor analysis," *NeuroImage*, vol. 22, no. 3, pp. 1035–1045, 2004.
- [17] D. M. Dunlavy, T. G. Kolda, and W. P. Kegelmeyer, "Multilinear algebra for analyzing data with multiple linkages," in *Graph algorithms in the language of linear algebra*, pp. 85–114, SIAM, 2011.
- [18] D. T. Tran, M. Magris, J. Kannianen, M. Gabbouj, and A. Iosifidis, "Tensor representation in high-frequency financial data for price change prediction," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–7, IEEE, 2017.
- [19] D. T. Tran, A. Iosifidis, and M. Gabbouj, "Improving efficiency in convolutional neural networks with multilinear filters," *Neural Networks*, vol. 105, pp. 328–339, 2018.
- [20] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Processing Magazine*, vol. 32, no. 2, pp. 145–163, 2015.
- [21] F. Malgouyres and J. Landsberg, "Multilinear compressive sensing and an application to convolutional linear networks," 2018.
- [22] D. T. Tran, A. Iosifidis, J. Kannianen, and M. Gabbouj, "Temporal attention-augmented bilinear network for financial time-series data analysis," *IEEE transactions on neural networks and learning systems*, 2018.
- [23] T. L. Youd, C. M. Hansen, and S. F. Bartlett, "Revised multilinear regression equations for prediction of lateral spread displacement," *Journal of Geotechnical and Geoenvironmental Engineering*, vol. 128, no. 12, pp. 1007–1017, 2002.
- [24] Q. Zhao, C. F. Caiafa, D. P. Mandic, Z. C. Chao, Y. Nagasaka, N. Fujii, L. Zhang, and A. Cichocki, "Higher order partial least squares (hopls): a generalized multilinear regression method," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 7, pp. 1660–1673, 2013.
- [25] Q. Li and D. Schonfeld, "Multilinear discriminant analysis for higher-order tensor data classification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 12, pp. 2524–2537, 2014.
- [26] D. T. Tran, M. Gabbouj, and A. Iosifidis, "Multilinear class-specific discriminant analysis," *Pattern Recognition Letters*, vol. 100, pp. 131–136, 2017.
- [27] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in neural information processing systems*, pp. 1269–1277, 2014.
- [28] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," *arXiv preprint arXiv:1405.3866*, 2014.
- [29] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," *arXiv preprint arXiv:1412.6553*, 2014.
- [30] Y. Yang, D. Krompass, and V. Tresp, "Tensor-train recurrent neural networks for video classification," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3891–3900, JMLR.org, 2017.
- [31] C. F. Caiafa and A. Cichocki, "Multidimensional compressed sensing and their applications," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 3, no. 6, pp. 355–380, 2013.
- [32] Y. Yu, J. Jin, F. Liu, and S. Crozier, "Multidimensional compressed sensing mri using tensor decomposition-based sparsifying transform," *PLoS one*, vol. 9, no. 6, p. e98441, 2014.
- [33] R. Robucci, L. K. Chiu, J. Gray, J. Romberg, P. Hasler, and D. Anderson, "Compressive sensing on a cmos separable transform image sensor," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5125–5128, IEEE, 2008.
- [34] M. Aharon, M. Elad, A. Bruckstein, *et al.*, "K-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on signal processing*, vol. 54, no. 11, p. 4311, 2006.
- [35] D. L. Donoho and M. Elad, "Optimally sparse representation in general (nonorthogonal) dictionaries via  $l_1$  minimization," *Proceedings of the National Academy of Sciences*, vol. 100, no. 5, pp. 2197–2202, 2003.
- [36] J. A. Tropp and S. J. Wright, "Computational methods for sparse solution of linear inverse problems," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 948–958, 2010.
- [37] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM review*, vol. 43, no. 1, pp. 129–159, 2001.

- [38] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Transactions on Information theory*, vol. 50, no. 10, pp. 2231–2242, 2004.
- [39] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on information theory*, vol. 53, no. 12, pp. 4655–4666, 2007.
- [40] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000.
- [41] M. F. Duarte and R. G. Baraniuk, "Kronecker compressive sensing," *IEEE Transactions on Image Processing*, vol. 21, no. 2, pp. 494–504, 2012.
- [42] C. F. Caiafa and A. Cichocki, "Computing sparse representations of multidimensional signals using kronecker bases," *Neural computation*, vol. 25, no. 1, pp. 186–220, 2013.
- [43] R. G. Baraniuk and M. B. Wakin, "Random projections of smooth manifolds," *Foundations of computational mathematics*, vol. 9, no. 1, pp. 51–77, 2009.
- [44] K. Kulkarni and P. Turaga, "Recurrence textures for human activity recognition from compressive cameras," in *2012 19th IEEE International Conference on Image Processing*, pp. 1417–1420, IEEE, 2012.
- [45] K. Kulkarni and P. Turaga, "Reconstruction-free action inference from compressive imagers," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 4, pp. 772–784, 2016.
- [46] B. Hollis, S. Patterson, and J. Trinkle, "Compressed learning for tactile object recognition," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1616–1623, 2018.
- [47] A. Değerli, S. Aslan, M. Yamac, B. Sankur, and M. Gabbouj, "Compressively sensed image recognition," in *2018 7th European Workshop on Visual Information Processing (EUVIP)*, pp. 1–6, IEEE, 2018.
- [48] Y. Xu and K. F. Kelly, "Compressed domain image classification using a multi-rate neural network," *arXiv preprint arXiv:1901.09983*, 2019.
- [49] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.
- [50] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*, pp. 630–645, Springer, 2016.
- [51] F. Wu, X. Tan, Y. Yang, D. Tao, S. Tang, and Y. Zhuang, "Supervised nonnegative tensor factorization with maximum-margin constraint," in *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.
- [52] Y.-D. Kim and S. Choi, "Nonnegative tucker decomposition," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, IEEE, 2007.
- [53] L. Grasedyck, D. Kressner, and C. Tobler, "A literature survey of low-rank tensor approximation techniques," *GAMM-Mitteilungen*, vol. 36, no. 1, pp. 53–78, 2013.
- [54] E. Zisselman, A. Adler, and M. Elad, "Compressed learning for image classification: A deep neural network approach," *Processing, Analyzing and Learning of Images, Shapes, and Forms*, vol. 19, p. 1, 2018.
- [55] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," tech. rep., Citeseer, 2009.
- [56] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.
- [57] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.
- [58] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.
- [59] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

# Performance Indicator in Multilinear Compressive Learning

Dat Thanh Tran\*, Moncef Gabbouj\* & Alexandros Iosifidis†

\*Department of Computing Sciences, Tampere University, Finland

†Department of Engineering, Aarhus University, Denmark

Email: {thanh.tran, moncef.gabbouj}@tuni.fi, alexandros.iosifidis@eng.au.dk

**Abstract**—Recently, the Multilinear Compressive Learning (MCL) framework was proposed to efficiently optimize the sensing and learning steps when working with multidimensional signals, i.e. tensors. In Compressive Learning in general, and in MCL in particular, the number of compressed measurements captured by a compressive sensing device characterizes the storage requirement or the bandwidth requirement for transmission. This number, however, does not completely characterize the learning performance of a MCL system. In this paper, we analyze the relationship between the input signal resolution, the number of compressed measurements and the learning performance of MCL. Our empirical analysis shows that the reconstruction error obtained at the initialization step of MCL strongly correlates with the learning performance, thus can act as a good indicator to efficiently characterize learning performances obtained from different sensor configurations without optimizing the entire system.

## I. INTRODUCTION

Compressive Sensing (CS) [1] is an efficient signal acquisition method that acquires the measurement of the signal by sampling and linearly interpolating the samples at the hardware level, i.e. by using CS devices. Particularly, let  $\mathbf{y} \in \mathbb{R}^I$  be the discrete measurements of the input signal. Using a CS device, we obtain the compressed measurements  $\mathbf{z}$  of the signal, instead of  $\mathbf{y}$ , with the compression step as follows:

$$\mathbf{z} = \Phi \mathbf{y} \quad (1)$$

where  $\mathbf{z} \in \mathbb{R}^M$  often has significantly lower dimension than  $\mathbf{y}$ , i.e.,  $M \ll I$ .  $\Phi \in \mathbb{R}^{M \times I}$  is called the sensing operator.

This is different from the traditional approach where we obtain the discrete samples  $\mathbf{y}$  from the signal acquisition device, and compression step is often conducted at the software level, *being separate from the acquisition step*. Since signal compression is performed before signal registration during the sampling phase, CS devices require significantly lower temporary storage and bandwidth requirement. This paradigm is therefore prevalent in many applications that involve high-dimensional signals or critical computational requirements.

Although, in general, ideal sampling requires the signal to be sampled at higher rates than the Nyquist rate to ensure perfect reconstruction, in CS, the undersampled signal (due to  $M \ll I$ ) can still be reconstructed almost perfectly if the sparsity assumption holds and the sensing operators possess

certain properties [2], [3]. While the possibility to recover  $\mathbf{y}$  from compressed measurements  $\mathbf{z}$  is critical in some applications, like Magnetic Resonance Imaging (MRI) for expert diagnosis, there are other applications where the main goal is to detect certain patterns or to infer special properties from the acquired signal, rather than signal recovery. Thus, arises the idea of learning from compressed measurement.

Compressive Learning (CL) [4], [5], [6], [7] combines Compressive Sensing and Machine Learning into a single optimization problem which focuses on maximizing the learning performance, rather than performance on signal reconstruction. In the early works, the design of the sensing operator  $\Phi$  was decoupled from the construction of the learning model. Following the developments and wide adoption of stochastic optimization, recent works [8], [9], [10], [11], [12], [13] have adopted an end-to-end learning paradigm that jointly optimizes the sensing operator and the inference model.

In order to work efficiently with multidimensional signals, Multilinear Compressive Learning (MCL) was recently proposed in [12]. MCL formulates sensing and feature synthesis based on multilinear algebra. Multilinear sensing and feature synthesis operators not only preserve the natural tensor format of the multidimensional signal but also require fewer computations and memory compared to other CL models which operate on vectorized signals. This makes MCL highly suitable for applications requiring the analysis of high-dimensional signals like images/videos on constrained computation/bandwidth platforms, such as drones and robots.

When building an MCL model to tackle a particular learning task, the configuration of the CS device plays an important role in the design process. Particularly, the choice of input resolution ( $I$ ), i.e., the number of discrete samples initially captured by the device, and the size of compressed measurements ( $M$ ) directly affects the learning performance.  $I$  and  $M$  characterize the computational complexity of CS device while  $M$  alone characterizes the requirement for storage or transmission bandwidth. Given that a small increment or decrement of  $I$  and/or  $M$  only leads to a small changes in computational complexity which might be within the design requirements, extensive experimentation is needed to determine the a good combination of the dimensions  $I$  and  $M$  for the problem at hand.

In this work, by analyzing the performance under different combinations of  $I$  and  $M$ , we seek to find a performance indicator of MCL models that can help us rapidly gauge different configurations of the CS component without the need of conducting the entire optimization process. Our empirical analysis reveals that the reconstruction error obtained at the initialization step of MCL models strongly correlates with the final learning performance, thus can act as a good performance indicator.

## II. RELATED WORK

We are not aware of any work that aims to characterize the learning performance of a Compressive Learning system in terms of the sensing configurations or that investigates possible surrogate measures for its performance. Existing works only evaluate few configurations of the compressed measurement or the resolution of the input signal, while their experiments are not designed to isolate the effect of CS device configuration for studying its importance to the final learning performance. Remotely related to our work is the class of Neural Architecture Search (NAS) methods [14], [15], [16], [17] that estimates the performance of a candidate architecture by a surrogate model [18], [19] or by learning curve extrapolation [20], [21]. Instead of learning to predict the performance, we extensively evaluate several configurations of CS device on different learning problems and analyze the results to seek for a consistent performance indicator.

A Multilinear Compressive Learning (MCL) system [12] consists of three modules: the Compressive Sensing (CS) component, the Feature Synthesis (FS) component and the task-specific neural network  $\mathfrak{N}$ .

The Compressive Sensing (CS) component of MCL adopts multidimensional compressive sensing which is implemented via separable sensing operators, each of which operates on a mode of the input signal (tensor). Specifically, let us denote by  $\mathcal{Y} \in \mathbb{R}^{I_1 \times \dots \times I_K}$  and  $\mathcal{Z} \in \mathbb{R}^{M_1 \times \dots \times M_K}$  the discrete samples of the input tensor signal and the compressed measurements obtained from CS component, respectively. Here  $I_1 \times \dots \times I_K$  denotes the resolution of sensors of the CS device. The CS component performs signal acquisition as follows:

$$\mathcal{Z} = \mathcal{Y} \times_1 \Phi_1 \times \dots \times_K \Phi_K \quad (2)$$

where  $\Phi_k \in \mathbb{R}^{M_k \times I_k}$ ,  $k = 1, \dots, K$  denote the separable sensing operators and  $\times_k$  denotes the mode- $k$  product.

Here we should note that the CS device performs discrete sampling (obtaining  $\mathcal{Y}$ ) and compression simultaneously, producing the compressed measurements  $\mathcal{Z}$  as the digital output, while  $\mathcal{Y}$  is not registered digitally.  $\mathcal{Y}$  can be considered as an intermediate state of the signal when acquired by a CS device. The dimension of  $\mathcal{Y}$  represents the resolution at which the sensor inside a CS device performs discrete sampling. In addition, when the system is deployed, the CS component is implemented at the hardware level with configuration parameters  $\Phi_k$ . That is, the CS component is a signal acquisition device and what we obtain from this device is the compressed version of the signal, i.e.  $\mathcal{Z}$ , rather than its high-resolution

version  $\mathcal{Y}$ . Since MCL is an end-to-end Compressive Learning method [12], the values of  $\Phi_k$  which are used to build or configure the CS device are determined via optimization. Thus, the CS component is simulated at the software level using Eq. (2) during the optimization stage.

Given the compressed measurements  $\mathcal{Z}$ , relevant features for the learning task that preserve the tensor structure of the input signal are synthesized by the FS component. Any arbitrary design that preserves the tensor structure can be used for the FS component. For example, in [13], the authors use a highly-nonlinear design which consists of multiple convolution layers for the FS component. To isolate the effect of CS and FS component, here we adopt the original formulation in [12] which mirrors the sensing step in Eq. (2) by a multilinear transformation to synthesize new features, i.e.:

$$\mathcal{T} = \mathcal{Z} \times_1 \Theta_1 \times \dots \times_K \Theta_K \quad (3)$$

where  $\mathcal{T} \in \mathbb{R}^{\tilde{I}_1 \times \dots \times \tilde{I}_K}$  denotes the synthesized features and  $\Theta_k \in \mathbb{R}^{\tilde{I}_k \times M_k}$ ,  $k = 1, \dots, K$  denote the parameters of the FS component.

Finally, the task-specific neural network  $\mathfrak{N}$  takes the synthesized feature  $\mathcal{T}$  as input and outputs the predicted label.

Similar to other CL methods [8], [9], [11], during system optimization, MCL utilizes high-resolution signal  $\mathcal{Y}$  (often obtained from standard sensors with higher computational cost than the CS sensor) and the corresponding class label to optimize the system's parameters. That is, the parameters of the three components of the MCL model are jointly optimized to maximize the learning performance using stochastic gradient descend. An important processing step in this process is the MCL model's initialization. In [12], the authors propose an initialization scheme that preserves the energy of the signal in the compressed measurements  $\mathcal{Z}$ . This is done by decomposing  $\mathcal{Y}$  using the HOSVD [22]:

$$\mathcal{Y} = \mathcal{S} \times_1 \mathbf{U}_1 \times \dots \times_K \mathbf{U}_K \quad (4)$$

where  $\mathcal{S} \in \mathbb{R}^{M_1 \times \dots \times M_K}$  and  $\mathbf{U}_k \in \mathbb{R}^{M_k \times I_k}$ ,  $k = 1, \dots, K$ . Then, the CS components are initialized with  $\Phi_k = \mathbf{U}_k^T$ . Furthermore, the parameters of the FS component is initialized with values that optimally reconstruct (in the least-square sense) the high-resolution signal  $\mathcal{Y}$ . This is done by setting the dimensions of the synthesized features  $\mathcal{T}$  equal those of high-resolution signal  $\mathcal{Y}$ , i.e.,  $\tilde{I}_k = I_k, \forall k = 1, \dots, K$ , and setting  $\Theta_k = \mathbf{U}_k$ .

## III. METHOD

When building a MCL model for deployment, the computational requirements determine the range of feasible dimensions for  $\mathcal{Y}$  and  $\mathcal{Z}$ . That is:

$$\begin{aligned} I_k^{\min} &\leq I_k \leq I_k^{\max} \\ M_k^{\min} &\leq M_k \leq M_k^{\max} \\ \forall k &= 1, \dots, K \end{aligned} \quad (5)$$

where  $I_k^{\min}$  and  $I_k^{\max}$  denote the lower- and upper-bounds of the feasible values for  $I_k$ .

When  $K$ ,  $(I_k^{\max} - I_k^{\min})$  or  $(M_k^{\max} - M_k^{\min})$  are large, the number of possible combinations of  $I_k$  and  $M_k$  can be enormous. Thus, the motivation of our work lies in the attempt to efficiently determine an optimal configuration of CS device (i.e., the choice of  $I_k$  and  $M_k$ ), without the need of conducting the entire optimization process of MCL for every feasible combination of  $I_k$  and  $M_k$ . One might guess that the higher the resolution  $I_k$  and number of compressed measurements  $M_k$  are, the higher the learning performance will be. However, this is not necessarily true as it will be shown in the Experiment Section of this paper.

One approach to tackle our problem is to empirically characterize the learning performance in terms of  $I_k$  and  $M_k$  and seek to find an indicator that reflects the performance ranking. Given a learning problem expressed via the training set, the performance of a MCL model depends on its architectural design. There are three main factors that affect the model's complexity, and thus its learning capacity: the CS device configuration, the FS configuration and the architecture of the task-specific neural network  $\mathfrak{N}$ . The CS device configuration refers to the resolution of the sensor ( $I_1 \times \dots \times I_K$ ) and the dimensions of the compressed measurements ( $M_1 \times \dots \times M_k$ ), while the FS configuration refers to the dimensions of the synthesized features ( $\tilde{I}_1 \times \dots \times \tilde{I}_K$ ).

In order to analyze and characterize the learning performance in terms of the CS device configuration, it is important to ultimately limit variations in the FS component and the architecture of  $\mathfrak{N}$  when evaluating multiple choices of CS configuration across multiple learning problems. To do so, we fix the architecture of  $\mathfrak{N}$  given any configuration of CS component. In addition, we also fix the dimensions of  $\mathcal{T}$  to  $I_1^{\max} \times \dots \times I_K^{\max}$  for any given value of  $I_k$  and  $M_k$ . That is, the parameters of the FS component have the following dimensions:

$$\begin{aligned} \Theta_1 &\in \mathbb{R}^{I_1^{\max} \times M_1}, \forall M_1 \in [M_1^{\min}, M_1^{\max}] \\ &\vdots \\ \Theta_K &\in \mathbb{R}^{I_K^{\max} \times M_K}, \forall M_K \in [M_K^{\min}, M_K^{\max}] \end{aligned} \quad (6)$$

Since we fix the dimensions of  $\mathcal{T}$ , we can no longer initialize parameters of the FS component using HOSVD if the resolution of CS component is different from the highest feasible resolution, i.e.,  $I_1 \times \dots \times I_K \neq I_1^{\max} \times \dots \times I_K^{\max}$ . As it has been shown in [12], initialization is a crucial step when optimizing MCL models. Thus, to circumvent the inability to use HOSVD, we propose to use a different initialization strategy that still pertains to preserving energy in  $\mathcal{Z}$  and  $\mathcal{T}$ .

As mentioned previously in Section II, in order to train any end-to-end CL model, high-resolution signals and the corresponding labels are needed. In our work, we only need to acquire the set of training signals with labels at the highest resolution, i.e.,  $I_1^{\max} \times \dots \times I_K^{\max}$  using a standard signal acquisition device with higher computational and time complexity than a CS device. For using training data at a lower resolution, instead of using a different device to acquire at a lower resolution, we simulate them by applying down-

sampling to the high-resolution signal  $\mathcal{Y} \in \mathbb{R}^{I_1^{\max} \times \dots \times I_K^{\max}}$ .

Let  $\mathbf{S}@ (I_1 \times \dots \times I_K) = \{(\mathcal{Y}_i@ (I_1 \times \dots \times I_K), c_i) | i = 1, \dots, N\}$  denote the training set of  $N$  samples at resolution  $I_1 \times \dots \times I_K$ .  $\mathbf{S}@ (I_1 \times \dots \times I_K)$  represents the training data that is used to optimize an MCL model with the CS device sampling at resolution  $I_1 \times \dots \times I_K$ . In order to initialize the parameters of the CS and FS components, we obtain the initial values of  $\Phi_k$  and  $\Theta_k$  ( $k = 1, \dots, K$ ) by solving the following optimization problem:

$$\arg \min_{\{\Phi_k\}, \{\Theta_k\}} \sum_{i=1}^N \|\text{FS}(\text{CS}(\mathcal{Y}@ (I_1 \times \dots \times I_K))) - \mathcal{Y}@ (I_1^{\max} \times \dots \times I_K^{\max})\|_F^2 \quad (7)$$

where  $\text{FS}(\text{CS}(\mathcal{Y}@ (I_1 \times \dots \times I_K)))$  denotes the features synthesized by the FS component, given the CS device operating at resolution  $I_1 \times \dots \times I_K$ . In addition,  $\|\cdot\|_F$  denotes the Frobenius norm.

The objective in Eq. (7) is used to initialize  $\Phi_k$  and  $\Theta_k$  with values that produce features resembling (in the least-square sense) the input signals at the highest resolution. This initialization strategy of the CS and FS components thus resembles the one in [12], which uses HOSVD.

To initialize the parameters of the task-specific neural network  $\mathfrak{N}$ , we optimize the following objective:

$$\arg \min_{\Omega} \sum_{i=1}^N \mathcal{L}(\mathfrak{N}(\mathcal{Y}@ (I_1^{\max} \times \dots \times I_K^{\max})); c_i) \quad (8)$$

where  $\Omega$  denotes the parameters of  $\mathfrak{N}$ , and  $\mathcal{L}$  denotes the inference loss function while  $\mathfrak{N}(\mathcal{Y}@ (I_1^{\max} \times \dots \times I_K^{\max}))$  denotes the prediction generated by  $\mathfrak{N}$  given the high-resolution input  $\mathcal{Y}$ .

After applying the initialization steps in Eq. (7) and Eq. (8), all parameters of the MCL model are jointly optimized to minimize the inference loss:

$$\arg \min_{\{\Phi_k\}, \{\Theta_k\}, \Omega} \sum_{i=1}^N \mathcal{L}(\mathfrak{N}(\text{FS}(\text{CS}(\mathcal{Y}@ (I_1 \times \dots \times I_K))))), c_i) \quad (9)$$

We optimize the objective functions in Eqs. (7), (8), and (9) using stochastic gradient descent. In the next section, we provide detailed description of our experimental setup as well as our analysis of the effects of CS device configuration based on the empirical results.

## IV. EXPERIMENTS

### A. Datasets and Experiment Protocol

We conducted our empirical analysis using image data. Two image datasets representing two different learning tasks were used in our experiments: face recognition and object recognition. These datasets are:

- PubFig83 [23] is a medium-size dataset that contains 13002 facial images of 83 public figures. The dataset was curated from the list of URLs compiled by [24] by removing near-duplicate samples and individuals with few samples. Since the photos were collected from the

TABLE I

TEST PERFORMANCES OF PUBFIG83 DATASET. THE UPPER SECTION SHOWS TEST ACCURACY WHILE THE LOWER SECTION SHOWS MEAN SQUARED ERROR (MSE) MEASURED ON TEST SET WHEN OPTIMIZING EQ. (7). **BOLD-FACE** NUMBERS INDICATE THE TOP-3 ACCURACY AND THE CORRESPONDING MSE

Test Accuracy (%)		$\mathcal{Y}$ Dimension ( $I_1 \times I_2 \times I_3$ )				
		$256 \times 256 \times 3$	$224 \times 224 \times 3$	$192 \times 192 \times 3$	$160 \times 160 \times 3$	$128 \times 128 \times 3$
$\mathcal{Z}$ Dimension ( $M_1 \times M_2 \times M_3$ )	$30 \times 30 \times 1$	66.01	74.38	67.69	78.36	44.49
	$28 \times 28 \times 1$	<b>80.86</b>	<b>79.46</b>	57.17	72.12	58.17
	$26 \times 26 \times 1$	77.53	<b>79.32</b>	67.47	47.03	47.59
	$24 \times 24 \times 1$	58.54	53.70	54.43	62.00	58.85
	$22 \times 22 \times 1$	57.53	71.16	72.23	77.39	75.67
	$20 \times 20 \times 1$	58.44	68.17	38.35	43.23	71.45
MSE during initialization (Eq. (7))		$\mathcal{Y}$ Dimension ( $I_1 \times I_2 \times I_3$ )				
		$256 \times 256 \times 3$	$224 \times 224 \times 3$	$192 \times 192 \times 3$	$160 \times 160 \times 3$	$128 \times 128 \times 3$
$\mathcal{Z}$ Dimension ( $M_1 \times M_2 \times M_3$ )	$30 \times 30 \times 1$	0.1368	0.1161	0.0408	0.0256	0.2490
	$28 \times 28 \times 1$	<b>0.0185</b>	<b>0.0196</b>	0.0395	0.1357	0.1957
	$26 \times 26 \times 1$	0.0192	<b>0.0173</b>	0.0320	0.2008	0.2135
	$24 \times 24 \times 1$	0.0480	0.1078	0.1675	0.0567	0.0425
	$22 \times 22 \times 1$	0.1927	0.0241	0.0306	0.0167	0.0189
	$20 \times 20 \times 1$	0.0323	0.0254	0.1218	0.0616	0.0199

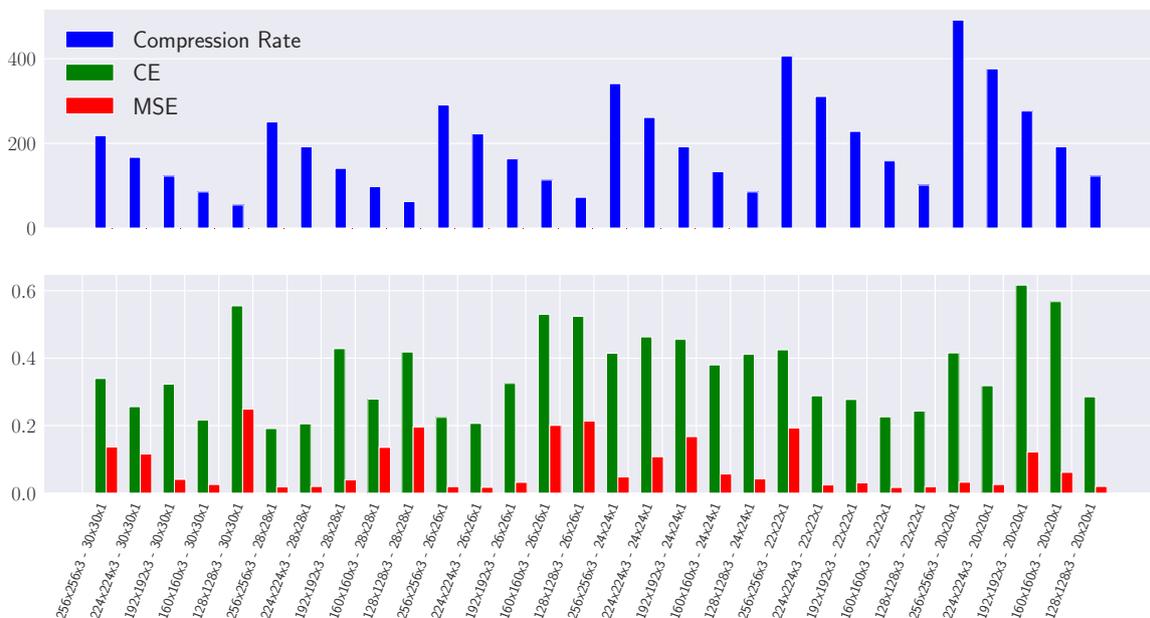


Fig. 1. PubFig83 Performance

internet, the dataset represents the task of recognizing identities in uncontrolled situations using near-frontal faces.

- Caltech101 [25] is an object recognition dataset that contains pictures of objects from 101 categories. Besides 101 categories, the dataset also contains a background class which represents non-object images. The dataset is not well-balanced with the number of images per category ranging from 40 to 800. In total, there are 9145 images in this dataset.

For both datasets, we randomly selected 60%, 20%, 20% of the samples from each class for training, validation and testing, respectively. PubFig83 and Caltech101 both contain RGB images of varying resolutions. In order to simulate different resolutions of the CS device, we resized the images to 5 different resolutions, ranging from  $256 \times 256 \times 3$  to  $128 \times 128 \times 3$ . That is, we experimented with the set of feasible resolutions of  $\mathcal{Y}$ :  $I_1 \times I_2 \times I_3 \in \{256 \times 256 \times 3, 224 \times 224 \times 3, 192 \times 192 \times 3, 160 \times 160 \times 3, 128 \times 128 \times 3\}$ . Regarding compressed measurements  $\mathcal{Z}$ , we considered the following set

TABLE II

TEST PERFORMANCES OF CALTECH101 DATASET. THE UPPER SECTION SHOWS TEST ACCURACY WHILE THE LOWER SECTION SHOWS MEAN SQUARED ERROR (MSE) MEASURED ON TEST SET WHEN OPTIMIZING EQ. (7). **BOLD-FACE** NUMBERS INDICATE THE TOP-3 ACCURACY AND THE CORRESPONDING MSE

Test Accuracy (%)		$\mathcal{Y}$ Dimension ( $I_1 \times I_2 \times I_3$ )				
		$256 \times 256 \times 3$	$224 \times 224 \times 3$	$192 \times 192 \times 3$	$160 \times 160 \times 3$	$128 \times 128 \times 3$
$\mathcal{Z}$ Dimension ( $M_1 \times M_2 \times M_3$ )	$30 \times 30 \times 1$	53.35	<b>71.08</b>	<b>71.16</b>	61.67	53.67
	$28 \times 28 \times 1$	60.68	47.48	65.18	60.09	64.22
	$26 \times 26 \times 1$	54.88	48.15	47.75	57.22	58.02
	$24 \times 24 \times 1$	<b>68.72</b>	54.77	52.17	55.79	63.46
	$22 \times 22 \times 1$	60.78	53.03	56.87	61.48	54.32
	$20 \times 20 \times 1$	47.72	52.90	53.00	51.42	47.34
MSE during initialization (Eq. (7))		$\mathcal{Y}$ Dimension ( $I_1 \times I_2 \times I_3$ )				
		$256 \times 256 \times 3$	$224 \times 224 \times 3$	$192 \times 192 \times 3$	$160 \times 160 \times 3$	$128 \times 128 \times 3$
$\mathcal{Z}$ Dimension ( $M_1 \times M_2 \times M_3$ )	$30 \times 30 \times 1$	0.3617	<b>0.0692</b>	<b>0.0396</b>	0.2611	0.2655
	$28 \times 28 \times 1$	0.1045	0.3819	0.0569	0.2000	0.0638
	$26 \times 26 \times 1$	0.1243	0.3693	0.3571	0.2018	0.2488
	$24 \times 24 \times 1$	<b>0.0837</b>	0.1753	0.3200	0.1513	0.1823
	$22 \times 22 \times 1$	0.1035	0.2127	0.1805	0.1694	0.3151
	$20 \times 20 \times 1$	0.3857	0.2061	0.2322	0.1522	0.3746

of 6 feasible dimensions:  $M_1 \times M_2 \times M_3 \in \{30 \times 30 \times 1, 28 \times 28 \times 1, 26 \times 26 \times 1, 24 \times 24 \times 1, 22 \times 22 \times 1, 20 \times 20 \times 1\}$ . This leads to 30 combinations for the sizes of  $\mathcal{Y}$  and  $\mathcal{Z}$ . For each combination, the experiment was run 5 times and the average performance on the test set is reported.

Regarding the architecture of the task-specific neural network  $\mathfrak{N}$ , we adopted the DenseNet121 architecture proposed in [26], which was pretrained on the ILSVRC2012 database. We first performed the initialization of  $\mathfrak{N}$  by optimizing Eq. (8) for each dataset. To limit the possible variations in the effect of  $\mathfrak{N}$  to different CS configurations, the values of  $\Omega$  obtained by optimizing Eq. (8) is used in all experiments and all combinations of  $I_1 \times I_2 \times I_3$  and  $M_1 \times M_2 \times M_3$ . Different from  $\mathfrak{N}$ , the initialization of CS and FS components using Eq. (7) is repeated for every experiment.

Stochastic optimization was done using ADAM optimizer [27]. Eq. (7) was optimized for a total of 35 epochs with the learning rate schedule  $\{10^{-3}, 10^{-4}, 10^{-5}\}$ , changing at epoch 6 and 26. In addition, weight decay regularization of  $5 \times 10^{-5}$  was used when optimizing Eq. (7). For optimizing Eqs. (8) and (9), we updated the parameters for 120 epochs starting with learning rate of  $10^{-3}$ , then dropping to  $10^{-4}$ , and to  $10^{-5}$ , at epoch 21 and 101, respectively. The weight decay coefficient was set to  $10^{-4}$ .

## B. Experiment Results

The test accuracies obtained by using different configurations on PubFig83 and Caltech101 datasets are illustrated in the upper section of Tables I and II, respectively. Moreover, in the lower section of Tables I and II, we also show the Mean Squared Error (MSE) measured on the test set obtained when optimizing Eq. (7), i.e., *during initializing the CS and FS components*.

The first observation from our experimental results is that higher resolutions of the CS device and higher numbers of

measurements do not always yield better learning performance. In fact, for both datasets, at the maximum resolution ( $256 \times 256 \times 3$ ) and the maximum number of compressed measurements ( $30 \times 30 \times 1$ ), we obtain test accuracies that are far below the best achieved and highlighted with bold-face numbers.

On a closer look, no clear monotonic relationship between the learning performance and the CS resolution or the number of measurements can be observed from both datasets. For example, when we fix the number of measurements and increase or decrease the CS resolution, we do not observe the corresponding increase or decrease in test accuracy. Similarly, when we fix the CS resolution, the learning performances do not change linearly with the number of measurements.

On the other hand, the MSE obtained during the initialization of the CS and FS components reflects well the final learning performances. For example, by inspecting the top-3 configurations for both datasets, we can see that the corresponding MSE values are among the lowest. Similarly, those configurations with high MSE values achieve very poor accuracies.

To better illustrate the trend, we plot the classification error (CE) versus MSE as well as the compression rate ( $(I_1 * I_2 * I_3) / (M_1 * M_2 * M_3)$ ) for PubFig83 and Caltech101 in Figures 1 and 2, respectively. By observing both figures, it can be seen that the compression rate shows no clear linear relationship with the learning performance. Quantitatively, the Pearson correlation values between the final classification error (CE) and the MSE during initialization are equal to **0.65** and **0.82** for PubFig83 and Caltech101, indicating a strong correlation between the final performance and the performance obtained when initializing CS and FS components. On the other hand, the Pearson correlation values between CE and the compression rate are equal to **-0.02** and **0.23** for PubFig83 and Caltech101, respectively.

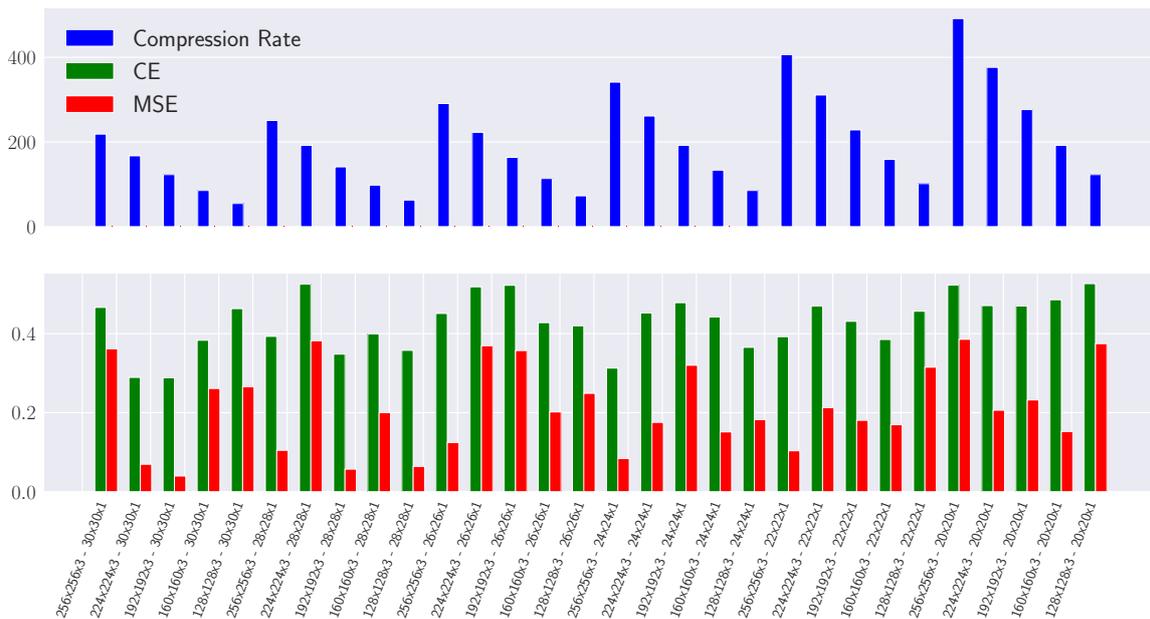


Fig. 2. Caltech101 Performance

## V. CONCLUSION

Multilinear Compressive Learning (MCL) is an efficient framework to tackle the problem of learning with compressed measurements from high-dimensional multidimensional signals. In this paper, we empirically investigated the learning performance of Multilinear Compressive Learning models with respect to the configurations of the Compressive Sensing device in MCL. Our analysis showed that higher sensor resolutions and higher number of measurements do not always lead to better learning performance. In addition, the compression rate also showed no clear linear relationship with the final learning performance. On the other hand, the Mean Squared Error (MSE) obtained during initializing the CS and FS components of MCL strongly correlates with the final learning performance. This suggests that this metric can be used as a surrogate measure of the final learning performance to gauge between different configurations of the CS device without conducting the entire optimization procedure, which is often time-consuming.

## VI. ACKNOWLEDGEMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR). This publication reflects the authors views only. The European Commission is not responsible for any use that may be made of the information it contains.

## REFERENCES

- [1] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling [a sensing/sampling paradigm that goes against the common knowledge in data acquisition]," *IEEE signal processing magazine*, vol. 25, no. 2, pp. 21–30, 2008.
- [2] E. J. Candès, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 59, no. 8, pp. 1207–1223, 2006.
- [3] D. L. Donoho, "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.
- [4] R. Calderbank and S. Jafarpour, "Finding needles in compressed haystacks," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3441–3444, IEEE, 2012.
- [5] M. A. Davenport, M. F. Duarte, M. B. Wakin, J. N. Laska, D. Takhar, K. F. Kelly, and R. G. Baraniuk, "The smashed filter for compressive classification and target recognition," in *Computational Imaging V*, vol. 6498, p. 64980H, International Society for Optics and Photonics, 2007.
- [6] M. A. Davenport, P. Boufounos, M. B. Wakin, R. G. Baraniuk, *et al.*, "Signal processing with compressive measurements," *J. Sel. Topics Signal Processing*, vol. 4, no. 2, pp. 445–460, 2010.
- [7] H. Reberdo, F. Renna, R. Calderbank, and M. R. Rodrigues, "Compressive classification," in *2013 IEEE International Symposium on Information Theory*, pp. 674–678, IEEE, 2013.
- [8] A. Adler, M. Elad, and M. Zibulevsky, "Compressed learning: A deep neural network approach," *arXiv preprint arXiv:1610.09615*, 2016.
- [9] S. Lohit, K. Kulkarni, and P. Turaga, "Direct inference on compressive measurements using convolutional neural networks," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 1913–1917, IEEE, 2016.
- [10] B. Hollis, S. Patterson, and J. Trinkle, "Compressed learning for tactile object recognition," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1616–1623, 2018.
- [11] E. Zisselman, A. Adler, and M. Elad, "Compressed learning for image classification: A deep neural network approach," *Processing, Analyzing and Learning of Images, Shapes, and Forms*, vol. 19, p. 1, 2018.
- [12] D. T. Tran, M. Yamac, A. Degerli, M. Gabbouj, and A. Iosifidis, "Multilinear compressive learning," *IEEE Transactions on Neural Networks and Learning Systems (2020) in press*, 2020.
- [13] D. T. Tran, M. Gabbouj, and A. Iosifidis, "Multilinear compressive learning with prior knowledge," *arXiv preprint arXiv:2002.07203*, 2020.

- [14] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *arXiv preprint arXiv:1808.05377*, 2018.
- [15] D. T. Tran, S. Kiranyaz, M. Gabbouj, and A. Iosifidis, "Heterogeneous multilayer generalized operational perceptron," *IEEE transactions on neural networks and learning systems*, 2019.
- [16] D. T. Tran, S. Kiranyaz, M. Gabbouj, and A. Iosifidis, "Progressive operational perceptrons with memory," *Neurocomputing*, vol. 379, pp. 172–181, 2020.
- [17] S. Kiranyaz, T. Ince, A. Iosifidis, and M. Gabbouj, "Operational neural networks," *Neural Computing and Applications*, pp. 1–24, 2020.
- [18] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. P. Xing, "Neural architecture search with bayesian optimisation and optimal transport," in *Advances in Neural Information Processing Systems*, pp. 2016–2025, 2018.
- [19] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 19–34, 2018.
- [20] T. Domhan, J. T. Springenberg, and F. Hutter, "Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.
- [21] A. Klein, S. Falkner, J. T. Springenberg, and F. Hutter, "Learning curve prediction with bayesian neural networks," 2016.
- [22] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000.
- [23] N. Pinto, Z. Stone, T. Zickler, and D. Cox, "Scaling up biologically-inspired computer vision: A case study in unconstrained face recognition on facebook," in *CVPR 2011 WORKSHOPS*, pp. 35–42, IEEE, 2011.
- [24] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar, "Attribute and simile classifiers for face verification," in *2009 IEEE 12th international conference on computer vision*, pp. 365–372, IEEE, 2009.
- [25] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," in *2004 conference on computer vision and pattern recognition workshop*, pp. 178–178, IEEE, 2004.
- [26] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.
- [27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

### **7.13 Multilinear Compressive Learning with Prior Knowledge**

The appended paper follows.

# Multilinear Compressive Learning with Prior Knowledge

Dat Thanh Tran\*, Moncef Gabbouj\*, Alexandros Iosifidis†

\*Department of Computing Sciences, Tampere University, Tampere, Finland

†Department of Engineering, Aarhus University, Aarhus, Denmark

Email:{thanh.tran, moncef.gabbouj}@tuni.fi, alexandros.iosifidis@eng.au.dk

**Abstract**—The recently proposed Multilinear Compressive Learning (MCL) framework combines Multilinear Compressive Sensing and Machine Learning into an end-to-end system that takes into account the multidimensional structure of the signals when designing the sensing and feature synthesis components. The key idea behind MCL is the assumption of the existence of a tensor subspace which can capture the essential features from the signal for the downstream learning task. Thus, the ability to find such a discriminative tensor subspace and optimize the system to project the signals onto that data manifold plays an important role in Multilinear Compressive Learning. In this paper, we propose a novel solution to address both of the aforementioned requirements, i.e., *How to find those tensor subspaces in which the signals of interest are highly separable?* and *How to optimize the sensing and feature synthesis components to transform the original signals to the data manifold found in the first question?* In our proposal, the discovery of a high-quality data manifold is conducted by training a nonlinear compressive learning system on the inference task. Its knowledge of the data manifold of interest is then progressively transferred to the MCL components via multi-stage supervised training with the supervisory information encoding how the compressed measurements, the synthesized features, and the predictions should be like. The proposed knowledge transfer algorithm also comes with a semi-supervised adaption that enables compressive learning models to utilize unlabeled data effectively. Extensive experiments demonstrate that the proposed knowledge transfer method can effectively train MCL models to compressively sense and synthesize better features for the learning tasks with improved performances, especially when the complexity of the learning task increases.

## I. INTRODUCTION

Compressive Sensing (CS) [1] is an efficient signal acquisition paradigm that performs the measurement of the signal at sub-Nyquist rates by sensing and linearly interpolating the samples at the sensor level. Due to the property that the compression step is performed before signal registration during the sampling phase, CS significantly lowers the temporary storage and bandwidth requirement of the sensing devices. Therefore, this paradigm plays an important role in many applications that involve high-dimensional signals since the signal collection process can be very computationally demanding, time-consuming, or even prohibitive when using the traditional point-based sensing-then-compressing approach [2]. For example, Hyperspectral Compressive Imaging (HCI) [3] and Synthetic Aperture Radar (SAR) imaging [4] often

perform remote sensing of large volume of data constrained by limited electrical power, storage capacity and transmission bandwidth, or in 3D Magnetic Resonance Imaging (MRI) [5], long scanning time can make patients feel uncomfortable, promoting body movements, thus degrading the image quality.

Although the ideal sampling theorem requires the signal to be sampled at higher rates than the Nyquist rate to ensure perfect reconstruction, in CS, the undersampled signal can still be reconstructed almost perfectly if the sparsity assumption holds and the sensing operators possess certain properties [6], [7]. The sparsity prior assumes the existence of a set of basis or a dictionary in which the signal of interest has sparse representations, i.e., having few non-zero coefficients. Sparsity manifests in many classes of signals that we operate on since they are often smooth or piece-wise smooth, thus having sparse representations in Fourier or wavelet domains. This form of prior knowledge incorporated into the model allows us to acquire the signals at a much lower cost.

In fact, over the past decade, a large body of works in CS has been dedicated to *finding* relevant priors from the class of signals of interest, and *incorporating* them into the model to achieve better compression and reconstruction [8]. A typical form of priors that has been widely used in CS is the existence of a signal similar to the original signal of interest. This information is available in many scenarios such as video acquisitions [9], [10] and estimation problems [11] where signals acquired in succession are very similar. Similarity also exists in signals captured by nearby sensors in sensor networks [12] or images from spatially close cameras in multiview camera systems [13], [14]. Other examples of prior information include the available estimate of the support of the signal [15], Gaussian mixture models [16], weighted sparsity [17] or block sparsity [18], [19]. The process of *finding* prior information often relies on the available prior knowledge related to the signal, e.g., the spatial arrangement of sensor networks, while the *incorporation* of such information is often expressed as another term to be minimized such as the difference between the current frame and the last frame when reconstructing MRI images. For a detailed discussion on prior information in CS and its theoretical and empirical improvements, we refer the readers to [8].

While signal reconstruction is a necessary processing step

in some applications, there exist many scenarios in which the primary purpose is to detect certain patterns or to infer some properties from the acquired signal, rather than the recovery of the entire signal. Learning from compressed measurements, also known as Compressive Learning (CL) [20], [21], [22], [23], [24], [25], [26], focuses on the task of inference directly from compressive domain without explicit signal reconstruction. That is, CL models are often formed under the assumption that only certain aspects or representative features of the signal need to be captured to make inferences, even when perfect signal recovery might be impossible from the retained information. It is worth noting that in many scenarios such as security and surveillance applications [27], [28], signal recovery might disclose privacy information. Thus, it is undesirable.

Since the main objective of CL is to extract relevant information given the learning problem, the literature in CL mainly concerns with the selection of the sensing operators [29], [30] or the optimization of the learning model [21] that operates in the compressive domain. In the early works, the selection or design of sensing matrices was decoupled from the optimization of the inference model. Nowadays, with the developments on both hardware and algorithmic levels, stochastic optimization has become more and more efficient, making end-to-end learning systems applicable and appealing to a wide range of applications. Initiated by the work of [31], [25], recent CL systems [32], [33], [34], [26] have adopted an end-to-end approach which jointly optimizes the sensing matrices and the inference operators, leaving the task of discovering relevant features to stochastic gradient descent.

The majority of CL systems employs linear sensing operators which operate on the vector-based input, regardless of the natural representation of the signal. Recently, the authors in [26] proposed Multilinear Compressive Learning (MCL) framework, which takes into consideration the natural structure of multidimensional signals by adopting multilinear sensing and feature extraction components. MCL has been shown both theoretically and empirically to be more efficient than its vector-based counterpart. The assumption made in MCL is that the original signals can be linearly projected along each tensor mode to a tensor subspace which retains relevant information for the learning problem, and from which discriminative features can be synthesized. In a general setting, the authors in [26] propose to initialize the sensing matrices with the left singular vectors obtained from Higher Order Singular Value Decomposition (HOSVD), a kind of *weak prior* that initially guides the model towards an energy-preserving tensor subspace during stochastic optimization.

As we have seen from CS literature, finding good priors and successfully incorporating them into the signal model lead us to better solutions. While prior information exists in many forms in the reconstruction task, it is not straightforward to define relevant information given the learning problem. Indeed, representation learning [35] has been an active research area that aims to extract meaningful representations from the raw data, ideally without any human given label such as object

categories. From the human perception, in certain cases, we do have an idea of what kind of information is relevant or irrelevant to the learning task, e.g., color cues might be irrelevant in face recognition problems but relevant for the traffic light detection problem. From the optimization point of view, it is hard to make such a claim before any trial. Therefore, we often want to strike a balance between hand-crafted features and end-to-end solutions.

In this paper, we aim to tackle the problems of *How to find good priors in Compressive Learning?* and *How to incorporate such priors into an end-to-end Compressive Learning system without hard-coding?* Based on two observations:

- Although the sensing operators in CL are limited to linear/multilinear form, the feature synthesis component that operates on compressed measurements can adopt nonlinear transformations.
- Nonlinear sensing operators have better capacities to discover representative tensor subspaces

we propose a novel approach that utilizes nonlinear compressive learning models to discover the structures of the compressive domain and the informative features that should be synthesized from this domain for the learning task. This knowledge is subsequently transferred to the compressive learning models via progressive Knowledge Distillation [36]. Our contributions can be summarized as follows:

- In this paper, a novel methodology to discover and incorporate prior knowledge into existing end-to-end Compressive Learning systems is proposed. Although we limit our investigation of the proposed method to Multilinear Compressive Learning framework, the approach described in this paper can be applied to any end-to-end Compressive Learning system to improve its performance.
- The proposed approach naturally leads to the semi-supervised extension in which the availability of unlabeled signals coming from similar distributions can benefit Compressive Learning systems via prior knowledge incorporation.
- We carefully designed and conducted extensive experiments in object classification and face recognition tasks to investigate the effectiveness of the proposed approach to learn and incorporate prior information. In addition, to facilitate future research and reproducibility, we publicly provide our implementation of all experiments reported in this paper<sup>1</sup>.

The remainder of our paper is organized as follows: in Section II, we review the related works in Compressive Learning, Knowledge Distillation, and give a brief description of Multilinear Compressive Learning framework. Section III provides a detailed description of our approach to learn and incorporate prior knowledge into MCL models in both supervised and semi-supervised settings. Section IV details our empirical analysis, including experiment protocols, experiment designs, and quantitative and qualitative evaluation. Section V concludes our work.

<sup>1</sup><https://github.com/viebbboy/MultilinearCompressiveLearningWithPrior>

## ABBREVIATION & NOMENCLATURE

CS	Compressive Sensing
CL	Compressive Learning
MCL	Multilinear Compressive Learning
MCLwP	Multilinear Compressive Learning with Priors
HOSVD	Higher Order Singular Value Decomposition
KD	Knowledge Distillation
FS	Feature Synthesis
$\mathcal{Y}$	Multi-dimensional analog signal
$\mathbf{z}$	Compressed measurements (vector) in CL
$\mathcal{Z}$	Compressed measurements (tensor) in MCL
$\mathcal{T}$	Tensor features, output of FS component in MCL
E	CS component in MCL
$\theta_E$	Parameters of E
D	FS component in MCL
$\theta_D$	Parameters of D
N	Task-specific neural network in MCL
$\theta_N$	Parameters of N
P	Prior-generating model
$\mathcal{Z}_P$	Compressed measurements in P
$\mathcal{T}_P$	Tensor features, output of FS component in P
$E_P$	Nonlinear sensing component in P
$\theta_{E_P}$	Parameters of $E_P$
$D_P$	FS component in P
$\theta_{D_P}$	Parameters of $D_P$
$N_P$	Task-specific neural network in P
$\theta_{N_P}$	Parameters of $N_P$
$\mathcal{Y}_i$	$i$ -th data sample
$c_i$	label of the $i$ -th sample
$N$	Number of labeled samples
$M$	Total number of samples, including labeled and unlabeled samples
$\mathcal{L}$	The set of labeled data
$\mathcal{U}$	The set of unlabeled data
$\hat{\mathcal{L}}$	Enlarged labeled set
$L_I$	Inference loss function
$L_D$	Distillation loss function

## II. RELATED WORK

### A. Background

Throughout the paper, we denote scalar values by either lower-case or upper-case characters ( $x, y, X, Y \dots$ ), vectors by lower-case bold-face characters ( $\mathbf{x}, \mathbf{y}, \dots$ ), matrices by upper-case or Greek bold-face characters ( $\mathbf{A}, \mathbf{B}, \Phi, \dots$ ) and tensor as calligraphic capitals ( $\mathcal{X}, \mathcal{Y}, \dots$ ). A tensor with  $K$  modes and dimension  $I_k$  in the mode- $k$  is represented as  $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_K}$ . The entry in the  $i_k$ -th index in mode- $k$  for  $k = 1, \dots, K$  is denoted as  $\mathcal{X}_{i_1, i_2, \dots, i_K}$ . In addition,  $\text{vec}(\mathcal{X})$  denotes the vectorization operation that rearranges elements in  $\mathcal{X}$  to the vector representation.

The mode- $k$  product between a tensor  $\mathcal{X} \in \mathbb{R}^{I_1 \times \dots \times I_K}$  and a matrix  $\mathbf{W} \in \mathbb{R}^{J_k \times I_k}$  is another tensor of size  $I_1 \times \dots \times J_k \times \dots \times I_K$  and denoted by  $\mathcal{X} \times_k \mathbf{W}$ . The element of

$\mathcal{X} \times_k \mathbf{W}$  is defined as  $[\mathcal{X} \times_k \mathbf{W}]_{i_1, \dots, i_{k-1}, j_k, i_{k+1}, \dots, i_K} = \sum_{i_k=1}^{I_k} [\mathcal{X}]_{i_1, \dots, i_{k-1}, i_k, \dots, i_K} [\mathbf{W}]_{j_k, i_k}$ .

In CS, given the original analog signal  $\mathbf{y} \in \mathbb{R}^I$ , signal acquisition model is described via the following equation:

$$\mathbf{z} = \Phi \mathbf{y} \quad (1)$$

where  $\mathbf{z} \in \mathbb{R}^M$ ,  $M \ll I$  denotes the measurements obtained from CS devices and  $\Phi$  denotes the sensing operator or sensing matrix that performs linear interpolation of  $\mathbf{y}$ .

For a multidimensional signal  $\mathcal{Y} \in \mathbb{R}^{I_1 \times \dots \times I_K}$ , we have Multilinear Compressive Sensing (MCS) model as follows:

$$\mathcal{Z} = \mathcal{Y} \times_1 \Phi_1 \times \dots \times_K \Phi_K \quad (2)$$

where  $\mathcal{Z} \in \mathbb{R}^{M_1 \times \dots \times M_K}$ ,  $M_k \ll I_k, \forall k = 1, \dots, K$  is the compressed measurements having a tensor form, and  $\Phi_k \in \mathbb{R}^{M_k \times I_k}$  denotes separable sensing operators that perform linear transformation along each mode of the original signal  $\mathcal{Y}$ .

Since Eq. (2) can also be written as:

$$\mathbf{z} = (\Phi_1 \otimes \dots \otimes \Phi_K) \mathbf{y} \quad (3)$$

where  $\mathbf{z} = \text{vec}(\mathcal{Z})$ ,  $\mathbf{y} = \text{vec}(\mathcal{Y})$ , and  $\otimes$  denotes the Kronecker product, we can always express MCS in the vector-based fashion, but not vice versa.

### B. Compressive Learning

The objective of Compressive Learning is to create learning models that generate predictions from the compressed measurements  $\mathbf{z}$  or  $\mathcal{Z}$ . This idea was first proposed in the early work of [21], where the authors train a classifier directly on the compressed measurements without the signal reconstruction step. It has been shown in this work that when the number of measurements approximates the dimensionality of the data manifold, accurate classifiers can be trained directly in the compressive domain. To achieve good classification performance, an extension of [21] later showed that the number of measurements only depends on the intrinsic dimensionality of the data manifold [37].

A similar theoretical result was derived later in [20], which proves that the performance of a linear Support Vector Machine classifier trained on compressed measurements is equivalent to the best linear threshold classifier operating in the original signal domain, given the Distance-Preserving Property holds for the sensing matrices. Asymptotic behavior and sensing operator optimization in CL models of signals described by Gaussian Mixture Model have also been proposed in [23], [29].

The idea of using past measurements as a prior to jointly learn better sensing matrices and the classifier was proposed in [30]. Since the advancement in stochastic optimization, linear classifiers and shallow learning models have been replaced with deep neural networks [24], [31], and separate optimization of the sensing operators and the classifiers have been replaced by the end-to-end training paradigm, which has been

shown to significantly outperform predefined sensing operators when the compression rates are high [25], [32], [33], [34], [26].

### C. Multilinear Compressive Learning

While previous works in learning from compressed measurements adopt the signal acquisition model in Eq. (1), recently, the authors in [26] proposed Multilinear Compressive Learning (MCL), a framework that utilizes Eq. (2) as the signal acquisition model in order to retain the multidimensional structure of the original signal.

MCL consists of three components: the CS component described by Eq. (2), the Feature Synthesis (FS) component that transforms the measurements  $\mathcal{Z}$  to tensor feature  $\mathcal{T}$  via:

$$\mathcal{T} = \mathcal{Z} \times_1 \Theta_1 \times \cdots \times_K \Theta_K \quad (4)$$

and a task-specific neural network  $N$  that operates on  $\mathcal{T}$  to generate prediction, i.e., the prediction is  $N(\mathcal{T})$ .

Since the transformation in Eq. (2) preserves the tensor structure of  $\mathcal{Y}$ , MCL assumes the existence of a tensor subspace which captures the essential information in  $\mathcal{Y}$  through  $\mathcal{Z}$ , and from which discriminative features can be synthesized.

Similar to the end-to-end vector-based framework [25], [34], MCL incorporates a kind of *weak prior* that initializes  $\Phi_k$  and  $\Theta_k$  with energy-preserving values, i.e., the singular vectors corresponding to the largest singular values in mode- $k$  obtained from HOSVD of the training data. The task-specific network  $N$  is initialized with the weights trained on uncompressed signals. The whole system is then optimized via stochastic gradient descent. As pointed out in the empirical analysis in [26], although simple, this initialization scheme contributes significantly to the performances of the system compared to the de facto random initialization scheme often utilized in deep neural networks [38].

### D. Knowledge Distillation

The term Knowledge Distillation (KD) was coined in [36], in which the authors proposed a neural network training technique that utilizes the prediction from a pre-trained high capacity network (the teacher network) to provide supervisory signals to a smaller network (the student network) along with labeled data. The intuition behind this training technique is that with higher capacity, it is easier for the teacher network to discover better data representation, and the form of knowledge provided via the teacher network's prediction helps guide the student network to better solutions.

There have been several works investigating variants of this technique. For example, KD that involves multiple student and teacher networks [39] has been shown to be more robust than a single teacher-student pair. In other works [40], [41], [42], in addition to the predicted probabilities, knowledge coming from intermediate layers of the teacher network has also been proven to be useful for the student. While KD has often been considered in the context of model compression, i.e., to train low capacity models with better performances, this paradigm has also been successfully applied to distributed training with

student and teacher networks having the same architecture [43].

In our work, we propose to use KD as a method to progressively incorporate prior information into CL models. While in a general setting it is unclear how to select the immediate source and target layers to transfer knowledge from the teacher to the student, we will show next that in the context of CL, and especially in the MCL framework, pairing the intermediate teachers and students is easy since the learning system is modularized into different components with different functionalities.

## III. PROPOSED METHODS

### A. Finding Prior Knowledge

A common constraint which exists in all CL models is that the sensing operator is linear or multilinear. This limits the capacity of CL models and the amount of information that can be captured from the original signal. Thus, in CL in general and MCL in particular, given a fixed structure of the Feature Synthesis (FS) component and the task-specific neural network  $N$ , there is always an upper-bound on the capacity of the whole learning system, which makes the problem of learning from compressed measurements  $\mathcal{Z}$  generally harder than other unconstrained learning tasks.

As mentioned in the Introduction Section, finding and incorporating prior knowledge is a recurring theme in CS community since good priors have been shown both theoretically and empirically to improve signal recovery and compression efficiency [8]. In case of CL, prior knowledge is less obvious. However, we may still formulate the following question: *What kind of prior knowledge do we know about the measurements  $\mathcal{Z}$  and the types of features  $\mathcal{T}$  that are representative for the learning task?*

Although we might not have direct knowledge of optimal  $\mathcal{Z}$  or  $\mathcal{T}$ , we know as a prior knowledge that, given sufficient training data, a higher capacity neural network is expected to perform better than a less complex one when both networks are structurally similar. By structure here we mean that both networks have similar connectivity patterns and by higher capacity, we mean higher number of parameters and/or higher number of layers. The structural similarity ensures a feasible capacity comparison between networks since one cannot make such a guess about the learning ability, e.g., between a feed-forward architecture having a higher parameter count and a residual architecture having a lower parameter count. Therefore, we almost know for certain that given an MCL model with upper-bounded learning capacity, one can construct and train other learning systems with similar FS component and task-specific network  $N$ , but having nonlinear sensing operators to achieve better learning performance, thus better  $\mathcal{Z}$  and  $\mathcal{T}$ . The representations produced by higher capacity, nonlinear compressive learning models become our direct prior knowledge of the compressive domain and the feature space.

The advantage of the above presented approach to define prior knowledge in MCL is two-fold:

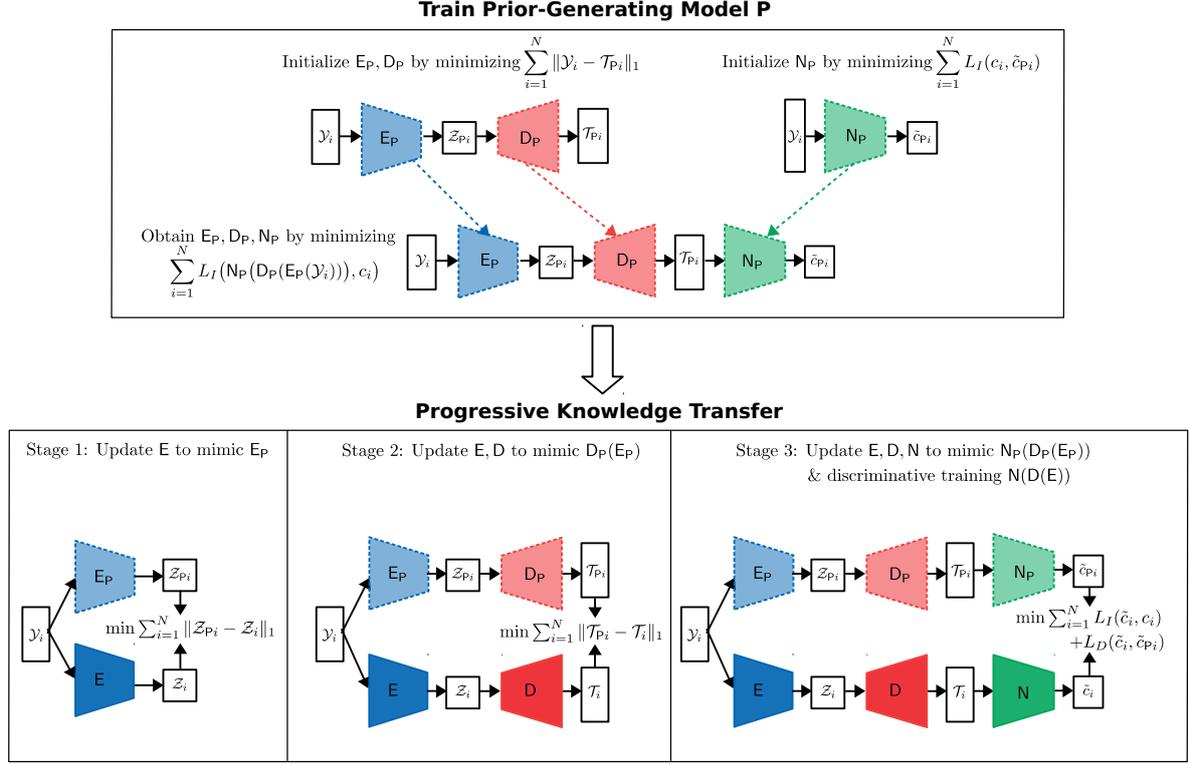


Fig. 1. Illustration of the proposed algorithm: we first train prior-generating model P. The knowledge in each component of P is then progressively transferred to the MCL model. Lighter color blocks indicate components of P while darker color ones indicate the corresponding components in MCL model.

- Since the priors are generated from another learning system trained on the same learning task, we incorporate completely data-dependent, task-specific priors into our model instead of hand-crafted priors.
- By sharing the same structure of the FS component and task-specific neural network N between the MCL model and the prior-generating model, the weights obtained from the latter provide a good initialization when optimizing the former.

The proposed approach reduces the problem of finding priors to the task of designing and training a nonlinear compressive learning model; hereupon we refer to it as the prior-generating model P, which has no structural constraint. To construct and optimize P, we propose to mirror the strategy in MCL [26]:

- The nonlinear sensing component of P, denoted as  $E_P$  with parameters  $\theta_{E_P}$ , has several layers to reduce the dimensionality of the original signal gradually.  $E_P$  transforms the input signal  $\mathcal{Y}$  to a compressed representation  $\mathcal{Z}_P$ , i.e.,  $\mathcal{Z}_P = E_P(\mathcal{Y}; \theta_{E_P})$ , which has the same size as  $\mathcal{Z}$ .
- Similarly, the nonlinear FS component of P, denoted as  $D_P$  with parameters  $\theta_{D_P}$ , has several layers to increase

the dimensionality of  $\mathcal{Z}_P$  gradually, producing features  $\mathcal{T}_P = D_P(\mathcal{Z}_P; \theta_{D_P})$ , which has the same dimensionality as the original signal.  $E_P$  and  $D_P$  together resemble the encoder and decoder of a nonlinear autoencoder.

- Denote  $\mathcal{Y}_i$  and  $c_i$  ( $i = 1, \dots, N$ ) the  $i$ -th training sample and its label. In order to initialize the weights  $\theta_{E_P}$  and  $\theta_{D_P}$ , we update  $E_P$  and  $D_P$  to minimize  $\sum_{i=1}^N \|\mathcal{Y}_i - D_P(E_P(\mathcal{Y}_i))\|_1$  via stochastic gradient descent. The weights  $\theta_{N_P}$  of task-specific network  $N_P$  are initialized with weights minimizing  $\sum_{i=1}^N L_I(N_P(\mathcal{Y}_i), c_i)$  with  $L_I(\cdot)$  is the inference loss.
- After the initialization step,  $\theta_{E_P}$ ,  $\theta_{D_P}$ , and  $\theta_{N_P}$  are updated altogether via stochastic gradient descent to minimize  $\sum_{i=1}^N L_I(N_P(D_P(E_P(\mathcal{Y}_i))), c_i)$

### B. Incorporating Prior Knowledge

Let us denote the CS, FS and task-specific neural network component of the MCL model as E, D and N, having learnable parameters  $\theta_E$ ,  $\theta_D$  and  $\theta_N$ , respectively. It should be noted that D and N have the same functional form as  $D_P$  and  $N_P$  while E possesses a multilinear form as in Eq. (2).

After training the prior-generating model P, we perform progressive KD to transfer the knowledge of P to the MCL model. During this phase, the parameters of P, i.e.,  $\theta_{E_P}$ ,  $\theta_{D_P}$ ,

---

**Algorithm 1** Training MCL Model via Knowledge Transfer Algorithm in Supervised Learning Setting

---

- 1: **Inputs:**
  - 2: Labeled data set  $\mathcal{L} = \{(\mathcal{Y}_i, c_i) | i = 1, \dots, N\}$ .
  - 3: Structure of prior-generating model  $E_P, D_P, N_P$
  - 4: Structure of MCL model  $E, D, N$
  - 5: Inference loss  $L_I$ , Distillation loss  $L_D$
  - 6: **Training prior-generating model P:**
  - 7: Initialize  $\theta_{N_P}$  with values from
  - 8:  $\arg \min_{\theta_{N_P}} \sum_{i=1}^N L_I(N_P(\mathcal{Y}_i), c_i)$
  - 9: Initialize  $\theta_{E_P}$  and  $\theta_{D_P}$  with values from
  - 10:  $\arg \min_{\theta_{E_P}, \theta_{D_P}} \sum_{i=1}^N \|\mathcal{Y}_i - D_P(E_P(\mathcal{Y}_i))\|_1$
  - 11: Obtain  $\theta_{E_P}, \theta_{D_P}, \theta_{N_P}$  by solving
  - 12:  $\arg \min_{\theta_{E_P}, \theta_{D_P}, \theta_{N_P}} \sum_{i=1}^N L_I(N_P(D_P(E_P(\mathcal{Y}_i))), c_i)$
  - 13: **Training MCL model:**
  - 14: After obtaining  $\theta_{E_P}, \theta_{D_P}, \theta_{N_P}$ , they are fixed.
  - 15: Transfer sensing knowledge:
  - 16: Update  $\theta_E$  by solving Eq. (5)
  - 17: Transfer feature synthesis knowledge:
  - 18: Update  $\theta_E, \theta_D$  by solving Eq. (6)
  - 19: Transfer inference knowledge & discriminative training:
  - 20: Update  $\theta_E, \theta_D, \theta_N$  by solving Eq. (7)
  - 21: **Outputs:** Parameters of MCL model:  $\theta_E, \theta_D, \theta_N$
- 

and  $\theta_{N_P}$  are fixed. Since the teacher (P) and the student (MCL) model share the same modular structure which has three distinct components, training MCL with prior knowledge given by P consists of three stages:

- **Stage 1** (Transferring the sensing knowledge from  $E_P$  to  $E$ ): during this stage, the weights  $\theta_E$  of the sensing component  $E$  in MCL are obtained by optimizing the following criterion:

$$\arg \min_{\theta_E} \sum_{i=1}^N \|E_P(\mathcal{Y}_i; \theta_{E_P}) - E(\mathcal{Y}_i; \theta_E)\|_1 \quad (5)$$

The purpose of this stage is to enforce the sensing component in MCL to mimic that of the prior-generating model P.

- **Stage 2** (Transferring the feature synthesis knowledge from  $D_P(E_P)$  to  $D(E)$ ): during this stage, we aim to optimize  $E$  and  $D$  in MCL to synthesize similar features produced by the prior-generating model. Before the optimization process, the weights of the sensing component ( $\theta_E$ ) are initialized with values obtained from the 1<sup>st</sup> stage, and the weights of the feature synthesis component  $\theta_D$  are initialized with values from  $\theta_{D_P}$  since  $D$  in MCL and  $D_P$  in P have the same functional form. After the initialization step, both sensing ( $\theta_E$ ) and feature synthesis ( $\theta_D$ ) components in MCL are updated together

to minimize the following criterion:

$$\arg \min_{\theta_E, \theta_D} \sum_{i=1}^N \|D_P(E_P(\mathcal{Y}_i; \theta_{E_P}); \theta_{D_P}) - D(E(\mathcal{Y}_i; \theta_E); \theta_D)\|_1 \quad (6)$$

- **Stage 3** (Transferring the inference knowledge from  $N_P(D_P(E_P))$  to  $N(D(E))$  and discriminative training  $N(D(E))$ ): in this final stage, the MCL model is trained to minimize the inference loss as well as the difference between its prediction and the prediction produced by the prior-generating model P. That is, the minimization objective in this stage is:

$$\arg \min_{\theta_E, \theta_D, \theta_N} \sum_{i=1}^N L_I(N(D(E(\mathcal{Y}_i))), c_i) + \lambda L_D(N_P(D_P(E_P(\mathcal{Y}_i))), N(D(E(\mathcal{Y}_i)))) \quad (7)$$

where  $L_I(\cdot)$  and  $L_D(\cdot)$  denote the inference loss and distillation loss, respectively.  $\lambda$  is a hyper-parameter that allows the adjustment of distillation loss. The specific form of  $L_I(\cdot)$  and  $L_D(\cdot)$  are chosen depending on the inference problem. For example, in classification problems, we can select cross-entropy function for  $L_I(\cdot)$  and Kullback-Leibler divergence for  $L_D(\cdot)$  while in regression problems,  $L_I(\cdot)$  and  $L_D(\cdot)$  can be mean-squared-error or mean-absolute-error function. To avoid lengthy representation, Eq. (7) omits the parameters of each component. Here we should note that before optimizing Eq. (7), parameters  $\theta_E$  and  $\theta_D$  of the sensing and synthesis component in MCL are initialized with values obtained from the previous stage, while parameters of the task-dependent neural network  $\theta_N$  in MCL is initialized with values from  $\theta_{N_P}$  in the prior-generating model P.

The pseudo-code and the illustration of our proposed algorithm to train MCL model via progressive knowledge transfer is presented in Algorithm 1 and Figure 1, respectively.

To use KD and the prior-generating model P, a simpler and more straightforward approach is to directly distill the predictions of P to the MCL model as in the 3<sup>rd</sup> stage described above, skipping the 1<sup>st</sup> and 2<sup>nd</sup> stages. However, by gradually transferring knowledge from each component of the prior-generating model P to the MCL model, we argue that the proposed training scheme directly incorporates the knowledge of a good compressive domain and feature space, facilitating the MCL model to mimic the internal representations of its teacher, which is better than simply imitating its teacher's predictions. To validate our argument, we provide empirical analysis of the effects of each training stage in Section IV-F.

The aforementioned progressive knowledge transfer scheme also allows us to directly use the predictions produced by the teacher model in the 3<sup>rd</sup> stage, instead of the “softened” predictions as in the original work [36], eliminating the need to select the associated temperature hyper-parameter to soften the teacher’s predictions.

### C. Semi-supervised Learning Extension

It should be noted that without sufficient labeled data, training a high-capacity prior-generating model as proposed in Section III-A can lead to an over-fitted teacher, which might provide misleading supervisory information to the MCL model. In addition, a limited amount of data might also prevent effective knowledge transfer from the teacher to the student model. In certain scenarios, labeled data is scarce, however, we can easily obtain a large amount of data coming from the same distribution without labels. Semi-supervised learning refers to the learning paradigm that takes advantages of unlabeled data, usually available in abundance, along with a limited amount of labeled data. Here we also describe a semi-supervised adaptation for the above MCL model training technique to remedy possible over-fitting cases and improve generalization of both prior-generating and MCL models in classification tasks. To this end, unlabeled data is utilized to both initialize, and then optimize the weights of the prior-generating model P via an incremental self-labeling procedure. Subsequently, when transferring knowledge to the MCL model, the class predictions of P on unlabeled data are used as *hard* labels in the inference loss term, while its probability predictions on unlabeled data are used as *soft* labels in the distillation loss term.

Let us denote  $\mathcal{L} = \{(\mathcal{Y}_i, c_i) | i = 1, \dots, N\}$  the labeled training set and  $\mathcal{U} = \{\mathcal{Y}_i | i = N + 1, \dots, M\}$  the unlabeled training set. To take advantage of  $\mathcal{L} \cup \mathcal{U}$ , we propose the following modifications to the training procedure of prior-generating model P:

- Initialization of  $E_P$  and  $D_P$ : the weights of the sensing ( $\theta_{E_P}$ ) and feature synthesis ( $\theta_{D_P}$ ) component are initialized with values obtained from minimizing the reconstruction error on  $\mathcal{L} \cup \mathcal{U}$ , i.e.,  $\sum_{i=1}^M \|\mathcal{Y}_i - D_P(E_P(\mathcal{Y}_i))\|_1$ .
- Incremental optimization of  $E_P, D_P, N_P$  via self-labeling: after the initialization step, all parameters of the prior-generating model P are optimized with respect to the inference loss, which is calculated on the enlarged labeled set  $\tilde{\mathcal{L}}$ :

$$\arg \min_{\theta_{E_P}, \theta_{D_P}, \theta_{N_P}} \sum_{(\mathcal{Y}_i, c_i) \in \tilde{\mathcal{L}}} L_I(N_P(D_P(E_P(\mathcal{Y}_i))), c_i) \quad (8)$$

Initially, the enlarged labeled set is formed from the labeled data, i.e.,  $\tilde{\mathcal{L}} = \mathcal{L}$ . After every  $T$  backpropagation epochs,  $\tilde{\mathcal{L}}$  is augmented with those data instances (with their predicted labels) in  $\mathcal{U}$  that have the most confident predictions from the current P, given a probability threshold  $\rho$ , i.e.,  $\tilde{\mathcal{L}} = \tilde{\mathcal{L}} \cup \mathcal{C}$ :

$$\mathcal{C} = \{(\mathcal{Y}, c) | \mathcal{Y} \in \mathcal{U} \wedge N_P(D_P(E_P(\mathcal{Y})))_{\max} \geq \rho, \quad (9)$$

$$c = \operatorname{argmax}(N_P(D_P(E_P(\mathcal{Y}))))\}$$

After the enlargement of  $\tilde{\mathcal{L}}$  with the most confident instances, they are removed from the unlabeled set  $\mathcal{U}$ , i.e.,  $\mathcal{U} = \mathcal{U} \setminus \mathcal{C}$ . The training terminates when the enlargement of  $\tilde{\mathcal{L}}$  stops, i.e.,  $\mathcal{C} = \emptyset$ .

Self-labeling is a popular technique in semi-supervised algorithms. While there are many sophisticated variants of this technique [44], the simple modifications proposed above work well as illustrated in our empirical study in Section IV-G.

Given a prior-generating model P trained on  $\mathcal{L} \cup \mathcal{U}$ , in order to adapt the progressive knowledge transfer algorithm proposed in Section III-B to the semi-supervised setting, we propose to replace the objectives in Eq. (5), (6) and (7) with the following objectives respectively:

$$\arg \min_{\theta_E} \sum_{i=1}^M \|E_P(\mathcal{Y}_i; \theta_{E_P}) - E(\mathcal{Y}_i; \theta_E)\|_1 \quad (10)$$

and

$$\arg \min_{\theta_E, \theta_D} \sum_{i=1}^M \|D_P(E_P(\mathcal{Y}_i; \theta_{E_P}); \theta_{D_P}) - D(E(\mathcal{Y}_i; \theta_E); \theta_D)\|_1 \quad (11)$$

and

$$\arg \min_{\theta_E, \theta_{D_P}, \theta_{N_P}} \sum_{i=1}^M L_I(N(D(E(\mathcal{Y}_i))), c_i) + \lambda L_D(N_P(D_P(E_P(\mathcal{Y}_i))), N(D(E(\mathcal{Y}_i)))) \quad (12)$$

where in (12),  $c_i$  denotes the class label predicted by the prior-generating model P for  $i = N + 1, \dots, M$ .

A summary of our proposed algorithm in the semi-supervised setting is presented in Algorithm 2.

## IV. EXPERIMENTS

In this Section, we provide a detailed description and results of our empirical analysis which demonstrates the advantages of the proposed algorithms that incorporate data-dependent prior knowledge into the training procedure of MCL models, compared to [26]. For this purpose, we provide various comparisons between the proposed algorithm and the original algorithm proposed in [26] in the supervised learning setting. The experiments are designed to benchmark the learning performances and the quality of signal representation in compressive domain produced by the two competing algorithms, with increasing difficulty in the learning tasks. In addition, we also study the significance of prior-generating model and different knowledge transfer stages via different ablation experiments. Lastly, we demonstrate the necessity of the proposed modifications presented in Section III-C to our algorithm in the semi-supervised learning setting, i.e., when the databases are large, but only limited amounts of labeled data exist.

### A. Datasets

We conducted experiments on the object classification and face recognition tasks of the following datasets:

- CIFAR dataset [45] is a color (RGB) image dataset used for evaluating object recognition methods. The dataset consists of 50K images for training and 10K images for testing with resolution  $32 \times 32$  pixels. In our work, CIFAR-10 refers to the 10-class objection recognition

---

**Algorithm 2** Training MCL Model via Knowledge Transfer Algorithm in Semi-supervised Learning Setting

---

```

1: Inputs:
2:   Labeled data set  $\mathcal{L} = \{(\mathcal{Y}_i, c_i) | i = 1, \dots, N\}$ .
3:   Unlabeled data set  $\mathcal{U} = \{\mathcal{Y}_i | i = N + 1, \dots, M\}$ .
4:   Structure of prior-generating model  $E_P, D_P, N_P$ 
5:   Structure of MCL model  $E, D, N$ 
6:   Inference loss  $L_I$ , Distillation loss  $L_D$ 
7:   Number of backpropagation iteration  $T$ 
8:   Confidence probability threshold  $\rho$ 
9: Training prior-generating model P:
10:  Initialize  $\theta_{N_P}$  with values from
11:     $\arg \min_{\theta_{N_P}} \sum_{i=1}^N L_I(N_P(\mathcal{Y}_i), c_i)$ 
12:  Initialize  $\theta_{E_P}$  and  $\theta_{D_P}$  with values from
13:     $\arg \min_{\theta_{E_P}, \theta_{D_P}} \sum_{i=1}^M \|\mathcal{Y}_i - D_P(E_P(\mathcal{Y}_i))\|_1$ 
14:  Let  $\tilde{\mathcal{L}} = \mathcal{L}$ 
15:  while True do
16:    Update  $\theta_{E_P}, \theta_{D_P}, \theta_{N_P}$  by solving Eq. (8) for  $T$ 
    iterations
17:    Find set of confident instances according to Eq. (9)
18:    if  $\mathcal{C} = \emptyset$  then
19:      break
20:    Augment enlarged labeled set  $\tilde{\mathcal{L}} = \tilde{\mathcal{L}} \cup \mathcal{C}$ 
21:    Reduce unlabeled set  $\mathcal{U} = \mathcal{U} \setminus \mathcal{C}$ 
22: Training MCL model:
23:   After obtaining  $\theta_{E_P}, \theta_{D_P}, \theta_{N_P}$ , they are fixed.
24:   Transfer sensing knowledge:
25:   Update  $\theta_E$  by solving Eq. (10)
26:   Transfer feature synthesis knowledge:
27:   Update  $\theta_E, \theta_D$  by solving Eq. (11)
28:   Transfer inference knowledge & discriminative training:
29:   Update  $\theta_E, \theta_D, \theta_N$  by solving Eq. (12)
30: Outputs: Parameters of MCL model:  $\theta_E, \theta_D, \theta_N$ 

```

---

task in which each individual image has a single class label coming from 10 different categories. A more fine-grained and difficult classification task also exists in CIFAR dataset with each image having a label coming from 100 different categories, which we denote as CIFAR-100. Since there is no validation set in the original database, in our experiments, we randomly selected 5K images from the training set of CIFAR-10 and CIFAR-100 for validation and only trained the algorithms on 45K images.

- **CelebA:** CelebA [46] is a large-scale human face image dataset with more than 200K images at different resolutions from more than 10K identities. In our experiment, we created three versions of CelebA with increasing difficulties by increasing the set of identities to be recognized: CelebA-100, CelebA-200, and CelebA-500 having 100, 200, and 500 identities respectively. Here we note that CelebA-100 is a subset of CelebA-200, and CelebA-200

is a subset of CelebA-500.

- To study the performances of the proposed algorithm in semi-supervised settings, we also created CIFAR-10S, CIFAR-100S, CelebA-500S, which have the same number of training instances as CIFAR-10, CIFAR-100, and CelebA-500, respectively, but only 20% of them are labeled. The test sets in CIFAR-10S, CIFAR-100S and CelebA-500S are the same as those in CIFAR-10, CIFAR-100, and CelebA-500.

The information of all datasets used in our experiments is summarized in Table I.

### B. Experiment Protocols

In this Section, to differentiate between MCL models trained by different algorithms, we use the abbreviation MCL to refer to the original algorithm proposed in [26], MCLwP to refer to our proposed algorithm that trains MCL models with a prior-generating model with labeled data only, and MCLwP-S to refer to our proposed algorithm that can take advantages of additional unlabeled data.

Regarding the architectural choices in MCL, MCLwP, MCLwP-S, we adopted the AllCNN architecture [26] for the task-specific neural network component in all algorithms. In MCL, the CS and FS components both perform multilinear transformation while in MCLwP and MCLwP-S, the CS component performs multilinear transformation and the FS component performs nonlinear transformation which consists of both convolution and multilinear projection. The exact network architecture used for each algorithm can be found in our publicly available implementation<sup>2</sup>.

To compare the learning performances at different measurement rates, we have conducted experiments at four measurement rates 0.01, 0.02, 0.05 and 0.10, corresponding to the following configurations of  $\mathcal{Z}$ :  $6 \times 6 \times 1$ ,  $9 \times 7 \times 1$ ,  $13 \times 12 \times 1$ , and  $14 \times 11 \times 2$ , respectively.

Since all learning tasks are classification tasks, we represented the labels with one-hot encoding and used cross-entropy function and symmetric Kullback-Leibler divergence for the inference loss  $L_I$  and distillation loss  $L_D$ , respectively. For each experiment configuration, we performed three runs, and the median values of accuracy on the test set are reported. For  $\lambda$ , the hyper-parameter that controls the amount of distillation loss in Eq. (7), we set it equal to 1.0. Confidence probability threshold  $\rho$  in semi-supervised learning experiments was selected from the set  $\{0.7, 0.8, 0.9\}$ .

Regarding stochastic optimization protocol, we used ADAM optimizer [47] with the following learning rate schedule  $\{10^{-3}, 10^{-4}, 10^{-5}\}$ , changing at epochs 80 and 120. Each stochastic optimization procedure was conducted for 160 epochs in total. Max-norm constraint with a value of 6.0 was used to regularize the parameters in all networks. No data preprocessing was conducted, except the scaling of all pixel values to  $[0, 1]$ . During stochastic optimization, we performed random flipping on the horizontal axis and image shifting

<sup>2</sup><https://github.com/viebbboy/MultilinearCompressiveLearningWithPrior>

TABLE I  
DATASETS INFORMATION

Dataset	Input Dimension	Output Dimension	#Labeled Train	#Unlabeled Train	#Validation	#Test
CIFAR-10	$32 \times 32 \times 3$	10	45000	0	5000	10000
CIFAR-100	$32 \times 32 \times 3$	100	45000	0	5000	10000
CelebA-100	$32 \times 32 \times 3$	100	7063	0	2373	2400
CelebA-200	$32 \times 32 \times 3$	200	14305	0	4807	4860
CelebA-500	$32 \times 32 \times 3$	500	35805	0	12026	12169
CIFAR-10S	$32 \times 32 \times 3$	10	9000	36000	1500	10000
CIFAR-100S	$32 \times 32 \times 3$	100	9000	36000	1500	10000
CelebA-500S	$32 \times 32 \times 3$	500	7009	28796	3431	12169

within 10% of the spatial dimensions to augment the training set. In all experiments, the final model weights, which are used to measure the performance on the test sets, are obtained from the epoch, which has the highest validation accuracy.

### C. Comparisons with MCL [26]

Table II shows the performances of MCL proposed by [26] and our proposed algorithm MCLwP. The last four rows of Table II present the learning performances of the corresponding prior-generating models. It is clear that with the presence of prior knowledge, our proposed algorithm outperforms MCL for most of the configurations in all five datasets.

For a simpler problem such as recognizing 10 objects in CIFAR-10 dataset, the performance gaps between MCLwP and MCL are relatively small. However, when the complexity of the learning problem increases, i.e., when the number of objects or facial identities increases, the differences in performance between MCLwP and MCL are significant. This can be observed by looking at the last row of each measurement configuration, with a direction from left to right (the direction of increasing difficulties of the learning tasks). For example, at configuration  $9 \times 7 \times 1$ , moving from CIFAR-10 to CIFAR-100, the improvement changes from 0.67 to 2.97, while from CelebA-100 to Celeb-200, then to CelebA-500, the improvement changes from 3.42 to 4.63 and to 7.20, respectively.

The only setting that MCLwP performs slightly worse than MCL is in CIFAR-10 dataset at the lowest measurement rate ( $6 \times 6 \times 1$ ). By inspecting the prior-generating models' performances in CIFAR-10 dataset, we can see that the prior-generating model at measurement  $6 \times 6 \times 1$  has very high performance, and even performs far better than its counterpart at a higher measurement rate ( $9 \times 7 \times 1$ ). Thus, the reason we see the degradation in learning performance might be because there is a huge gap between the teacher's and the student's learning capacity that makes the student model unable to learn effectively. This phenomenon in KD has been observed previously in [48].

One might make an assumption that lower numbers of measurements always associate with lower learning performances. This, however, is not necessarily true for our prior-generating

models having nonlinear CS component with different numbers of downsampling layers. In fact, in our prior-generating models, we use two  $2 \times 2$  max-pooling layers to reduce the spatial dimensions for configuration  $6 \times 6 \times 1$  while only one for configuration  $9 \times 7 \times 1$ . Although better performances for each measurement configuration can be achieved by carefully adjusting the corresponding teacher's capacity as in [48], it is sufficient for us to use a simple design pattern of autoencoder in order to demonstrate the effectiveness of the proposed MCLwP.

### D. Effects of Learning Capacity in Prior-Generating Models

While individual tweaking of each prior-generating model's topology requires elaborate experimentation and is out of the scope of this work, we still conducted a simple set of experiments to study the overall effects when changing the prior-generating models' capacity. In particular, we increased the capacity of the teacher models (P) in Section IV-C by adding more convolution layers in the CS and FS components. The set of higher-capacity teachers are denoted as P\*, and the resulting student models are denoted as MCLwP\*.

Table III shows the learning performances of MCLwP in comparison with MCLwP\*, and P in comparison with P\*. While there are clear improvements in the learning performance of the teachers when we switch from P to P\*, we observe mixed behaviors in the corresponding student models. Here we should note that this phenomenon is expected since different measurement configurations in different datasets would require different adjustments (increase or decrease) of the teacher's capacity to ensure the most efficient knowledge distillation. As observed in [48], the distribution of a student model's performance with respect to different teacher models' capacity has a bell shape. Thus, Table III can act as a guideline whether to increase or decrease the capacity of the prior-generating model in each configuration to maximize the learning performance of its student.

For example, in CIFAR-10 and CIFAR-100 dataset at measurement  $6 \times 6 \times 1$ , increasing the teacher's capacity from P to P\* leads to further degradation in the student's performances; thus we should lower the capacity of P to move toward the bell curve's peak. On the other hand, in CIFAR-100 at  $14 \times 11 \times 2$  or in CelebA-200 at  $6 \times 6 \times 1$ , we should

TABLE II  
LEARNING PERFORMANCES OF MCL [26] AND MCLwP (OUR PROPOSED ALGORITHM) MEASURED ON TEST SET (ACCURACY IN %). THE LAST FOUR ROWS SHOW THE PERFORMANCES OF PRIOR-GENERATING MODEL P FOR REFERENCE

Measurements	Models	CIFAR-10	CIFAR-100	CelebA-100	CelebA-200	CelebA-500
$6 \times 6 \times 1$	MCL [26]	<b>58.43</b>	27.55	53.96	44.52	38.07
	MCLwP (our)	57.80	<b>31.17</b>	<b>55.31</b>	<b>47.61</b>	<b>42.51</b>
	$\Delta$ (MCLwP-MCL)	-0.63	+3.62	+1.35	+3.09	+4.44
$9 \times 7 \times 1$	MCL [26]	66.14	33.70	72.08	67.40	61.53
	MCLwP (our)	<b>66.81</b>	<b>36.60</b>	<b>75.50</b>	<b>72.02</b>	<b>68.73</b>
	$\Delta$ (MCLwP-MCL)	+0.67	+2.90	+3.42	+4.62	+7.20
$13 \times 12 \times 1$	MCL [26]	77.17	43.49	86.27	83.54	83.63
	MCLwP (our)	<b>77.58</b>	<b>47.92</b>	<b>87.35</b>	<b>86.39</b>	<b>85.97</b>
	$\Delta$ (MCLwP-MCL)	+0.41	+4.43	+1.08	+2.85	+2.34
$14 \times 11 \times 2$	MCL [26]	84.38	59.08	86.90	84.36	83.93
	MCLwP (our)	<b>85.84</b>	<b>59.83</b>	<b>88.17</b>	<b>86.99</b>	<b>87.29</b>
	$\Delta$ (MCLwP-MCL)	+1.46	+0.75	+1.27	+2.63	+3.36
$6 \times 6 \times 1$	Prior (P)	80.71	42.31	75.79	72.24	71.46
$9 \times 7 \times 1$	Prior (P)	74.77	44.08	81.04	78.70	75.23
$13 \times 12 \times 1$	Prior (P)	82.19	53.10	89.75	87.94	87.08
$14 \times 11 \times 2$	Prior (P)	87.62	61.83	90.83	90.47	91.30

further upgrade the capacity of  $P^*$  to possibly obtain better performing student models. As mentioned previously, this type of empirical hill climbing for each measurement configuration requires extensive experiments, however, might be necessary for certain practical applications.

#### E. Quality of Compressive Domain

Although both MCL and MCLwP optimize the compressive learning models in an end-to-end manner, and there is no explicit loss term that regulates the compressive domain, it is still intuitive to expect models with better learning performances to possess better compressed representation at the same measurement rate. In order to quantify the representation produced by the competing algorithms in the compressive domain, we performed K-Nearest-Neighbor classification using the compressed representation at  $6 \times 6 \times 1$  and  $9 \times 7 \times 1$  after training MCL and MCLwP. Table IV shows the learning performances with two different neighbor values ( $k = 5$  and  $k = 20$ ).

It is clear that MCLwP outperforms MCL in the majority of configurations. The performance gaps between MCLwP and MCL are more significant in facial image recognition tasks than in object recognition tasks. Here we should note that Euclidean distance was used to measure the similarity between data points, which might not be the optimal metric which can entirely reflect the semantic similarity and the quality of the compressive domain, especially when our compressive domains possess tensor forms that also encode spatial information. Besides, we cannot compare the performances of K-Nearest-Neighbor on a dataset across different measurements since measurements having larger spatial dimensions potentially retain more spatial variances, and Euclidean distance becomes less effective when measuring the similarity. For

example, we can observe a decrease in performance on the CIFAR-10 and CIFAR-100 datasets, although the number of measurements increases from  $6 \times 6 \times 1$  to  $9 \times 7 \times 1$ . For this reason, we did not study the performances of K-Nearest-Neighbor at higher measurements.

#### F. Effects of Prior Knowledge

In order to study the effects of exploiting prior knowledge, we conducted two sets of experiments. While MCL and MCLwP models share the same structures of the CS and task-specific neural network component, there are architectural differences in the FS components: in MCL, the FS component performs multilinear transformation while in MCLwP, the synthesis step consists of both convolution and multilinear operations. Thus, in the first set of experiments, we aim to remove the architectural differences between the MCL and MCLwP models in the FS component that can potentially affect the quantification of prior knowledge. This is done by training the architectures which are trained by MCLwP in a similar manner as proposed in [26], i.e., without prior knowledge: we first initialized the CS and FS components by minimizing the  $l_1$  reconstructed error via stochastic optimization, then trained the whole model with respect to the inference loss. This set of models are denoted as MCLw/oP and the comparisons between MCLwP and MCLw/oP are presented in Table V. Although we observe mixed behaviors at measurement  $6 \times 6 \times 1$ , it is obvious that the presence of prior knowledge leads to improved learning performances without any architectural differences.

In the second set of experiments, we studied the effect of different knowledge transfer stages in MCLwP by either skipping or performing a particular knowledge transfer stage. This setup leads to  $2^3 = 8$  different variants of the training

TABLE III

LEARNING PERFORMANCES OF OUR PROPOSED ALGORITHM TRAINED WITH TWO DIFFERENT PRIOR-GENERATING MODELS (P AND P\*). P\* DENOTES THE HIGHER-CAPACITY ONE. THE LAST EIGHT ROWS SHOW THE PERFORMANCES OF THE PRIOR-GENERATING MODELS FOR REFERENCE.

Measurements	Models	CIFAR-10	CIFAR-100	CelebA-100	CelebA-200	CelebA-500
$6 \times 6 \times 1$	MCLwP	<b>57.80</b>	<b>31.17</b>	55.31	47.61	<b>42.51</b>
	MCLwP*	54.94	30.67	<b>55.56</b>	<b>48.77</b>	41.75
$9 \times 7 \times 1$	MCLwP	<b>66.81</b>	<b>36.60</b>	75.50	<b>72.02</b>	<b>68.73</b>
	MCLwP*	66.72	36.07	75.77	71.78	68.49
$13 \times 12 \times 1$	MCLwP	77.58	<b>47.92</b>	87.35	<b>86.39</b>	85.97
	MCLwP*	<b>78.27</b>	47.75	<b>87.50</b>	85.99	<b>86.10</b>
$14 \times 11 \times 2$	MCLwP	<b>85.84</b>	59.83	<b>88.17</b>	86.99	87.29
	MCLwP*	85.78	<b>60.83</b>	87.87	<b>87.09</b>	<b>87.41</b>
$6 \times 6 \times 1$	Prior P	80.71	42.31	75.79	72.24	71.46
	Prior P*	<b>84.29</b>	<b>49.47</b>	<b>79.75</b>	<b>76.48</b>	<b>76.74</b>
$9 \times 7 \times 1$	Prior P	74.77	44.08	81.04	78.70	75.23
	Prior P*	<b>82.91</b>	<b>48.58</b>	<b>84.25</b>	<b>82.49</b>	<b>81.27</b>
$13 \times 12 \times 1$	Prior P	82.19	53.10	89.75	87.94	87.08
	Prior P*	<b>84.53</b>	<b>57.23</b>	<b>91.04</b>	<b>89.22</b>	<b>89.27</b>
$14 \times 11 \times 2$	Prior P	87.62	61.83	90.83	90.47	91.30
	Prior P*	<b>87.93</b>	<b>63.69</b>	<b>91.42</b>	<b>90.53</b>	<b>92.11</b>

TABLE IV

K-NEAREST NEIGHBOR PERFORMANCES (TEST ACCURACY IN %) CALCULATED USING COMPRESSED MEASUREMENTS PRODUCED BY MCL [26] AND MCLwP (OUR PROPOSED ALGORITHM)

Measurements	Models	CIFAR-10	CIFAR-100	CelebA-100	CelebA-200	CelebA-500
<i>#neighbors k=5</i>						
$6 \times 6 \times 1$	MCL [26]	<b>38.21</b>	14.60	25.15	19.68	13.75
	MCLwP (our)	37.70	<b>15.02</b>	<b>31.92</b>	<b>22.81</b>	<b>15.45</b>
$9 \times 7 \times 1$	MCL [26]	<b>37.50</b>	<b>13.79</b>	30.15	25.02	18.37
	MCLwP (our)	35.52	12.56	<b>36.77</b>	<b>29.69</b>	<b>21.41</b>
<i>#neighbors k=20</i>						
$6 \times 6 \times 1$	MCL [26]	38.87	14.73	24.21	19.95	14.94
	MCLwP (our)	<b>39.36</b>	<b>15.32</b>	<b>31.48</b>	<b>23.46</b>	<b>17.08</b>
$9 \times 7 \times 1$	MCL [26]	<b>37.16</b>	13.01	30.21	25.31	19.59
	MCLwP (our)	34.93	<b>13.16</b>	<b>35.60</b>	<b>29.42</b>	<b>22.50</b>

TABLE V

LEARNING PERFORMANCES WITH (MCLwP) AND WITHOUT (MCLw/oP) THE PRESENCE OF PRIOR KNOWLEDGE

Measurements	Models	CIFAR-10	CIFAR-100	CelebA-100	CelebA-200	CelebA-500
$6 \times 6 \times 1$	MCLw/oP	<b>58.33</b>	30.27	<b>55.88</b>	<b>48.49</b>	42.40
	MCLwP	57.80	<b>31.17</b>	55.31	47.61	<b>42.51</b>
$9 \times 7 \times 1$	MCLw/oP	66.55	34.02	73.83	69.02	67.24
	MCLwP	<b>66.81</b>	<b>36.60</b>	<b>75.50</b>	<b>72.02</b>	<b>68.73</b>
$13 \times 12 \times 1$	MCLw/oP	77.49	44.80	86.71	83.12	83.37
	MCLwP	<b>77.58</b>	<b>47.92</b>	<b>87.35</b>	<b>86.39</b>	<b>85.77</b>
$14 \times 11 \times 2$	MCLw/oP	84.42	58.09	85.21	84.76	84.00
	MCLwP	<b>85.84</b>	<b>59.83</b>	<b>88.17</b>	<b>86.99</b>	<b>87.09</b>

procedure, which are presented in Table VI. Here we should note that when skipping the last transfer stage, we only discarded the distillation loss term and still trained the models with respect to the inference loss, instead of discarding the entire 3<sup>rd</sup> stage.

The first row of each dataset shows the results when all knowledge transfer activities are skipped. In other words, the architectures are initialized with a standard neural network initialization technique [49] and trained only with the inference loss. It is clear that this training procedure produces the worst performing models.

The next three rows in each dataset represent the cases where we performed only one of the knowledge transfer stages. In this scenario, conducting only the 2<sup>nd</sup> stage has noticeably better results than the other two cases, indicating the importance of the 2<sup>nd</sup> knowledge transfer stage.

Regarding the cases when one of the stages is skipped, we can also observe a homogeneous phenomenon across different measurements and different datasets that skipping the 1<sup>st</sup> stage leads to the least degradation. In fact, with this training procedure, we can obtain performances relatively close to the original MCLwP. During the 2<sup>nd</sup> knowledge transfer stage, the FS component (D) is updated in conjunction with the CS component (E) to mimic the features  $D_P(E_P(\mathcal{J}))$  synthesized by the prior-generating model, which might imply an implicit knowledge transfer from  $E_P$  to E, thus might explain why skipping the 1<sup>st</sup> transfer stage only leads to minor degradations in performance. Overall, performing all three knowledge transfer stages yields the best results.

### G. Semi-supervised Learning

Finally, we present the empirical results in the semi-supervised setting. Here we should note that although the total amount of data is large, only a small fraction has labels for training. The competing algorithms include MCL [26] and MCLwP, both of which were trained with the labeled data only, and MCLwP-S, the semi-supervised extension of MCLwP which takes advantages of the unlabeled data in addition to the labeled set. We denote P and P-S the corresponding prior-generating models of MCLwP and MCLwP-S, respectively. The results are shown in Table VII.

It is clear that without sufficient data, the prior-generating models (P) cannot effectively transfer the knowledge to their student models (MCLwP) as can be seen by the inferior performances of MCLwP compared to MCL in half of the cases. However, when unlabeled data is utilized during the training process as proposed in our MCLwP-S algorithm, it is obvious that not only the prior-generating models (P-S) improve in performance but also their student models (MCLwP-S). The enhancement of the knowledge transfer process (via additional data) and prior-generating models (via self-labeling procedure) results in MCLwP-S having the best performances among the competing algorithms.

## V. CONCLUSIONS

In this work, we proposed a novel methodology to find and incorporate data-dependent prior knowledge into the training

process of MCL models. In addition to the traditional supervised learning setting, we also proposed a semi-supervised adaptation that enables our methodology to take advantage of unlabeled data that comes from similar distributions of the signal of interest. Although we limited our investigation to MCL framework, the proposals presented in this work are sufficiently generic to be applicable to any CL systems. With extensive sets of experiments, we demonstrated the effectiveness of our algorithms to train MCL models in comparison with the previously proposed algorithm and provided insights into different aspects of the proposed methodologies.

## VI. ACKNOWLEDGEMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR). This publication reflects the authors views only. The European Commission is not responsible for any use that may be made of the information it contains.

## REFERENCES

- [1] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling [a sensing/sampling paradigm that goes against the common knowledge in data acquisition]," *IEEE signal processing magazine*, vol. 25, no. 2, pp. 21–30, 2008.
- [2] C. F. Caiafa and A. Cichocki, "Multidimensional compressed sensing and their applications," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 3, no. 6, pp. 355–380, 2013.
- [3] Y. August, C. Vachman, Y. Rivenson, and A. Stern, "Compressive hyperspectral imaging by random separable projections in both the spatial and the spectral domains," *Applied optics*, vol. 52, no. 10, pp. D46–D54, 2013.
- [4] V. M. Patel, G. R. Easley, D. M. Healy Jr, and R. Chellappa, "Compressed synthetic aperture radar," *IEEE Journal of selected topics in signal processing*, vol. 4, no. 2, pp. 244–254, 2010.
- [5] P. Irarrazabal and D. G. Nishimura, "Fast three dimensional magnetic resonance imaging," *Magnetic resonance in medicine*, vol. 33, no. 5, pp. 656–662, 1995.
- [6] E. J. Candès, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 59, no. 8, pp. 1207–1223, 2006.
- [7] D. L. Donoho et al., "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.
- [8] J. F. Mota, N. Deligiannis, and M. R. Rodrigues, "Compressed sensing with prior information: Strategies, geometry, and bounds," *IEEE Transactions on Information Theory*, vol. 63, no. 7, pp. 4472–4496, 2017.
- [9] V. Stanković, L. Stanković, and S. Cheng, "Compressive image sampling with side information," in *2009 16th IEEE International Conference on Image Processing (ICIP)*, pp. 3037–3040, IEEE, 2009.
- [10] L.-W. Kang and C.-S. Lu, "Distributed compressive video sensing," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1169–1172, IEEE, 2009.
- [11] A. Charles, M. S. Asif, J. Romberg, and C. Rozell, "Sparsity penalties in dynamical system estimation," in *2011 45th annual conference on information sciences and systems*, pp. 1–6, IEEE, 2011.
- [12] D. Baron, M. B. Wakin, M. F. Duarte, S. Sarvotham, and R. G. Baraniuk, "Distributed compressed sensing," 2005.
- [13] V. Cevher, A. Sankaranarayanan, M. F. Duarte, D. Reddy, R. G. Baraniuk, and R. Chellappa, "Compressive sensing for background subtraction," in *European Conference on Computer Vision*, pp. 155–168, Springer, 2008.
- [14] M. Trocan, T. Maugey, J. E. Fowler, and B. Pesquet-Popescu, "Disparity-compensated compressed-sensing reconstruction for multiview images," in *2010 IEEE International Conference on Multimedia and Expo*, pp. 1225–1229, IEEE, 2010.

TABLE VI  
LEARNING PERFORMANCES (TEST ACCURACY IN %) OF OUR PROPOSED ALGORITHM (MCLWP) WHEN DIFFERENT KNOWLEDGE TRANSFER STAGES ARE SKIPPED. THE CHECKMARK INDICATES WHEN A PARTICULAR TRANSFER STAGE WAS CONDUCTED

Transfer $E_P$	Transfer $D_P(E_P)$	Transfer $N_P(D_P(E_P))$	$6 \times 6 \times 1$	$9 \times 7 \times 1$	$13 \times 12 \times 1$	$14 \times 11 \times 2$
CIFAR-10						
			52.74	56.16	57.36	66.02
✓			53.99	57.68	61.63	66.82
	✓		57.91	65.97	77.12	84.86
		✓	54.38	59.31	61.23	67.74
✓	✓		57.17	65.97	77.16	84.80
	✓	✓	57.75	66.70	<b>77.89</b>	85.10
✓		✓	54.77	59.04	63.07	68.40
✓	✓	✓	<b>57.80</b>	<b>66.81</b>	77.58	<b>85.84</b>
CIFAR-100						
			25.13	27.00	28.81	37.77
✓			26.39	28.78	30.03	40.84
	✓		29.52	35.31	46.36	58.96
		✓	27.81	31.16	30.76	41.04
✓	✓		29.67	35.45	46.42	58.90
	✓	✓	31.12	36.41	47.73	<b>59.89</b>
✓		✓	28.14	31.84	31.87	43.43
✓	✓	✓	<b>31.17</b>	<b>36.60</b>	<b>47.92</b>	59.83
CelebA-100						
			33.58	47.19	55.75	57.13
✓			47.75	59.04	65.08	65.10
	✓		53.88	74.77	86.35	87.21
		✓	46.42	48.88	52.71	58.77
✓	✓		54.40	75.17	86.96	87.94
	✓	✓	53.96	74.06	86.60	87.50
✓		✓	48.88	33.19	65.67	65.77
✓	✓	✓	<b>55.31</b>	<b>75.50</b>	<b>87.35</b>	<b>88.17</b>
CelebA-200						
			38.81	36.21	46.73	53.96
✓			42.23	56.17	51.24	61.00
	✓		46.68	70.69	85.31	86.41
		✓	42.61	49.27	54.28	56.62
✓	✓		46.73	71.50	86.06	86.40
	✓	✓	47.53	71.52	85.86	86.42
✓		✓	43.59	58.72	61.88	63.37
✓	✓	✓	<b>47.61</b>	<b>72.02</b>	<b>86.39</b>	<b>86.99</b>
CelebA-500						
			33.60	46.46	54.40	53.03
✓			37.46	51.31	61.53	61.48
	✓		40.68	67.70	85.31	86.46
		✓	37.01	53.06	57.45	55.92
✓	✓		41.61	67.81	85.47	86.60
	✓	✓	41.89	<b>68.78</b>	85.95	87.27
✓		✓	39.11	56.91	64.83	63.22
✓	✓	✓	<b>42.51</b>	68.73	<b>85.97</b>	<b>87.29</b>

TABLE VII

LEARNING PERFORMANCES IN SEMI-SUPERVISED SETTING OF MCL [26], MCLwP (TRAINED WITH LABELED DATA ONLY), AND MCLwP-S (TRAINED WITH LABELED AND UNLABELED DATA). THE LAST EIGHT ROWS SHOW THE PERFORMANCES OF THE CORRESPONDING PRIOR-GENERATING MODELS (P AND P-S WHICH PROVIDE PRIOR KNOWLEDGE FOR MCLwP AND MCLwP-S, RESPECTIVELY.)

Measurements	Models	CIFAR-10S	CIFAR-100S	CelebA-500S
$6 \times 6 \times 1$	MCL	<b>51.19</b>	17.59	12.93
	MCLwP	47.78	18.35	17.70
	MCLwP-S	50.13	<b>19.63</b>	<b>21.91</b>
$9 \times 7 \times 1$	MCL	56.55	22.86	27.46
	MCLwP	55.75	20.05	25.72
	MCLwP-S	<b>57.59</b>	<b>23.03</b>	<b>34.16</b>
$13 \times 12 \times 1$	MCL	67.23	31.12	49.09
	MCLwP	67.13	28.97	44.56
	MCLwP-S	<b>69.37</b>	<b>32.67</b>	<b>51.55</b>
$14 \times 11 \times 2$	MCL	75.31	41.10	41.47
	MCLwP	75.86	41.55	42.07
	MCLwP-S	<b>78.03</b>	<b>43.87</b>	<b>58.34</b>
$6 \times 6 \times 1$	Prior P	<b>66.81</b>	25.88	27.32
	Prior P-S	63.96	<b>27.00</b>	<b>32.79</b>
$9 \times 7 \times 1$	Prior P	62.29	27.46	26.68
	Prior P-S	<b>64.44</b>	<b>29.11</b>	<b>38.76</b>
$13 \times 12 \times 1$	Prior P	70.95	33.97	47.69
	Prior P-S	<b>73.28</b>	<b>37.11</b>	<b>54.37</b>
$14 \times 11 \times 2$	Prior P	76.99	43.05	45.97
	Prior P-S	<b>79.70</b>	<b>46.85</b>	<b>63.83</b>

- [15] N. Vaswani and W. Lu, "Modified-cs: Modifying compressive sensing for problems with partially known support," *IEEE Transactions on Signal Processing*, vol. 58, no. 9, pp. 4595–4607, 2010.
- [16] F. Renna, R. Calderbank, L. Carin, and M. R. Rodrigues, "Reconstruction of signals drawn from a gaussian mixture via noisy compressive measurements," *IEEE Transactions on Signal Processing*, vol. 62, no. 9, pp. 2265–2277, 2014.
- [17] M. A. Khajehnejad, W. Xu, A. S. Avestimehr, and B. Hassibi, "Analyzing weighted  $\ell_1$  minimization for sparse recovery with nonuniform sparse models," *IEEE Transactions on Signal Processing*, vol. 59, pp. 1985–2001, May 2011.
- [18] M. Stojnic, F. Parvaresh, and B. Hassibi, "On the reconstruction of block-sparse signals with an optimal number of measurements," *IEEE Transactions on Signal Processing*, vol. 57, no. 8, pp. 3075–3085, 2009.
- [19] Y. C. Eldar and M. Mishali, "Robust recovery of signals from a structured union of subspaces," *IEEE Transactions on Information Theory*, vol. 55, no. 11, pp. 5302–5316, 2009.
- [20] R. Calderbank and S. Jafarpour, "Finding needles in compressed haystacks," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3441–3444, IEEE, 2012.
- [21] M. A. Davenport, M. F. Duarte, M. B. Wakin, J. N. Laska, D. Takhar, K. F. Kelly, and R. G. Baraniuk, "The smashed filter for compressive classification and target recognition," in *Computational Imaging V*, vol. 6498, p. 64980H, International Society for Optics and Photonics, 2007.
- [22] M. A. Davenport, P. Boufounos, M. B. Wakin, R. G. Baraniuk, *et al.*, "Signal processing with compressive measurements," *J. Sel. Topics Signal Processing*, vol. 4, no. 2, pp. 445–460, 2010.
- [23] H. Reboredo, F. Renna, R. Calderbank, and M. R. Rodrigues, "Compressive classification," in *2013 IEEE International Symposium on Information Theory*, pp. 674–678, IEEE, 2013.
- [24] S. Lohit, K. Kulkarni, P. Turaga, J. Wang, and A. C. Sankaranarayanan, "Reconstruction-free inference on compressive measurements," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 16–24, 2015.
- [25] A. Adler, M. Elad, and M. Zibulevsky, "Compressed learning: A deep neural network approach," *arXiv preprint arXiv:1610.09615*, 2016.
- [26] D. T. Tran, M. Yamac, A. Degerli, M. Gabbouj, and A. Iosifidis, "Multilinear compressive learning," *arXiv preprint arXiv:1905.07481*, 2019.
- [27] M. Yamac, M. Ahishali, N. Passalis, J. Raitoharju, B. Sankur, and M. Gabbouj, "Reversible privacy preservation using multi-level encryption and compressive sensing," *arXiv preprint arXiv:1906.08713*, 2019.
- [28] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 19–38, IEEE, 2017.
- [29] H. Reboredo, F. Renna, R. Calderbank, and M. R. Rodrigues, "Projections designs for compressive classification," in *2013 IEEE Global Conference on Signal and Information Processing*, pp. 1029–1032, IEEE, 2013.
- [30] P. K. Baheti and M. A. Neifeld, "Adaptive feature-specific imaging: a face recognition example," *Applied optics*, vol. 47, no. 10, pp. B21–B31, 2008.
- [31] S. Lohit, K. Kulkarni, and P. Turaga, "Direct inference on compressive measurements using convolutional neural networks," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 1913–1917, IEEE, 2016.
- [32] B. Hollis, S. Patterson, and J. Trinkle, "Compressed learning for tactile object recognition," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1616–1623, 2018.
- [33] A. Değerli, S. Aslan, M. Yamac, B. Sankur, and M. Gabbouj, "Compressively sensed image recognition," in *2018 7th European Workshop on Visual Information Processing (EUVIP)*, pp. 1–6, IEEE, 2018.

- [34] E. Zisselman, A. Adler, and M. Elad, "Compressed learning for image classification: A deep neural network approach," *Processing, Analyzing and Learning of Images, Shapes, and Forms*, vol. 19, p. 1, 2018.
- [35] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.
- [36] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.
- [37] R. G. Baraniuk and M. B. Wakin, "Random projections of smooth manifolds," *Foundations of computational mathematics*, vol. 9, no. 1, pp. 51–77, 2009.
- [38] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*, pp. 630–645, Springer, 2016.
- [39] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, "Deep mutual learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4320–4328, 2018.
- [40] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *arXiv preprint arXiv:1412.6550*, 2014.
- [41] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4133–4141, 2017.
- [42] R. Yu, A. Li, V. I. Morariu, and L. S. Davis, "Visual relationship detection with internal and external linguistic knowledge distillation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1974–1982, 2017.
- [43] R. Anil, G. Pereyra, A. Passos, R. Ormandi, G. E. Dahl, and G. E. Hinton, "Large scale distributed neural network training through online distillation," *arXiv preprint arXiv:1804.03235*, 2018.
- [44] I. Triguero, S. García, and F. Herrera, "Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study," *Knowledge and Information systems*, vol. 42, no. 2, pp. 245–284, 2015.
- [45] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," tech. rep., Citeseer, 2009.
- [46] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.
- [47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
- [48] S.-I. Mirzadeh, M. Farajtabar, A. Li, and H. Ghasemzadeh, "Improved knowledge distillation via teacher assistant: Bridging the gap between student and teacher," *arXiv preprint arXiv:1902.03393*, 2019.
- [49] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

## **7.14 Attention-based Neural Bag-of-Feature Learning for Sequence Data**

The appended paper follows.

# Attention-based Neural Bag-of-Features Learning for Sequence Data

Dat Thanh Tran\*, Nikolaos Passalis<sup>†</sup>, Anastasios Tefas<sup>†</sup>, Moncef Gabbouj\*, Alexandros Iosifidis<sup>‡</sup>

\*Department of Computing Sciences, Tampere University, Tampere, Finland

<sup>†</sup>Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece

<sup>‡</sup>Department of Engineering, Aarhus University, Aarhus, Denmark

thanh.tran@tuni.fi, passalis@csd.auth.gr, tefas@aia.csd.auth.gr,

moncef.gabbouj@tuni.fi, alexandros.iosifidis@eng.au.dk

**Abstract**—In this paper, we propose 2D-Attention (2DA), a generic attention formulation for sequence data, which acts as a complementary computation block that can detect and focus on relevant sources of information for the given learning objective. The proposed attention module is incorporated into the recently proposed Neural Bag of Feature (NBoF) model to enhance its learning capacity. Since 2DA acts as a plug-in layer, injecting it into different computation stages of the NBoF model results in different 2DA-NBoF architectures, each of which possesses a unique interpretation. We conducted extensive experiments in financial forecasting, audio analysis as well as medical diagnosis problems to benchmark the proposed formulations in comparison with existing methods, including the widely used Gated Recurrent Units. Our empirical analysis shows that the proposed attention formulations can not only improve performances of NBoF models but also make them resilient to noisy data.

## I. INTRODUCTION

Learning problems in many fields involve sequence data such as time-series forecasting [1], [2], audio analysis [3], [4] or natural language processing [5], [6], all of which have been extensively studied. In many application scenarios, the observed sequence is highly non-stationary and noisy, which makes the task of modeling the underlying generating process more difficult. For example, in sound source separation in which the objective is to recover different unknown sources by filtering the observed mixtures, the existence of environmental noise is inherent and often complicates the separation process. Several mathematical techniques have been proposed to model the underlying data and noise distributions or to extract hand-crafted features, capturing certain desirable properties. In financial time-series analysis, representative examples include autoregressive (AR) and moving average (MA) [7] features, which were later extended with a differencing step to eliminate nonstationarity, known as autoregressive integrated moving average (ARIMA) [8]. Gaussian processes and Hidden Markov Model were popular mathematical frameworks in audio analysis. To ensure mathematical and computational tractability, these classical models are often formulated under many assumptions, which are sensitive to initialization and misaligned with real-world conditions, thus limiting their professional usage in practice.

During the last decade, thanks to the development in stochastic optimization techniques and computing hardware, as well as the declining costs of data acquisition and storage, a data-driven approach based on deep neural networks and stochastic optimization has replaced the classical model-based approach and convex optimization. Nowadays, many of the state-of-the-art solutions for learning with sequence data are developed on the basis of neural networks. Notably, a class of neural network architecture called Recurrent Neural Networks (RNN), which is specifically designed to process variable-length sequences and to capture sequential patterns, has become the main workforce in different application domains. Another dedicated neural formulation for sequence data is the bilinear structures [1], [9], [10], which were proposed to separately capture the dependencies along the temporal and spatial dimension in financial time-series. Even existing neural architectures, which were originally proposed for visual inputs such as Convolutional Neural Network (CNN) [11] and Neural Bag-of-Features (NBoF) [12], have shown competitive performances in tackling sequence data compared to dedicated statistical models [13], [14]. The advantage of neural formulations over statistical learning and traditional hand-crafted features lies in the fact that fewer assumptions are made, and data is leveraged to automatically identify and extract task-relevant features in an end-to-end fashion.

Bag-of-Features (BoF) model [15] was originally proposed to build histogram representations from images. Later, it was shown that BoF could be successfully applied to extract high-level representations for other data modalities such as video and audio [16], [17], [18], [19]. Learning BoF representations consists of two steps: *dictionary learning* and *feature quantization and encoding*. In the dictionary learning step, each object is first represented by a set of low-level features, which could be, for example, a collection of local descriptors like SIFT [20] for image object or word-level vector-encodings for a sentence object. These features are then used to generate a compact dictionary (codebook) comprising of the most representative features, also known as *codewords*. In the second step, the histogram representation of each object is extracted by quantizing its low-level features using the codebook.

Recently, Neural Bag-of-Features (NBoF) [12], a neural network generalization of the BoF model, has been proposed. Similar to its predecessor, NBoF can generate a fixed-size histogram vector from variable-size inputs. This neural network generalization works as a feature extraction layer, which can be combined and optimized jointly with other neural network layers to tackle both unsupervised and supervised objectives via stochastic optimization. Since the dictionary learning step in NBoF is updated in conjunction with other layers towards the end goal of optimizing an objective function, histogram vectors synthesized by NBoF are more representative than those produced by BoF in different learning scenarios such as visual recognition, information retrieval, and financial forecasting [21], [12], [14].

While the NBoF model works well in different learning problems, the current formulations still possess some limitations. In the aggregation step, all of the quantized features are simply averaged to form the histogram vector. For sequence data, this implies that the model only allows equal contributions of the quantized features coming from different time steps to form the output representation. Similarly, the quantization results produced by each codeword are considered equally important for every sequence in the training set. These properties limit the dictionary learning, quantization, and encoding process to fully take advantage of the data-driven approach.

To incorporate a higher degree of flexibility into the NBoF model, a weighing mechanism on a sequence level is desirable. That is, for each individual sequence, the model has the flexibility to perform a weighted sum of quantized outputs in the aggregation step, with the coefficients being adaptively changed with respect to the input sequence, or to select/discard irrelevant codewords, given the input sequence. In neural network literature, this is often achieved by having some attention mechanisms [22], [23], [10]. The idea of attention is inspired by the phenomenon observed in the human visual cortex that visual stimuli from multiple objects actively compete for neural encoding.

Although various attention mechanisms have been proposed for existing neural network architectures such as CNN [22], [24], LSTM [23], [25] or Bilinear structure [10], there is yet any formulation for the NBoF model when learning with sequence data. To have a generic attention mechanism that can be applied in a plug-and-play manner, in this work, we propose 2D-Attention (2DA), a neural network module that promotes competitions among different rows or columns in the input matrix and only (soft) selects those which win for attention. We will then demonstrate that by injecting 2DA into NBoF, we can overcome those limitations mentioned previously. The contributions of our work can be summarized as follows:

- We propose a new type of attention formulation for matrix data, which is dubbed as 2DA. The proposed layer acts as a complementary computation block, which is capable of identifying relevant sources of information to perform selective masking on the given input matrix.

- We incorporate 2DA into different stages of the Neural Bag-of-Features (NBoF) model, creating various 2DA-NBoF extensions that can enhance the feature quantization or histogram accumulation step in the NBoF model. Extensive experiments were conducted in three different application domains: financial forecasting, audio analysis, and medical diagnosis, which demonstrate the effectiveness of our attention module in improving the NBoF model. In cases of noisy input, a variant of 2DA-NBoF shows resilience to noises by filtering out the noisy source of information before the feature quantization step.

The remainder of the paper is organized as follows: in Section II, we review the NBoF model and its extensions for time-series data, as well as previously proposed attention mechanisms in the neural network literature. In Section III, we first present the proposed attention module 2DA and its interpretation. Several extensions of the NBoF model that incorporates 2DA are then presented. In Section IV, we provide details of our experiment protocols and quantitative analysis. Section V concludes our work with possible future research directions.

## II. RELATED WORK

The NBoF model [12] consists of two components: a quantization layer and an accumulation layer. Each quantization neuron in the quantization layer performs like a codeword, which can be updated via BackPropagation algorithm. In the original formulation [12], the Radial Basis Function (RBF) layer was used for feature quantization. Recently, it has been shown that the hyperbolic kernel is also effective for the feature quantization step [26]. Here we describe the original formulation with RBF layer.

Let  $K$  be the number of neurons (codewords) in the RBF layer and  $\mathbf{v}_k \in \mathbb{R}^D$  be the  $k$ -th codeword. In addition, the shape of the Gaussian function modeled by each neuron can be adjusted via parameter  $\mathbf{w}_k$ . Let us denote the sequence of  $N$  features as  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$  with  $\mathbf{x}_n \in \mathbb{R}^D, n = 1, \dots, N$ . The output of the  $k$ -th RBF neuron given the input feature  $\mathbf{x}_n$  is the following:

$$\phi_{n,k} = \frac{\exp(-\|(\mathbf{x}_n - \mathbf{v}_k) \odot \mathbf{w}_k\|_2)}{\sum_{m=1}^K \exp(-\|(\mathbf{x}_n - \mathbf{v}_m) \odot \mathbf{w}_m\|_2)} \quad (1)$$

where  $\odot$  denotes element-wise product and  $\mathbf{w}_k \in \mathbb{R}^D$  is the learnable weight vector that enables the shape of Gaussian kernel associated with the  $k$ -th RBF neuron to change.

As the sequence  $\mathbf{X}$  goes through the quantization layer, each feature  $\mathbf{x}_n$  is quantized as  $\phi_n = [\phi_{n,1}, \dots, \phi_{n,K}]^T \in \mathbb{R}^K$ , producing a sequence of quantized features  $\Phi = [\phi_1, \dots, \phi_N] \in \mathbb{R}^{K \times N}$ . The accumulation layer aggregates the information in  $\Phi$  by calculating the averaged quantized feature:

$$\mathbf{y} = \frac{1}{N} \sum_{n=1}^N \phi_n \quad (2)$$

There have been few extensions of NBoF model for sequence data. For example, Temporal Neural Bag-of-Features

(TNBoF) model with different specialized codebooks has been proposed in [27] to capture both short-term and long-term temporal information in financial time-series. In [26], the authors derived the logistic formulation of the NBoF model using the hyperbolic kernel instead of the RBF kernel for the quantization step and proposed an adaptive scaling mechanism which showed significant improvements in training stability and performance of the NBoF networks.

While the attention mechanism was biologically inspired from the perspective of visual processing, this technique has also inspired and advanced several works in sequence data analysis, notably in sequence-to-sequence learning tasks. The first attention formulation applied to sequence data was proposed in [6] for tackling machine translation tasks. In this formulation, the authors proposed to construct the context vectors in Sequence-to-sequence Recurrent Neural Network model by selectively combining some hidden states, rather than using the last hidden state as the context vector. The selection coefficients, also known as attention weights, are computed adaptively based on the given input sequence, and updated jointly with other parameters during stochastic optimization.

The successful application of attention mechanism in machine translation tasks has led to the emergence of other attention formulations, which are designed to capture different types of salient information in sequence data. For example, in [28], the authors proposed a formulation that can detect pseudo-periods in certain types of time-series, such as energy consumption or meteorology data. To predict the future stock index, a dual-stage attention mechanism was proposed in [25] for RNN to actively select relevant exogenous series and temporal instances. Similarly, to highlight and focus on important temporal events in Limit Order Book, the authors in [10] proposed a method to calculate attention masks for bilinear networks. Although an attention formulation has been proposed for the convolutional NBoF model in [24] to estimate the true color of images capturing by different devices, this formulation only works with image data. To the best of our knowledge, there has been no attention formulation for the NBoF model to tackle sequence data.

### III. PROPOSED METHODS

In this Section, we will first present 2D-Attention (2DA), our proposed attention calculation for matrix data. Then, we will show how 2DA can be used to address different limitations of the NBoF model as described in Section I. Throughout the paper, we denote scalar values by either lower-case or upper-case characters ( $a, b, A, B, \dots$ ), vectors by lower-case bold-face characters ( $\mathbf{x}, \mathbf{y}, \dots$ ), matrices by upper-case bold-face characters ( $\mathbf{X}, \mathbf{Y}, \dots$ ), and mathematical functions by calligraphy characters  $\mathcal{F}, \mathcal{G}, \dots$ . In addition, we use  $x_{mn}$  to denote the element at position  $(m, n)$  in a matrix  $\mathbf{X}$ .

#### A. 2D-Attention

A matrix  $\mathbf{S} \in \mathbb{R}^{M \times N}$  is a second-order tensor which has two modes, with  $M$  and  $N$  are the dimensions of the

first and second mode, respectively. The matrix representation provides a natural way to represent a signal with two different sources of information. For example, a multivariate time-series is represented by a matrix with one mode representing the temporal dimension, and the other mode represents different sources that generate individual series.

The general idea of attention mechanism is to highlight important elements in the data while discarding irrelevant ones. For data represented as a matrix  $\mathbf{S}$ , rather than considering each element in  $\mathbf{S}$  individually, we would like to actively select certain columns or rows of  $\mathbf{S}$  while discarding the others. This is because columns or rows of  $\mathbf{S}$  usually form coherent sub-groups of the data. For example, discarding some temporal events or some individual series in a multivariate series corresponds to removing some rows or columns, depending on the orientation of the matrix.

To adaptively determine and focus on different columns or rows of a matrix, we propose 2D-Attention (2DA), with the functional form denoted by  $\mathcal{F}_{2DA}$ . This function takes a matrix  $\mathbf{S} \in \mathbb{R}^{M \times N}$  as the input, and returns  $\tilde{\mathbf{S}} \in \mathbb{R}^{M \times N}$  as the output. That is:

$$\tilde{\mathbf{S}} = \mathcal{F}_{2DA}(\mathbf{S}) \quad (3)$$

$\tilde{\mathbf{S}}$  can be considered as a filtered version of  $\mathbf{S}$ , where irrelevant columns of  $\mathbf{S}$  with respect to the learning problem are zeroed out. Here we should note that  $\mathcal{F}_{2DA}$  performs adaptive attention with respect to columns of  $\mathbf{S}$ . To focus on different rows of  $\mathbf{S}$ , we can simply apply  $\mathcal{F}_{2DA}$  to the transpose of  $\mathbf{S}$ .

The selection or rejection of the columns of  $\mathbf{S}$  is conducted via element-wise matrix multiplications as follows:

$$\tilde{\mathbf{S}} = \tau(\mathbf{S} \odot \mathbf{A}) + (1 - \tau)\mathbf{S} \quad (4)$$

where  $\mathbf{A} \in \mathbb{R}^{M \times N}$  denotes the attention mask with values in the range  $[0, 1]$ . Each column in  $\mathbf{A}$  encodes the importance of the corresponding column in  $\mathbf{S}$ . That is, the attention mask contains values that are close to 1 corresponding to those columns in  $\mathbf{S}$  that contain important information for the downstream learning task and vice versa. In Eq.(4), parameter  $\tau \in \mathbb{R}$ , which is jointly optimized with other parameters, is used to allow flexible control of the attention mechanism: when  $\mathbf{S}$  contains redundant or noisy information in its columns, the effect of attention mask  $\mathbf{A}$  is enabled by pushing  $\tau$  close to 1; on the other hand, when every column of  $\mathbf{S}$  is necessary, i.e., there is no need for attention, pushing  $\tau$  close to 0 will disable the effect of  $\mathbf{A}$ . The necessity of attention is thus automatically determined by optimizing  $\tau$  with respect to a given problem.

To calculate the attention mask  $\mathbf{A}$ , the proposed 2DA method learns to measure the relative importance between columns of  $\mathbf{S}$  via a specially designed weight matrix  $\mathbf{W} \in \mathbb{R}^{N \times N}$ : all elements of  $\mathbf{W}$  are learnable, i.e., they are updated during stochastic optimization, except the diagonal elements, which are fixed to  $1/N$ . The attention mask is calculated as follows:

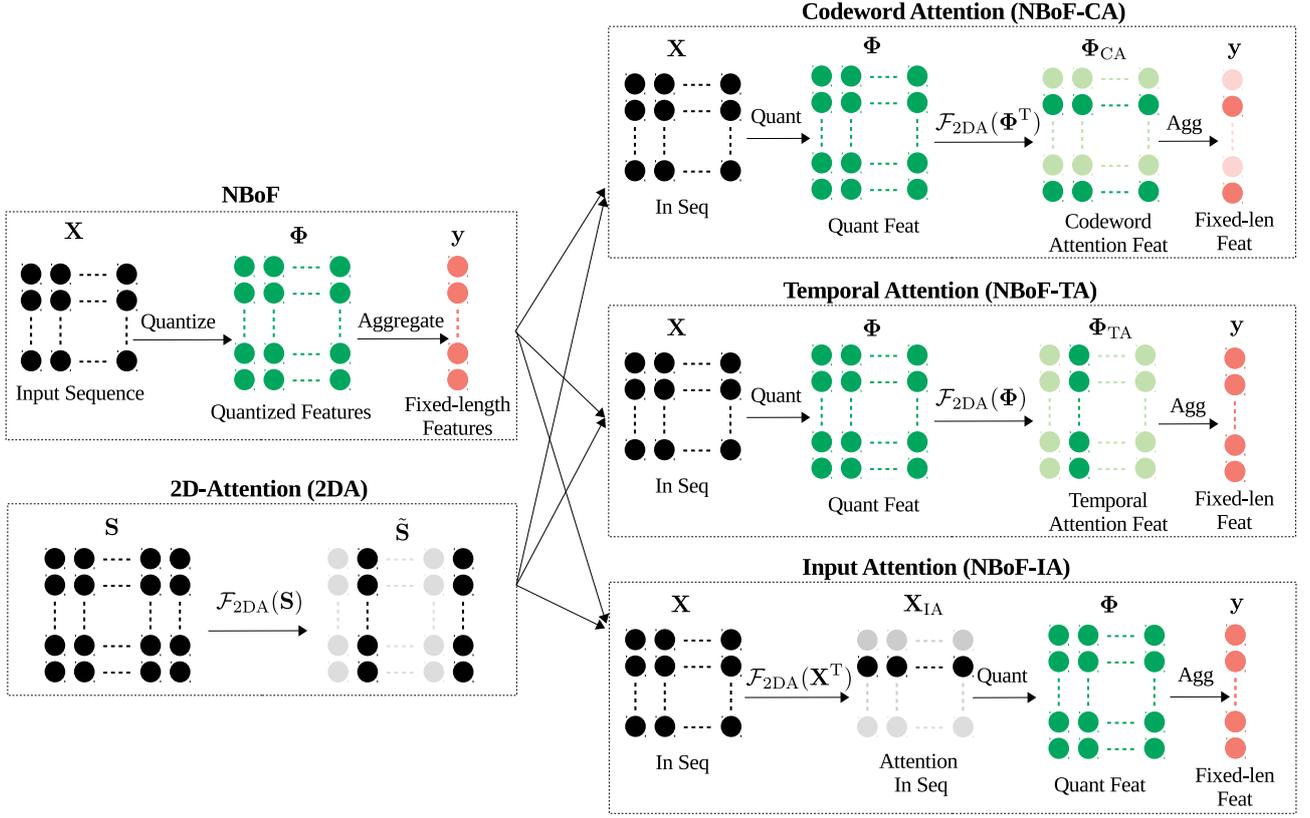


Fig. 1. Illustration of the proposed attention formulation (2DA) and different attention-based NBoF models

$$\begin{aligned} \mathbf{A} &= \mathcal{G}(\mathbf{Z}) \\ \mathbf{Z} &= \mathbf{S}\mathbf{W} \end{aligned} \quad (5)$$

where  $\mathcal{G}(\mathbf{Z})$  denotes the soft-max function that is applied to every row of  $\mathbf{Z}$ . That is, every element of  $\mathbf{A}$  is non-negative, and each row of  $\mathbf{A}$  sums up to 1. Similar to other attention formulations [25], [6], [22], we use soft-max normalization to promote competitions between different columns of  $\mathbf{Z}$ .

As mentioned previously, the weight matrix  $\mathbf{W}$  is used to measure the relative importance between columns of  $\mathbf{S}$ , which is encoded in  $\mathbf{Z}$ , and thus  $\mathbf{A}$ . In order to see this, let us denote by  $\mathbf{s}_n \in \mathbb{R}^M$  and  $\mathbf{z}_n \in \mathbb{R}^M$  the  $n$ -th column of  $\mathbf{S}$  and  $\mathbf{Z}$ , respectively. Since  $\mathbf{Z} = \mathbf{S}\mathbf{W}$ , the  $n$ -th column of  $\mathbf{Z}$ , i.e.,  $\mathbf{z}_n$ , is calculated as the weighted combination of  $N$  columns of  $\mathbf{S}$ , with the weight of the  $n$ -th column always equal to  $1/N$  since the diagonal elements of  $\mathbf{W}$  are fixed to  $1/N$ . In this way, element  $z_{mn}$  (in  $\mathbf{Z}$ ) encodes the relative importance of  $s_{mn}$  (in  $\mathbf{S}$ ) with respect to other  $s_{mk}$ , for  $k \neq n$ .

### B. Attention-based Neural Bag-of-Features

In this subsection, we will show how the proposed attention module 2DA can be used to address different limitations of the NBoF model described in Section I.

**Codeword Attention:** in the NBoF model, quantization results produced by each quantization neuron (codeword) are

considered equally important for every input sequence. This property limits the feature quantization step to fully take advantage of the data-driven approach. In order to overcome this limitation, the proposed 2DA block can be applied to the quantized features to highlight or discard the outputs of certain quantization neurons. By doing so, the NBoF model is explicitly encouraged to learn a subset of specialized codewords for a given input pattern.

Particularly, given the quantized features denoted by  $\Phi \in \mathbb{R}^{K \times N}$  as described in Section II, we propose to apply attention to the rows of  $\Phi$  because the first mode of  $\Phi$  with dimension  $K$  denotes the number of quantization neurons or codewords. Since 2DA operates on the columns of the input matrix, the attention-based quantized features is calculated as follows:

$$\Phi_{CA} = \mathcal{F}_{2DA}(\Phi^T) \quad (6)$$

where  $\Phi^T$  denotes the transpose of  $\Phi$ .

**Temporal Attention:** another limitation of the NBoF model lies in the aggregation step. In order to produce a fixed-length representation of the input sequence, the aggregation step in the NBoF model simply computes the mean of quantized features along the temporal mode. In this way, the NBoF model only allows equal contributions of all quantized features, disregarding the temporal information. In fact, the idea

of giving different weights to different time instances has been adopted in previous works under different formulations [10], [25]. Using our proposed 2DA formulation, it is straightforward to enable the NBoF model to attend to salient temporal information as follows:

$$\Phi_{\text{TA}} = \mathcal{F}_{2\text{DA}}(\Phi) \quad (7)$$

Since each column of  $\Phi$  contains quantized features of each time step, to obtain temporal attention-based features  $\Phi_{\text{TA}}$  we simply apply  $\mathcal{F}_{2\text{DA}}$  to  $\Phi$  as in Eq. (7).  $\Phi_{\text{TA}}$  is then averaged along the second dimension to produce the fixed-length representation of the input sequence. Although we still perform averaging in the aggregation step, the fixed-length representation is no longer the average of the quantized features, but a weighted average. This is because each time instance (column) in  $\Phi_{\text{TA}}$  has been scaled by different factors via the attention mechanism.

**Input Attention:** noisy data is an inherent problem in many real-world applications. Noises might surface during the data acquisition process, such as ambient noise in audio signals or power line interference and motion artifacts in Electrocardiogram signals. In other scenarios, noises are inherent in the problem formulation since the relevance between the input sources and the targets might be unclear. For example, in stock prediction, it is intuitive to use related stocks' data, e.g., those coming from the same market sector, as the input to construct forecasting models, although some of them might be irrelevant to the movement of the target stock.

The proposed attention mechanism can also be used to filter out potential noisy series in a multivariate series as follows:

$$\mathbf{X}_{\text{IA}} = \mathcal{F}_{2\text{DA}}(\mathbf{X}^{\text{T}}) \quad (8)$$

where  $\mathbf{X} \in \mathbb{R}^{D \times N}$  denotes an input sequence of  $N$  steps of the NBoF model as specified in Section II. Since we would like to apply attention over the individual series (rows of  $\mathbf{X}$ ),  $\mathcal{F}_{2\text{DA}}$  is applied to the transpose of  $\mathbf{X}$ .

The proposed attention variants of the NBoF model are illustrated in Figure 1.

#### IV. EXPERIMENTS

In this section, we provide detailed descriptions and results of our empirical analysis, which demonstrate the advantages of attention-based NBoF models proposed in Section III. Experiments were conducted in different types of sequence data, namely financial time-series in stock movement prediction problem, Electrocardiogram (ECG) and Phonocardiogram (PCG) in heart anomaly detection problems, and audio recording in music genre recognition and acoustic scene classification problems.

The experiments were conducted with the recently proposed logistic formulation of the NBoF model [26], i.e., the hyperbolic kernel was used in the quantization layer. In addition, we also experimented with the temporal variant of the NBoF model as proposed in [27] with a long-term and a short-term codebook. This variant is denoted as TNBoF. The codebook

attention, temporal attention, and input attention when applied to the NBoF model are denoted as NBoF-CA, NBoF-TA, and NBoF-IA, respectively. The corresponding attention variants for the TNBoF model are denoted as TNBoF-CA, TNBoF-TA, and TNBoF-IA. In addition to the NBoF and TNBoF models serving as the baseline models, we also evaluated RNN models using Gated Recurrent Units (GRU) [5].

##### A. Financial Forecasting Experiments

Although extensively studied over the last decades, financial forecasting still remains as the most challenging tasks among time-series predictions [29]. This is due to the complex dynamics of the financial markets, which make the observed data highly stationary and noisy. For this reason, we selected the stock movement prediction problem in FI2010 dataset [2] as a representative problem in time-series forecasting. FI2010 is the largest publicly available Limit Order Book (LOB) dataset, which contains approximately 4.5 million order events. The limit orders came from 5 Finnish stocks traded in Helsinki Exchange (operated by NASDAQ Nordic) over 10 business days. At each time instance, the dataset provides information (the prices and volumes) of the top 10 levels, leading to a 40-dimensional vector representation.

The FI2010 dataset is used to investigate the problem of mid-price movement prediction in the next  $H = \{10, 20, 50\}$  order events. The mid-price at a given time instance is the average between the best buy and best sell prices. This quantity is a virtual price since no trade can happen at this particular price at the given time instance. The movement of mid-price (stationary, increasing, decreasing) reflects the dynamic of the LOB and the market, thus plays an important role in financial analysis. The dataset provides the movement labels, given the future horizon  $H = \{10, 20, 50\}$ . Details regarding the FI2010 dataset and LOB can be found in [2].

We followed the same experimental setup proposed in [10], which used the first 7 days for training the models and the last 3 days to test the performances. Due to the imbalanced nature of the problem, we reported averaged F1 score per movement as the main performance metric, similar to prior experiments [1], [10]. Detailed information about the training hyper-parameters and the network architectures is provided in the Appendix.

The experiment results for FI2010 are shown in Table I. In the second column of Table I, we list the performances of all models without using any convolution layers as the preprocessing layers. That is, the results in the second column of Table I are produced by architectures consisting of only the layer of interests (such as GRU, NBoF, and so on), plus the fully connected layers for generating predictions. In this setting, the GRU models outperform all variants of the NBoF model. This is expected since the NBoF model, by construction, is not designed to capture local features and long-term dependency in the input sequence. We can easily observe that this limitation can be partially overcome with the TNBoF variant, which uses two separate codebooks to capture the short-term and long-term dependency. By applying

TABLE I

RESULTS ON THE FI2010 DATASET. THE SECOND COLUMN SHOWS PERFORMANCES (AVERAGED F1 IN %) OF ALL MODELS WITHOUT USING ANY CONVOLUTION LAYER AS PREPROCESSING LAYERS. THE THIRD COLUMN SHOWS THE CORRESPONDING PERFORMANCES (AVERAGE F1 IN %) WHEN ADDITIONAL CONVOLUTION LAYERS WERE USED AS PREPROCESSING LAYERS. THE BEST RESULTS IN EACH COLUMN ARE HIGHLIGHTED IN BOLD-FACE

Models	without conv	with conv
<i>Prediction Horizon <math>H = 10</math></i>		
GRU [5]	<b>60.92</b> $\pm$ 00.09	62.21 $\pm$ 00.30
NBoF [26]	33.07 $\pm$ 00.66	66.34 $\pm$ 00.60
NBoF-CA (our)	40.81 $\pm$ 00.05	67.56 $\pm$ 00.02
NBoF-TA (our)	40.83 $\pm$ 00.21	<b>67.98</b> $\pm$ 00.09
TNBoF [27]	36.66 $\pm$ 00.51	66.74 $\pm$ 00.36
TNBoF-CA (our)	45.61 $\pm$ 00.16	67.76 $\pm$ 00.05
TNBoF-TA (our)	45.97 $\pm$ 00.15	67.88 $\pm$ 00.13
<i>Prediction Horizon <math>H = 20</math></i>		
GRU [5]	<b>51.61</b> $\pm$ 00.25	53.83 $\pm$ 00.14
NBoF [26]	38.06 $\pm$ 00.53	58.85 $\pm$ 00.05
NBoF-CA (our)	40.08 $\pm$ 00.07	59.31 $\pm$ 00.44
NBoF-TA (our)	40.34 $\pm$ 00.06	<b>60.10</b> $\pm$ 00.03
TNBoF [27]	38.67 $\pm$ 00.50	59.61 $\pm$ 00.48
TNBoF-CA (our)	43.06 $\pm$ 00.03	59.73 $\pm$ 00.19
TNBoF-TA (our)	43.50 $\pm$ 00.15	60.04 $\pm$ 00.24
<i>Prediction Horizon <math>H = 50</math></i>		
GRU [5]	<b>63.13</b> $\pm$ 00.19	65.93 $\pm$ 00.03
NBoF [26]	48.25 $\pm$ 00.25	68.84 $\pm$ 02.29
NBoF-CA (our)	49.34 $\pm$ 00.17	73.25 $\pm$ 00.27
NBoF-TA (our)	49.21 $\pm$ 00.16	73.02 $\pm$ 00.04
TNBoF [27]	54.06 $\pm$ 00.14	69.27 $\pm$ 01.09
TNBoF-CA (our)	57.15 $\pm$ 00.21	<b>73.77</b> $\pm$ 00.37
TNBoF-TA (our)	57.41 $\pm$ 00.06	73.40 $\pm$ 00.08

our proposed attention mechanism, performances of both the NBoF and TNBoF models are further boosted.

The third column of Table I shows the performances of all models when using two additional convolution layers as the local feature extractor, prior to applying the layer of interest. It is clear that all of the models benefit from the additional convolution layers, especially the NBoF model and its variants. In this setting, the GRU models no longer dominate the family of NBoF models. In fact, the GRU models become the worst-performing ones in the third column of Table I. Furthermore, both codebook attention (NBoF-CA, TNBoF-CA) and temporal attention (NBoF-TA, TNBoF-TA) consistently enhance the baselines' performances, making attention-based models the best-performing ones.

Here we should note that although the baseline models (NBoF, TNBoF) use the adaptive scaling step proposed in [26] to improve training stability, we did not employ this step in attention-based models. The reason stems from the fact that adaptive scaling introduces additional degrees of freedom

TABLE II

PERFORMANCES (AVERAGED F1 IN %) OF ATTENTION-BASED MODELS ON THE FI2010 DATASET, WITH AND WITHOUT THE ADAPTIVE SCALING STEP PROPOSED IN [26]. THE BEST RESULTS IN EACH ROW ARE HIGHLIGHTED IN BOLD-FACE

Models	adaptive scale	no adaptive scale
<i>Prediction Horizon <math>H = 10</math></i>		
NBoF-CA	66.92 $\pm$ 00.08	<b>67.56</b> $\pm$ 00.02
NBoF-TA	67.34 $\pm$ 00.14	<b>67.98</b> $\pm$ 00.09
TNBoF-CA	<b>67.84</b> $\pm$ 00.16	67.76 $\pm$ 00.05
TNBoF-TA	67.16 $\pm$ 00.32	<b>67.88</b> $\pm$ 00.13
<i>Prediction Horizon <math>H = 20</math></i>		
NBoF-CA	59.25 $\pm$ 00.14	<b>59.31</b> $\pm$ 00.44
NBoF-TA	59.26 $\pm$ 00.18	<b>60.10</b> $\pm$ 00.03
TNBoF-CA	<b>59.75</b> $\pm$ 00.35	59.73 $\pm$ 00.19
TNBoF-TA	59.78 $\pm$ 00.11	<b>60.04</b> $\pm$ 00.24
<i>Prediction Horizon <math>H = 50</math></i>		
NBoF-CA	71.93 $\pm$ 00.14	<b>73.25</b> $\pm$ 00.27
NBoF-TA	47.89 $\pm$ 23.87	<b>73.02</b> $\pm$ 00.04
TNBoF-CA	71.30 $\pm$ 00.29	<b>73.77</b> $\pm$ 00.37
TNBoF-TA	72.32 $\pm$ 00.05	<b>73.40</b> $\pm$ 00.08

to the quantization step, which counteracts the constraining effects of the attention mechanism. Table II shows the performances of attention-based models on the FI2010 dataset, with and without the adaptive scaling step proposed in [26]. In most cases, the adaptive scaling step slightly degrades the performances of the attention-based models. As we will see in the next subsection, this effect is more noticeable in audio datasets.

### B. Audio Analysis Experiments

One of the important types of sequence data is audio recordings. In this subsection, we present our empirical analysis using two audio datasets, representing two different applications in audio signal analysis: music genre recognition and acoustic scene classification.

In the first application, the objective is to train an acoustic system that recognizes the genre of a short musical recording. For this purpose, we conducted experiments using the *small subset* of the FMA dataset [30], which contains 8000 tracks coming from the 8 most popular genres: *pop*, *instrumental*, *experimental*, *folk*, *rock*, *international*, *electronic*, and *hip-hop*. Each audio clip is 30s long, which is transformed to Mel-spectrogram representation with 128 frequency bands using a window of 10ms with an overlap of 2.5ms. The preprocessing step results in the input sequence having dimensions of  $128 \times 640$ .

In the second application, the objective is to train an acoustic system that can classify the type of environment based on its surrounding sounds. For this application, we used the TUT-UAS2018 dataset [31], which contains 8640

TABLE III

AUDIO ANALYSIS RESULTS OF FMA AND TUT-UAS2018 DATASETS. THE SECOND COLUMN SHOWS PERFORMANCES (TEST ACCURACY IN %) OF ALL MODELS WITHOUT USING ANY CONVOLUTION LAYER AS PREPROCESSING LAYERS. THE THIRD COLUMN SHOWS THE CORRESPONDING PERFORMANCES (TEST ACCURACY IN %) WHEN ADDITIONAL CONVOLUTION LAYERS WERE USED AS PREPROCESSING LAYERS. THE BEST RESULTS IN EACH COLUMN ARE HIGHLIGHTED IN BOLD-FACE.

Models	without conv	with conv
<b>FMA Dataset</b>		
GRU [5]	33.87±00.27	42.06±00.92
NBoF [26]	35.65±01.41	38.83±01.83
NBoF-CA (our)	<b>38.29</b> ±00.95	41.08±01.85
NBoF-TA (our)	36.79±00.29	41.46±01.64
TNBoF [27]	35.25±03.50	39.13±00.53
TNBoF-CA (our)	37.29±00.66	<b>42.67</b> ±01.23
TNBoF-TA (our)	36.50±01.49	42.58±00.91
<b>TUT-UAS2018 Dataset</b>		
GRU [5]	56.83±00.78	56.89±00.93
NBoF [26]	52.02±00.18	55.92±01.40
NBoF-CA (our)	<b>56.89</b> ±00.17	<b>57.68</b> ±00.65
NBoF-TA (our)	56.09±00.25	57.63±00.30
TNBoF [27]	52.62±00.78	55.30±00.13
TNBoF-CA (our)	56.19±00.23	56.73±00.51
TNBoF-TA (our)	56.34±00.62	57.33±00.20

audio clips recorded from 10 urban acoustic scenes: *airport*, *shopping\_mall*, *metro\_station*, *street\_pedestrian*, *public\_square*, *street\_traffic*, *tram*, *bus*, *metro*, *park*. Similar to the FMA dataset, we also transformed each audio clip to Mel-spectrogram with 128 frequency bands using a window of 40ms with an overlap of 20ms, which results in the input sequence of size  $128 \times 500$ .

For both applications, we report the test accuracy as the performance metric. Experiment results on FMA and TUT-UAS2018 dataset are shown in Table III. Performances of all models with and without using convolution layers for feature extraction are presented in the second and third columns, respectively.

In the FMA dataset, we can easily observe significant improvements in all models when using additional convolution layers. Without any convolution layer, the NBoF and TNBoF models outperform the GRU model on average, however, with larger variances. The order reverses when additional convolution layers were used: the GRU model enjoys a huge benefit from the preprocessing layers, outperforming the NBoF and TNBoF models. In both scenarios, i.e., with or without convolution layers, the proposed attention block greatly enhances the baseline NBoF and TNBoF models, making them the best performing models in this task.

In the TUT-UAS2018 dataset, while adding convolution layers leads to noticeable improvements for the baseline NBoF and TNBoF, we observe no similar improvement for the GRU

TABLE IV

PERFORMANCES (TEST ACCURACY IN %) OF ATTENTION-BASED MODELS ON FMA AND TUT-UAS2018 DATASETS, WITH AND WITHOUT THE ADAPTIVE SCALING STEP PROPOSED IN [26]. THE BEST RESULTS IN EACH ROW ARE HIGHLIGHTED IN BOLD-FACE.

Models	adaptive scale	no adaptive scale
<b>FMA Dataset</b>		
NBoF-CA	38.37±02.04	<b>41.08</b> ±01.85
NBoF-TA	34.29±05.42	<b>41.46</b> ±01.64
TNBoF-CA	39.96±00.66	<b>42.67</b> ±01.23
TNBoF-TA	40.21±00.16	<b>42.58</b> ±00.91
<b>TUT-UAS2018 Dataset</b>		
NBoF-CA	40.57±14.36	<b>57.68</b> ±00.65
NBoF-TA	56.62±00.39	<b>57.63</b> ±00.30
TNBoF-CA	56.59±00.51	<b>56.73</b> ±00.51
TNBoF-TA	55.05±01.25	<b>57.33</b> ±00.20

TABLE V

AUDIO ANALYSIS RESULTS UNDER NOISY DATA SETTING. NO PREPROCESSING CONVOLUTION LAYER WAS USED IN THIS SETTING.

Models	test accuracy
<b>Noisy FMA Dataset</b>	
GRU [5]	31.04±01.43
NBoF [26]	31.54±00.21
NBoF-IA (our)	36.21±01.93
TNBoF [27]	30.71±00.87
TNBoF-IA (our)	<b>36.67</b> ±00.95
<b>Noisy TUT-UAS2018 Dataset</b>	
GRU [5]	56.17±01.31
NBoF [26]	41.73±12.66
NBoF-IA (our)	<b>56.79</b> ±00.86
TNBoF [27]	51.48±00.85
TNBoF-IA (our)	56.04±00.42

model. Similar to the FMA dataset, we observe consistent performance boost in the TUT-UAS2018 dataset by incorporating the proposed attention block to the NBoF and TNBoF models.

Similar to Section IV-A, we also conducted experiments in FMA and TUT-UAS2018 datasets to analyze the effects of the adaptive scaling step proposed in [26]. The results are shown in Table IV. The results obtained from both audio analysis tasks in Table IV are consistent with what we observe from the stock movement prediction task in Table II: although the adaptive scaling step can enhance NBoF and TNBoF models as demonstrated in [26], the additional degrees of freedom introduced by this step negates the competition effects enforced by the attention mechanism, leading to performance degradation when combining both methods.

In order to evaluate how well the proposed input attention

TABLE VI  
PERFORMANCE (AVERAGED F1) ON AF DATASET

Models	F1
GRU [5]	76.42±00.86
NBoF [26]	78.15±00.82
NBoF-CA (our)	78.73±00.71
NBoF-TA (our)	78.55±00.90
TNBoF [27]	78.27±01.02
TNBoF-CA (our)	78.71±00.92
TNBoF-TA (our)	<b>79.52±00.81</b>

mechanism (NBoF-IA, TNBoF-IA) tackles noisy data, we simulated contaminated audio data by adding 10 synthetic frequency bands, which are generated by adding white noise to the averaged Mel coefficients. Here we should note that in this set of experiments, we did not use any convolution layers in order to gauge how well the layers of interests are resilient to noise. The results are shown in Table V. As can be seen from Table V, when moving from the noiseless to the noisy version of FMA and TUT-UAS2018 datasets, the accuracy of NBoF and TNBoF models dropped significantly. GRU models also exhibited similar behaviors, although the performance drops are less significant as compared to the NBoF and TNBoF. By incorporating the proposed input attention block to the NBoF and TNBoF models, we were able to achieve very similar performances compared to the noiseless scenario.

### C. Medical Diagnosis Experiments

Medical diagnosis, which plays a crucial role in ensuring human prosperity, is inherently an intricate process. The quality of the diagnosis is highly dependent on the expertise of the examiner. Since it takes several years and a great amount of resources to train human experts, medical diagnosis tools have been actively developed over the past decades to assist human examiners. In our empirical analysis using medical data, we investigated the effectiveness of the proposed models in diagnosing cardiovascular diseases using publicly available Electrocardiogram (ECG) and Phonocardiogram (PCG) signals.

The AF dataset focuses on the problem of atrial fibrillation detection from ECG recordings, which are provided as the development data (training set) in the Physionet/Computing in Cardiology Challenge 2017 [32]. The dataset contains 8528 single-lead ECG recordings lasting from 9 to 60 seconds. The objective of the challenge was to classify a given recording into one of the 4 classes: normal sinus rhythm, atrial fibrillation, alternative rhythm, and noise. We followed an experimental setup similar to [33], which evaluates a given model using 5-fold cross-validation. Additionally, the recordings were clipped or padded so that they have a constant length of 30 seconds. Since a single lead ECG recording is only a univariate sequence, it is necessary to use convolution layers as preprocessing layers to extract higher-level features,

TABLE VII  
PERFORMANCE (MEAN OF SENSITIVITY AND SPECIFICITY) ON PCG DATASET. THE HIGHER, THE BETTER.

Models	Anomaly Detection	Quality Detection
GRU [5]	<b>90.08±00.68</b>	<b>72.74±01.40</b>
NBoF [26]	50.31±00.42	49.69±00.34
NBoF-CA (our)	88.09±00.25	71.98±03.00
NBoF-TA (our)	89.32±01.02	72.57±02.20
TNBoF [27]	54.12±05.18	53.40±04.21
TNBoF-CA (our)	88.68±00.95	72.34±01.32
TNBoF-TA (our)	88.81±01.07	69.45±00.89

before the NBoF or GRU layers. To tackle the imbalanced nature of the training set, we scaled the loss term associated with each class, with the factor inversely proportional to the number of samples in that class.

PCG signal is often used in ambulatory diagnosis in order to evaluate the heart hemodynamic status and detect potential cardiovascular problems. The data used in our experiments come from the training set provided in the Physionet/Computing in Cardiology Challenge 2016 [34]. The objective of the challenge is to develop an automatic classification method for the anomaly (normal versus abnormal) and quality (good versus bad) detection given a PCG recording.

Since the length of the recordings varies greatly, from 5 to 120 seconds, we generated 5s segments from the recordings for training the models; during the test phase, the models were used to classify 5s sub-segments (with 4s overlap) of a given recording, and the overall label is inferred from the averaged classification of the sub-segments. PCG signal captures the acoustic nature of the heart sound; thus, we extracted Mel-spectrogram with 24 frequency bands, using a window of 25ms with an overlap of 10ms to represent each segment. With a smaller size compared to the AF dataset, we only employed a 3-fold cross-validation protocol for this problem. Further details regarding our experimental setup in AF and PCG datasets are provided in the Appendix.

Table VI shows the averaged F1 score, a metric adopted by the database [32], of all models on the AF dataset. In Table VII, we show the anomaly and quality detection performance. The performance metric used by the database [34] is calculated as the mean of sensitivity and specificity scores. In the AF dataset, the averaged F1 scores obtained from the baseline NBoF and TNBoF models are significantly higher than the one obtained from the recurrent model. Although the improvement margins for the NBoF model are minor in Table VI, both the NBoF and TNBoF models enjoy increases in performance when using the proposed attention blocks. The consistent performance gain produced by the attention blocks can also be observed in the PCG dataset in Table VII. In this dataset, while the NBoF and TNBoF models score far below the GRU model, the attention-based models perform nearly as well as

the recurrent model.

## V. CONCLUSIONS

In this paper, we proposed 2D-Attention, a generic attention mechanism for data represented in the form of matrices. The proposed attention computation can be used in a plug-and-play manner, and can be updated jointly with other components in a computation graph. Using the proposed attention block, we further proposed three variants of the Neural Bag-of-Features model when learning with sequence data. Our extensive experiments in financial forecasting, audio analysis and medical diagnosis demonstrated that the proposed attention consistently led to performance gains for the Neural Bag-of-Features models. Since 2D-Attention is a generic attention computation method for matrices, investigating its efficacy in other neural network models is an interesting research direction in the future works.

## VI. ACKNOWLEDGEMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR). This publication reflects the authors views only. The European Commission is not responsible for any use that may be made of the information it contains.

## APPENDIX

In all of our experiments, we used ADAM optimizer for stochastic optimization. Weight decay (0.0001) or max-norm constraint (4.0) was used to for regularization. In addition, dropout (0.2) was applied to the output of the layer before the classification layer. In all models, before the output layer, there is a fully-connected layer with 512 neurons. For NBoF, TNBoF and the attention models, we used 256 codewords in the quantization layer. Correspondingly, the number of units in GRU model was set to 256. Details that are specific to each experiment are provided below:

- *Financial Forecasting Experiments*: All models were trained for 80 epochs, with the initial learning rate set to 0.001. The learning rate was decreased by a factor of 0.1 at epoch 11 and 51. We followed [10] and scaled the loss term associated with each class with a factor that is inversely proportional to the number of samples of each class to counter the effect of class imbalanced. In experiments that used convolution layers as preprocessing layers, we used two 1D convolution layers, each of which has 64 filters with the filter size set to 5 and the stride set to 1. Batch normalization was used after each convolution layer, followed by the ReLU activation.
- *Audio Analysis Experiments*: The setup is similar to the financial forecasting experiments, except for the configuration of convolution layers: four 1D convolution layers with the filter size of 5 were used; the first two convolution layers have 32 filters, which are followed by a max-pooling layer to reduce the temporal dimension by half. The last two convolution layers have 64 filters. After

each convolution layer, we applied batch normalization, followed by ReLU activation.

- *Medical Diagnosis Experiments*: in both AF and PCG datasets, all models were trained for 90 epochs, with the initial learning rate set to 0.001, which was decreased to 0.0001 at epoch 11, then to 0.00001 at epoch 71. For the AF dataset, we adopted the convolution architecture proposed in [33] as the first computation block in all models. For PCG dataset, we used five 1D convolution layers with the filter size set to 3 as the preprocessing layers: the first two layers have 32 filters with strides of 1; the third layer has 64 filters with strides of 2; the fourth layer has 64 filters with strides of 1; the last layer has 128 filters with strides of 2. After each convolution layer, we applied batch normalization, followed by ReLU activation.

## REFERENCES

- [1] D. T. Tran, M. Magris, J. Kannianen, M. Gabbouj, and A. Iosifidis, "Tensor representation in high-frequency financial data for price change prediction," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–7, IEEE, 2017.
- [2] A. Ntakaris, M. Magris, J. Kannianen, M. Gabbouj, and A. Iosifidis, "Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods," *Journal of Forecasting*, vol. 37, no. 8, pp. 852–866, 2018.
- [3] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6645–6649, IEEE, 2013.
- [4] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events," *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1733–1746, 2015.
- [5] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.
- [6] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.
- [7] E. Slutsky, "The summation of random causes as the source of cyclic processes," *Econometrica: Journal of the Econometric Society*, pp. 105–146, 1937.
- [8] G. C. Tiao and G. E. Box, "Modeling multiple time series with applications," *journal of the American Statistical Association*, vol. 76, no. 376, pp. 802–816, 1981.
- [9] D. T. Tran, M. Gabbouj, and A. Iosifidis, "Multilinear class-specific discriminant analysis," *Pattern Recognition Letters*, vol. 100, pp. 131–136, 2017.
- [10] D. T. Tran, A. Iosifidis, J. Kannianen, and M. Gabbouj, "Temporal attention-augmented bilinear network for financial time-series data analysis," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 5, pp. 1407–1418, 2018.
- [11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [12] N. Passalis and A. Tefas, "Neural bag-of-features learning," *Pattern Recognition*, vol. 64, pp. 277–294, 2017.
- [13] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," *Journal of Systems Engineering and Electronics*, vol. 28, no. 1, pp. 162–169, 2017.
- [14] N. Passalis, A. Tsantekidis, A. Tefas, J. Kannianen, M. Gabbouj, and A. Iosifidis, "Time-series classification using neural bag-of-features," in *2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 301–305, IEEE, 2017.
- [15] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 2169–2178, IEEE, 2006.

- [16] Y.-G. Jiang, C.-W. Ngo, and J. Yang, "Towards optimal bag-of-features for object categorization and semantic video retrieval," in *Proceedings of the 6th ACM international conference on Image and video retrieval*, pp. 494–501, 2007.
- [17] M. Riley, E. Heinen, and J. Ghosh, "A text retrieval approach to content-based audio retrieval," in *Int. Symp. on Music Information Retrieval (ISMIR)*, pp. 295–300, 2008.
- [18] A. Iosifidis, A. Tefas, and I. Pitas, "Multidimensional sequence classification based on fuzzy distances and discriminant analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2564–2575, 2012.
- [19] A. Iosifidis, A. Tefas, and I. Pitas, "Discriminant bag of words based representation for human action recognition," *Pattern Recognition Letters*, vol. 49, pp. 185–192, 2014.
- [20] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2, pp. 1150–1157, Ieee, 1999.
- [21] N. Passalis and A. Tefas, "Entropy optimized feature-based bag-of-words representation for information retrieval," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 7, pp. 1664–1677, 2016.
- [22] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, pp. 2048–2057, 2015.
- [23] V. Mnih, N. Heess, A. Graves, *et al.*, "Recurrent models of visual attention," in *Advances in neural information processing systems*, pp. 2204–2212, 2014.
- [24] F. Laakom, N. Passalis, J. Raitoharju, J. Nikkanen, A. Tefas, A. Iosifidis, and M. Gabbouj, "Bag of color features for color constancy," *arXiv preprint arXiv:1906.04445*, 2019.
- [25] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," *arXiv preprint arXiv:1704.02971*, 2017.
- [26] N. Passalis, A. Tefas, J. Kannianen, M. Gabbouj, and A. Iosifidis, "Temporal logistic neural bag-of-features for financial time series forecasting leveraging limit order book data," *arXiv preprint arXiv:1901.08280*, 2019.
- [27] N. Passalis, A. Tefas, J. Kannianen, M. Gabbouj, and A. Iosifidis, "Temporal bag-of-features learning for predicting mid price movements using high frequency limit order book data," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2018.
- [28] Y. G. Cinar, H. Mirisae, P. Goswami, E. Gaussier, A. Ait-Bachir, and V. Strijov, "Position-based content attention for time series forecasting with sequence-to-sequence rnns," in *International Conference on Neural Information Processing*, pp. 533–544, Springer, 2017.
- [29] O. B. Sezer, M. U. Gudelek, and A. M. Ozbayoglu, "Financial time series forecasting with deep learning: A systematic literature review: 2005–2019," *Applied Soft Computing*, vol. 90, p. 106181, 2020.
- [30] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "Fma: A dataset for music analysis," *arXiv preprint arXiv:1612.01840*, 2016.
- [31] A. Mesaros, T. Heittola, and T. Virtanen, "Tut acoustic scenes 2017, evaluation dataset," Nov. 2017.
- [32] G. D. Clifford, C. Liu, B. Moody, H. L. Li-wei, I. Silva, Q. Li, A. Johnson, and R. G. Mark, "Af classification from a short single lead ecg recording: the physionet/computing in cardiology challenge 2017," in *2017 Computing in Cardiology (CinC)*, pp. 1–4, IEEE, 2017.
- [33] F. Andreotti, O. Carr, M. A. Pimentel, A. Mahdi, and M. De Vos, "Comparing feature-based classifiers and convolutional neural networks to detect arrhythmia from short segments of ecg," in *2017 Computing in Cardiology (CinC)*, pp. 1–4, IEEE, 2017.
- [34] G. D. Clifford, C. Liu, B. Moody, D. Springer, I. Silva, Q. Li, and R. G. Mark, "Classification of normal/abnormal heart sound recordings: The physionet/computing in cardiology challenge 2016," in *2016 Computing in Cardiology Conference (CinC)*, pp. 609–612, IEEE, 2016.

## **7.15 Probabilistic Class-Specific Classification**

The appended paper follows.

Received September 13, 2020, accepted October 4, 2020, date of publication October 8, 2020, date of current version October 20, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3029514

# Probabilistic Class-Specific Discriminant Analysis

ALEXANDROS IOSIFIDIS<sup>1</sup>, (Senior Member, IEEE)

Department of Engineering, Aarhus University, DK-8200 Aarhus, Denmark

e-mail: alexandros.iosifidis@eng.au.dk

This work was supported by the European Union's Horizon 2020 Research and Innovation Programme under Grant 871449 (OpenDR).

**ABSTRACT** In this paper we formulate a probabilistic model for class-specific discriminant subspace learning. The proposed model can naturally incorporate the multi-modal structure of the negative class, which is neglected by existing class-specific methods. Moreover, it can be directly used to define a class-specific probabilistic classification rule in the discriminant subspace. We show that existing class-specific discriminant analysis methods are special cases of the proposed probabilistic model and, by casting them as probabilistic models, they can be extended to class-specific classifiers. We illustrate the performance of the proposed model in both verification and classification problems.

**INDEX TERMS** Class-specific discriminant analysis, multi-modal data distributions, verification, classification.

## I. INTRODUCTION

Class-Specific Discriminant Analysis (CSDA) [1], [14], [19], [38], determines an optimal subspace suitable for verification problems, where the objective is the discrimination of the class of interest from the rest of the world. As an example, let us consider the person identification problem, either through face verification [22], or through exploiting appearance or movement information [4], [15]. Different from person recognition, which is a multi-class classification problem assigning a sample (facial image or movement sequence) to a class in a pre-defined set of classes (person IDs in this case), person identification discriminates the person of interest from all rest people.

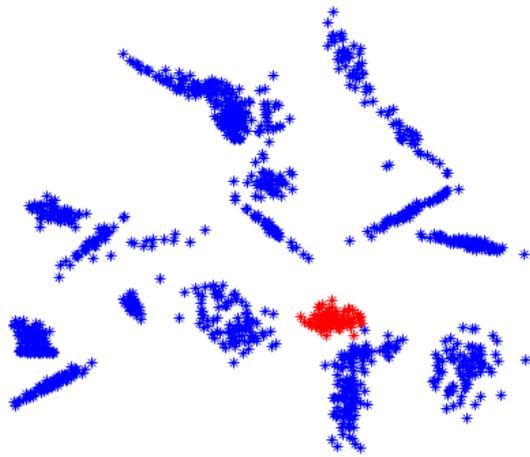
While multi-class discriminant analysis models, like Linear Discriminant Analysis (LDA) and its variants [7], [16], [18], [37], [40], [43] can be applied in such problems, they are inherently limited by the adopted class discrimination definition. That is, the maximal dimensionality of the resulting subspace is restricted by the number of classes, due to the rank of the between-class scatter matrix. In verification problems involving two classes LDA and its variants lead to one-dimensional subspaces. On the other hand, CSDA by expressing class discrimination using the out-of-class and intra-class scatter matrices is able to define subspaces the dimensionality of which is restricted by the the number of samples forming the smallest class (which is usually the

class of interest) or the number of original space dimensions. By defining multiple discriminant dimensions, CSDA has been shown to achieve better class discrimination and better performance in verification problems compared to LDA [1], [19].

While the definition of class discrimination in CSDA and its variants based on the intra-class and out-of-class scatter matrices overcomes the limitations of LDA related to the dimensionality of the discriminant subspace, it overlooks the structure of the negative class. Since in practice samples forming the negative class belong to many classes, different from the positive one, it is expected that they will form subclasses, as illustrated in Figure 1. Class discrimination as defined by CSDA and its variants disregards this structure. Related research in multi-class discriminant analysis indicates that exploitation of subclass information can enhance discrimination power [5], [17], [42], [44].

In this paper, we formulate the class-specific discriminant analysis optimization problem based on a probabilistic model that incorporates the above-described structure of the negative class. We show that the optimization criterion used by standard CSDA and its variants corresponds to a special case of the proposed probabilistic model, while new discriminant subspaces can be obtained by allowing samples of the negative class to form subclasses automatically determined by applying (unsupervised) clustering techniques on the negative class data. Moreover, the use of the proposed probabilistic model for class-specific discriminant learning naturally leads to a classification rule in the discriminant

The associate editor coordinating the review of this manuscript and approving it for publication was Hualong Yu<sup>1</sup>.



**FIGURE 1.** A toy example of a 2-dimensional problem illustrating the case where data forming the class of interest (red) is surrounded by data forming other classes. In class-specific learning the labels of the other classes are not available, leading to the assignment of the corresponding data to the negative class label (blue).

subspace, something that is not possible when the standard CSDA criterion is considered.

**II. RELATED WORK**

Let us denote by  $\mathcal{S}_p = \{\mathbf{x}_1, \dots, \mathbf{x}_{N_p}\}$  a set of  $N_p$   $D$ -dimensional vectors representing samples of the positive class and by  $\mathcal{S}_n = \{\mathbf{x}_{N_p+1}, \dots, \mathbf{x}_N\}$ , where  $N = N_p + N_n$ , a set of  $N_n$  vectors representing samples of the negative class. In the following we consider the linear class-specific subspace learning case and we will describe how to perform nonlinear (kernel-based) class-specific subspace learning following the same processing steps in Section III-D. We would like determine a linear projection  $\mathbf{W} \in \mathbb{R}^{D \times d}$ , mapping  $\mathbf{x}_i$  to a  $d$ -dimensional subspace, i.e.  $\mathbf{z}_i = \mathbf{W}^T \mathbf{x}_i$  that enhances discrimination of the two classes.

Class-specific Discriminant Analysis defines the projection matrix  $\mathbf{W}$  as the one maximizing the following criterion:

$$\mathcal{J}(\mathbf{W}) = \frac{\text{Tr}(\mathbf{W}^T \mathbf{S}_n \mathbf{W})}{\text{Tr}(\mathbf{W}^T \mathbf{S}_p \mathbf{W})}, \tag{1}$$

where  $\text{Tr}(\cdot)$  is the trace operator.  $\mathbf{S}_n \in \mathbb{R}^{D \times D}$  and  $\mathbf{S}_p \in \mathbb{R}^{D \times D}$  are the out-of-class and intra-class scatter matrices:

$$\mathbf{S}_n = \sum_{\mathbf{x}_i \in \mathcal{S}_n} (\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^T \tag{2}$$

$$\mathbf{S}_p = \sum_{\mathbf{x}_i \in \mathcal{S}_p} (\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^T \tag{3}$$

where  $\mathbf{m}$  is the mean vector of the positive class, i.e.  $\mathbf{m} = \frac{1}{N_p} \sum_{\mathbf{x}_i \in \mathcal{S}_p} \mathbf{x}_i$ .  $\mathbf{W}$  is obtained by solving the generalized eigen-analysis problem of  $\mathbf{S}_n \mathbf{w} = \lambda \mathbf{S}_p \mathbf{w}$  and keeping the eigenvectors corresponding to the  $d$  largest eigenvalues [20]. In the case where  $\mathbf{S}_n$  is singular, a regularized version of the above problem is solved.

A Spectral Regression [2] based solution of (1) has been proposed in [1], [19]. Let us denote by  $\mathbf{w}$  an eigenvector of

the generalized eigen-analysis problem  $\mathbf{S}_n \mathbf{w} = \lambda \mathbf{S}_p \mathbf{w}$  with eigenvalue  $\lambda$ . By setting  $\mathbf{X}^T \mathbf{w} = \mathbf{v}$  ( $\mathbf{X}$  being the data matrix), the original eigen-analysis problem can be transformed to the following eigen-analysis problem  $\mathbf{P}_n \mathbf{v} = \lambda \mathbf{P}_p \mathbf{v}$ , where  $\mathbf{P}_n = \mathbf{e}_n \mathbf{e}_n^T - \frac{1}{N_p} \mathbf{e}_n \mathbf{e}_p^T - \frac{1}{N_p} \mathbf{e}_p \mathbf{e}_n^T + \frac{1}{N_p^2} \mathbf{e}_p \mathbf{e}_p^T$  and  $\mathbf{P}_p = (1 - \frac{2}{N_p} + \frac{1}{N_p^2}) \mathbf{e}_p \mathbf{e}_p^T$ . Here  $\mathbf{e}_p \in \mathbb{R}^N$  and  $\mathbf{e}_n \in \mathbb{R}^N$  are the positive and negative class indicator vectors.  $\mathbf{W}$  is then obtained by applying a two-step process:

- Solution of the eigen-analysis problem  $\mathbf{P}_n \mathbf{v} = \lambda \mathbf{P}_p \mathbf{v}$  to determine the matrix  $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_d]$ , where  $\mathbf{v}_i$  is the eigenvector corresponding to the  $i$ -th largest eigenvalue.
- Calculation of  $\mathbf{w}_i, i = 1, \dots, d$  such that  $\mathbf{X}^T \mathbf{w}_i = \mathbf{v}_i$ .

It has been shown in [12] that the generalized eigenvectors  $\mathbf{v}$  of  $\mathbf{P}_p$  and  $\mathbf{P}_n$  can be directly obtained using the labeling information of the training vectors. However, in that case the order of the eigenvectors is not related to their discrimination ability. It has been also shown in [13] that the class-specific discriminant analysis problem in (1) is equivalent to a low-rank regression problem in which the target vectors are the same as those defined in [12], providing a new proof of the equivalence between class-specific discriminant analysis and class-specific spectral regression.

After determining the data projection matrix  $\mathbf{W}$ , the training vectors  $\mathbf{x}_i, i = 1, \dots, N$  are mapped to the discriminant subspace  $\mathbf{z}_i = \mathbf{W}^T \mathbf{x}_i$ . When a classification problem is considered, a classifier is trained using  $\mathbf{z}_i$ . For example, [19] trains a linear SVM on vectors  $\mathbf{d}_i = |\mathbf{z}_i - \boldsymbol{\mu}|$ , where  $\boldsymbol{\mu} = \mathbf{W}^T \mathbf{m}$  and the absolute value operator is applied element-wise on the resulting vector. Due to the need of training an additional classifier, class-specific discriminant analysis models are usually employed in class-specific ranking settings, where test vectors  $\mathbf{x}_j^*$  are mapped to the discriminant subspace  $\mathbf{z}_j^* = \mathbf{W}^T \mathbf{x}_j^*$  and are subsequently ordered based on their distance w.r.t. the positive class mean  $d_j = \|\mathbf{z}_j^* - \boldsymbol{\mu}\|_2$ .

**III. PROBABILISTIC CLASS-SPECIFIC LEARNING**

In this section, we follow a probabilistic approach for defining a class-specific discrimination criterion that is able to encode subclass information of the negative class. We call the proposed method Probabilistic Class-Specific Discriminant Analysis (PCSDA). PCSDA defines a subspace  $\mathbb{R}^d, d \leq D$  of a feature space  $\mathbb{R}^D$  such that the positive class is optimally discriminated from the negative class, based on the assumption that the negative class is formed by multiple subclasses having the same cardinality. We show how to relax this assumptions in the next subsection.

Let us denote by  $\mathbf{x} \in \mathbb{R}^D$  a random variable, realizations of which correspond to samples of the positive and negative classes in our problem. PCSDA assumes there exists a positive class  $c_p$  following a multivariate Gaussian distribution defined by the mean vector  $\mathbf{m} \in \mathbb{R}^D$  and the covariance matrix  $\boldsymbol{\Phi}_p \in \mathbb{R}^{D \times D}$ , i.e.:

$$P(\mathbf{x}|c_p) \sim N(\mathbf{x}|\mathbf{m}, \boldsymbol{\Phi}_p). \tag{4}$$

Since the negative class  $c_n$  is formed by samples belonging to multiple classes (which are not distinguished by the labeling information available during training), PCSDA assumes that the negative class forms subclasses, each of which follows a multivariate Gaussian distribution in  $\mathbb{R}^D$ . Let  $\mathbf{y} \in \mathbb{R}^D$  be a random variable a realization of which corresponds to the mean vector of a negative subclass. Since in class-specific learning we are interested in maximizing the scatter of the negative data from the positive class (represented by  $\mathbf{m}$ ), we model the distribution of  $\mathbf{y}$  with respect to  $\mathbf{m}$  as a multivariate Gaussian distribution:

$$P(\mathbf{y}) \sim N(\mathbf{y}|\mathbf{m}, \Phi_n), \quad (5)$$

where  $\Phi_n$  is the corresponding covariance matrix expressing the scatter of the negative subclasses with respect to the positive class mean  $\mathbf{m}$ . Then, the distribution of the samples from the negative subclasses with respect to the positive class is expressed by the following multi-modal distribution [29]:

$$P(\mathbf{x}|c_n) = \int N(\mathbf{y}|\mathbf{m}, \Phi_n) N(\mathbf{x}|\mathbf{y}, \Phi_w) d\mathbf{y}, \quad (6)$$

where  $\Phi_w$  is the (common) covariance matrix of the negative subclasses.

#### A. TRAINING PHASE

Given a set of positive i.i.d. samples  $S_p = \{\mathbf{x}_1, \dots, \mathbf{x}_{N_p}\}$  the probability of correct assignment under our model is:

$$P(S_p|c_p) = \prod_{i=1}^{N_p} P(\mathbf{x}_i|c_p). \quad (7)$$

Let us assume that negative class is formed by samples  $\mathbf{x}_i$ ,  $i = N_p + 1, \dots, N$  belonging to  $K$  subclasses, i.e.  $S_n = \{S_1, \dots, S_K\}$ , each having a cardinality of  $M = N_n/K$ . The probability of assigning each of the negative samples to the corresponding negative subclass is given by:

$$P(S_k|c_n) = \int N(\mathbf{y}|\mathbf{m}, \Phi_n) \prod_{\mathbf{x}_i \in S_k} P(\mathbf{x}_i|\mathbf{y}, \Phi_w) d\mathbf{y}. \quad (8)$$

Without loss of generality, we assume that  $\mathbf{m} = \mathbf{0}$  (this can always be done by setting  $\mathbf{x}_i - \mathbf{m} \rightarrow \mathbf{x}_i$ ). Then:

$$P(S_p|c_p) = \frac{1}{(2\pi)^{\frac{N_p D}{2}} |\Phi_p|^{\frac{N_p}{2}}} \exp\left(-\frac{1}{2} \text{Tr}(\Phi_p^{-1} \mathbf{S}_p)\right) \quad (9)$$

$$P(S_k|c_n) = \frac{1}{M^{-\frac{D}{2}} (2\pi)^{\frac{MD}{2}} |\Phi_w|^{\frac{M-1}{2}} |\Phi_n + \frac{1}{M} \Phi_w|^{\frac{1}{2}}} \cdot \exp\left(-\frac{1}{2} \text{Tr}((\Phi_n + \frac{1}{M} \Phi_w)^{-1} \mathbf{S}_n^{(k)})\right) \cdot \exp\left(-\frac{1}{2} \text{Tr}(\Phi_w^{-1} \mathbf{S}_w^{(k)})\right). \quad (10)$$

In the above,  $\mathbf{S}_p$  is the scatter matrix of the positive class, i.e.  $\mathbf{S}_p = \sum_{\mathbf{x}_i \in S_p} \mathbf{x}_i \mathbf{x}_i^T$ ,  $\mathbf{S}_w^{(k)}$  is the within-subclass scatter matrix of the  $k$ -th negative subclass, i.e.  $\mathbf{S}_w^{(k)} = \sum_{\mathbf{x}_i \in S_k} (\mathbf{x}_i - \bar{\mathbf{x}}_k)(\mathbf{x}_i - \bar{\mathbf{x}}_k)^T$  and  $\mathbf{S}_n^{(k)}$  is the scatter matrix of the  $k$ -th subclass

w.r.t. the mean vector of the positive class  $\mathbf{m} = \mathbf{0}$ , i.e.  $\mathbf{S}_n^{(k)} = \bar{\mathbf{x}}_k \bar{\mathbf{x}}_k^T$ , where  $\bar{\mathbf{x}}_k$  is the mean vector of the  $k$ -th negative subclass, i.e.  $\bar{\mathbf{x}}_k = \frac{1}{M} \sum_{\mathbf{x}_i \in S_k} \mathbf{x}_i$ . Derivations leading to the above results can be found in the supplementary document. Since the assignment of the negative samples to subclasses is not provided by the labels used during training, we define them by applying a clustering method (e.g.  $K$ -Means) on the negative class vectors.

The parameters of the proposed PCSDA are the covariance matrices  $\Phi_p$ ,  $\Phi_n$  and  $\Phi_w$  defining the data generation processes for the positive and negative classes. These parameters are estimated by maximizing the (log) probability of correct assignment of the training samples  $\mathbf{x}_i$ ,  $i = 1, \dots, N$ :

$$\mathcal{L} = \ln P(S_p|c_p) + \ln P(S_n|c_n), \quad (11)$$

where

$$\ln P(S_p|c_p) = C_1 - \frac{N_p}{2} \ln |\Phi_p| - \frac{1}{2} \text{Tr}(\Phi_p^{-1} \mathbf{S}_p) \quad (12)$$

and

$$\begin{aligned} \ln P(S_n|c_n) &= \sum_{k=1}^K \ln P(S_k|c_n) = C_2 - \frac{N_n - K}{2} \ln |\Phi_w| \\ &\quad - \frac{K}{2} \ln |\Phi_n + \frac{1}{M} \Phi_w| - \frac{1}{2} \text{Tr}(\Phi_w^{-1} \mathbf{S}_w) \\ &\quad - \frac{1}{2} \text{Tr}((\Phi_n + \frac{1}{M} \Phi_w)^{-1} \mathbf{S}_n), \end{aligned} \quad (13)$$

where  $\mathbf{S}_n$  and  $\mathbf{S}_w$  are the total scatter matrices of the negative subclasses, i.e.,  $\mathbf{S}_w = \sum_{k=1}^K \mathbf{S}_w^{(k)}$  and  $\mathbf{S}_n = \sum_{k=1}^K \mathbf{S}_n^{(k)}$ .

By substituting (12) and (13) in (11) the optimization problem takes the form:

$$\begin{aligned} \mathcal{L} &= C_3 - \frac{N_p}{2} \ln |\Phi_p| - \frac{1}{2} \text{Tr}(\Phi_p^{-1} \mathbf{S}_p) \\ &\quad - \frac{1}{2} \text{Tr}(\Phi_w^{-1} \mathbf{S}_w) - \frac{1}{2} \text{Tr}((\Phi_n + \frac{1}{M} \Phi_w)^{-1} \mathbf{S}_n) \\ &\quad - \frac{N_n - K}{2} \ln |\Phi_w| - \frac{K}{2} \ln |\Phi_n + \frac{1}{M} \Phi_w|. \end{aligned} \quad (14)$$

The saddle points of  $\mathcal{L}$  with respect to  $\Phi_p$ ,  $\Phi_n$ ,  $\Phi_w$  lead to:

$$\frac{\partial \mathcal{L}}{\partial \Phi_p} = \mathbf{0} \Rightarrow \Phi_p = \frac{1}{N_p} \mathbf{S}_p \quad (15)$$

$$\frac{\partial \mathcal{L}}{\partial \Phi_n} = \mathbf{0} \Rightarrow \Phi_n + \frac{1}{M} \Phi_w = \frac{1}{K} \mathbf{S}_n \quad (16)$$

$$\frac{\partial \mathcal{L}}{\partial \Phi_w} = \mathbf{0} \Rightarrow \Phi_w = \frac{1}{N_n - K} \mathbf{S}_w. \quad (17)$$

Note that when  $N_n = K$  (17) takes an indeterminate form (i.e.  $0/0$ ). However, in this case each negative subclass is formed by one sample and by definition  $\Phi_w = \mathbf{0}$ . This is discussed more in subsection III-D. Combining (16) and (17) we get:

$$\Phi_n = \frac{1}{K} \mathbf{S}_n - \frac{1}{M(N_n - K)} \mathbf{S}_w. \quad (18)$$

By combining (17) and (18) we can define the following matrix:

$$\Phi_O = \Phi_n + \Phi_w = \frac{1}{K} \mathbf{S}_n + \frac{1}{N_n} \mathbf{S}_w, \quad (19)$$

Thus, as can be seen from (15), (17), (18) and (19), the parameters of PCSDA can be calculated using  $\mathbf{S}_p$ ,  $\mathbf{S}_n$  and  $\mathbf{S}_w$  defined on the training vectors  $\mathbf{x}_i$ ,  $i = 1, \dots, N$ .

Using  $\mathbf{S}_p$ ,  $\mathbf{S}_n$  and  $\mathbf{S}_w$  a data transformation  $\mathbf{W} \in \mathbb{R}^{D \times D}$  can be obtained by optimizing for:

$$\mathbf{S}_n \mathbf{w} = \lambda (\mathbf{S}_p + \mathbf{S}_w) \mathbf{w}. \quad (20)$$

Problem (20) corresponds to the generalized eigen-analysis problem of the matrices  $\mathbf{S}_n$  and  $\mathbf{S}_I = \mathbf{S}_p + \mathbf{S}_w$ . Since the rank of  $\mathbf{S}_n$  is  $K$ , the dimensionality of the obtained subspace is restricted to  $d \leq \min(D, K)$ . Eigenvectors in PCSDA define a data transformation that minimizes the intra-class variance of the positive class and the intra-cluster variance of the negative subclasses, while at the same time maps the means of the negative subclasses as far as possible from the positive class mean. Since the above-described property is optimized by treating (20) as maximization problems, the eigenvectors forming  $\mathbf{W}$  are sorted in a decreasing order of the corresponding eigenvalues.

After obtaining  $\mathbf{W}$ , the intra-class and out-of-class covariance matrices of the transformed data  $\mathbf{z}_i = \mathbf{W}^T \mathbf{x}_i$  are given by:

$$\tilde{\Phi}_p = \mathbf{W}^T \Phi_p \mathbf{W} \quad \text{and} \quad \tilde{\Phi}_O = \mathbf{W}^T \Phi_O \mathbf{W}. \quad (21)$$

The process followed in the training phase of PCSDA is illustrated in Algorithm 1.

In the above we set the assumption that the number of samples forming the negative subclasses is equal. This assumption can be relaxed following one of the following approaches. After assigning all negative samples to subclasses and calculating the negative subclass distributions, one can sample  $M$  vectors from each distribution. Alternatively, one can calculate the total within-subclass matrix of the negative class by  $\mathbf{S}_w = \sum_{k=1}^K \mathbf{S}_w^{(k)}$ . The latter approach is commonly used in multi-class discriminant analysis variants [16]. Note that for the model's parameters calculation, only the overall scatter matrices  $\mathbf{S}_p$ ,  $\mathbf{S}_w$  and  $\mathbf{S}_n$  are used.

### B. TEST PHASE

After the estimation of the model's parameters, a new sample  $\mathbf{x}^*$  can be evaluated. The posterior probabilities of the positive and negative classes are given by:

$$P(c_p | \mathbf{x}^*) = \frac{p(\mathbf{x}^* | c_p) P(c_p)}{p(\mathbf{x}^*)} \quad (22)$$

$$P(c_n | \mathbf{x}^*) = \frac{p(\mathbf{x}^* | c_n) P(c_n)}{p(\mathbf{x}^*)}. \quad (23)$$

The a priori class probabilities  $P(c_p)$  and  $P(c_n)$  can be calculated by the proportion of the positive and negative samples in the training phase, i.e.  $P(c_p) = N_p/N$  and  $P(c_n) = N_n/N$ . Depending on the problem at hand, it may be sometimes preferable to consider equiprobable classes, i.e.  $P(c_p) = P(c_n) = 1/2$ , leading to the maximum likelihood classification case. The class-conditional probabilities of the

---

### Algorithm 1 PCSDA: Training Phase

---

**Data:**  $K$ ,  $d$  and training data  $\{\mathbf{x}_i, l_i\}_{i=1, \dots, N}$

**Result:**  $\tilde{\Phi}_p$ ,  $\tilde{\Phi}_O$ ,  $\mathbf{W}$ ,  $\mathbf{m}$ ,  $P(c_p)$ ,  $P(c_n)$

**if**  $K < N_n$  **then**

    | Cluster  $\{\mathbf{x}_i\}_{l_i \neq 1}$  to form  $\mathcal{S}_k$ ,  $k = 1, \dots, K$

**end if**

Calculate  $\mathbf{m}$ ,  $\mathbf{S}_p$ ,  $\mathbf{S}_n$  and  $\mathbf{S}_w$

Calculate  $\Phi_p$  from (15) and  $\Phi_O$  from (19)

Calculate  $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_d]$  from (20)

Calculate  $\tilde{\Phi}_p$  and  $\tilde{\Phi}_O$  from (21)

---

transformed sample  $\mathbf{z}^* = \mathbf{W}^T \mathbf{x}^*$  are given by:

$$\begin{aligned} p(\mathbf{x}^* | c_p) &= \frac{1}{(2\pi)^{\frac{D}{2}} |\Phi_p|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} \mathbf{x}^{*T} \Phi_p^{-1} \mathbf{x}^*\right) \\ &= \frac{1}{(2\pi)^{\frac{D}{2}} |\tilde{\Phi}_p|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} \mathbf{z}^{*T} \tilde{\Phi}_p^{-1} \mathbf{z}^*\right) \end{aligned} \quad (24)$$

and

$$\begin{aligned} p(\mathbf{x}^* | c_n) &= \frac{1}{(2\pi)^{\frac{D}{2}} |\Phi_O|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} \mathbf{x}^{*T} \Phi_O^{-1} \mathbf{x}^*\right) \\ &= \frac{1}{(2\pi)^{\frac{D}{2}} |\tilde{\Phi}_O|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} \mathbf{z}^{*T} \tilde{\Phi}_O^{-1} \mathbf{z}^*\right). \end{aligned} \quad (25)$$

In the above, we used the orthogonal property of the matrix  $\mathbf{W}$ . In the case where we want  $\mathbf{z}^* \in \mathbb{R}^d$ ,  $d < D$ , we keep the dimensions of  $\mathbf{z}^*$  corresponding to the first  $d$  columns of  $\mathbf{W}$ .

Combining (22)-(25) the ratio of class posterior probabilities is equal to:

$$\lambda(\mathbf{z}^*) = \frac{P(c_p) |\tilde{\Phi}_O|^{\frac{1}{2}} \exp\left(-\frac{1}{2} \mathbf{z}^{*T} \tilde{\Phi}_p^{-1} \mathbf{z}^*\right)}{P(c_n) |\tilde{\Phi}_p|^{\frac{1}{2}} \exp\left(-\frac{1}{2} \mathbf{z}^{*T} \tilde{\Phi}_O^{-1} \mathbf{z}^*\right)}. \quad (26)$$

$\lambda(\mathbf{z}^*)$  can be used to classify  $\mathbf{z}^*$  to the positive class when  $\lambda(\mathbf{z}^*) > 1$  and to the negative class otherwise. Alternatively, (26) can be also used to define the classification rule:

$$\begin{aligned} g(\mathbf{z}^*) &= \ln P(c_p) - \ln P(c_n) \\ &\quad + \frac{1}{2} \ln |\tilde{\Phi}_O| - \frac{1}{2} \ln |\tilde{\Phi}_p| \\ &\quad - \frac{1}{2} \mathbf{z}^{*T} \tilde{\Phi}_p^{-1} \mathbf{z}^* + \frac{1}{2} \mathbf{z}^{*T} \tilde{\Phi}_O^{-1} \mathbf{z}^* \end{aligned} \quad (27)$$

classifying  $\mathbf{z}^*$  to the positive class if  $g(\mathbf{z}^*) > 0$  and to the negative class otherwise.

In class-specific ranking settings one can follow the process applied when using the standard CSDA approach. First, test vectors  $\mathbf{x}_j^*$  are mapped to the discriminant subspace  $\mathbf{z}_j^* = \mathbf{W}^T \mathbf{x}_j^*$  and  $\mathbf{z}_j^*$ ,  $i = 1, \dots, N$  are obtained. Then,  $\mathbf{z}_j^*$ 's are ordered based on their distance w.r.t. the positive class mean  $d_j = \|\mathbf{z}_j^* - \boldsymbol{\mu}\|_2$ , where  $\boldsymbol{\mu}$  is the mean of positive training samples in  $\mathbb{R}^d$ . The process followed in the test phase of PCSDA is illustrated in Algorithm 2.

**Algorithm 2** PCSDA: Test Phase

---

**Data:**  $\tilde{\Phi}_p, \tilde{\Phi}_O, \mathbf{W}, \mathbf{m}, P(c_p), P(c_n)$  and test data  $\mathbf{x}_i^*, i = 1, \dots, M$

**Result:** Predicted labels  $l_i, i = 1, \dots, M$  or ranking order  $o_i, i = 1, \dots, M$

Calculate  $\boldsymbol{\mu} = \mathbf{W}^T \mathbf{m}$  and  $\mathbf{z}_i^* = \mathbf{W}^T \mathbf{x}_i^*, i = 1, \dots, M$

**if** Classification **then**

**for**  $i = 1:M$  **do**

Calculate  $\tilde{\mathbf{z}}_i^* = \mathbf{z}_i^* - \boldsymbol{\mu}$

**if**  $g(\tilde{\mathbf{z}}_i^*) \geq 0$  **then**

$l_i = 1;$

**else**

$l_i = 0;$

**end if**

**end for**

**else**

Calculate  $d_i = \|\boldsymbol{\mu} - \mathbf{z}_i^*\|_2, i = 1, \dots, M$

Sort  $d_i$ 's to obtain the order  $o_i, i = 1, \dots, M$

**end if**

---

**C. SPECTRAL REGRESSION SOLUTION OF PCSDA**

We further show in the following that PCSDA can be efficiently solved by following a spectral regression based process. Let us denote by  $\mathbf{1}_p \in \mathbb{R}^N$  a vector having ones in the elements corresponding to the positive training vectors and zeros elsewhere. In the same manner, we define the vector  $\mathbf{1}_k \in \mathbb{R}^N$  as a vector having ones in the elements corresponding to the negative samples belonging to the  $k$ -th subclass and zeros elsewhere. We also define by  $\mathbf{J}_p \in \mathbb{R}^{N \times N}$  and  $\mathbf{J}_k \in \mathbb{R}^{N \times N}$  the diagonal matrices having in their diagonal the vectors  $\mathbf{1}_p$  and  $\mathbf{1}_k$ , respectively. Then,  $\mathbf{S}_n$  and  $\mathbf{S}_l$  can be expressed as:

$$\mathbf{S}_n = \sum_{k=1}^K (\bar{\mathbf{x}}_k - \mathbf{m})(\bar{\mathbf{x}}_k - \mathbf{m})^T = \mathbf{X} \mathbf{L}_n \mathbf{X}^T, \quad (28)$$

$$\mathbf{S}_l = \mathbf{S}_p + \mathbf{S}_w = \mathbf{X} \mathbf{L}_l \mathbf{X}^T \quad (29)$$

where

$$\mathbf{L}_n = \sum_{k=1}^K \left( \frac{1}{N_k^2} \mathbf{1}_k \mathbf{1}_k^T - \frac{1}{N_k N_p} (\mathbf{1}_k \mathbf{1}_p^T + \mathbf{1}_p \mathbf{1}_k^T) + \frac{1}{N_p^2} \mathbf{1}_p \mathbf{1}_p^T \right) \quad (30)$$

$$\mathbf{L}_l = \left( \mathbf{J}_p - \frac{1}{N_p} \mathbf{1}_p \mathbf{1}_p^T \right) + \sum_{k=1}^K \left( \mathbf{J}_k - \frac{1}{N_k} \mathbf{1}_k \mathbf{1}_k^T \right). \quad (31)$$

Substituting (28) and (29) in (20) and setting  $\mathbf{v} = \mathbf{X}^T \mathbf{w}$ :

$$\mathbf{L}_n \mathbf{v} = \lambda \mathbf{L}_l \mathbf{v}. \quad (32)$$

Thus, the vectors  $\mathbf{v}$  are the eigenvectors of the matrix  $\tilde{\mathbf{L}} = \tilde{\mathbf{L}}_w^{-1} \mathbf{L}_n$ , where  $\tilde{\mathbf{L}}_w$  is a regularized version of  $\mathbf{L}_w$ , i.e.  $\tilde{\mathbf{L}}_w = \mathbf{L}_w + \epsilon \mathbf{I}$  where  $\epsilon > 0$ . It can be shown that the matrix  $\tilde{\mathbf{L}}$  is a block matrix formed by blocks the elements of which are indicated by the class labels of the positive data and the cluster labels of the negative data. The top  $K$  eigenvectors

of it are also formed by blocks indicated by the same labels. Thus, they can be defined without the need of solving the problem (32), in a similar manner as in [2], [13]. After the determination of the matrix  $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_K]$ , the solution of (20) is given by  $\mathbf{W} = (\mathbf{X}^T)^\dagger \mathbf{V}$ , where  $(\mathbf{X}^T)^\dagger$  is the pseudo-inverse of  $\mathbf{X}^T$ .

**D. CSDA VARIANTS UNDER THE PROBABILISTIC MODEL**

A special case of the PCSDA can be obtained by setting the assumption that each negative sample forms a negative subclass, i.e.  $K = N_n$  and  $M = 1$ . In that case  $\Phi_w = \mathbf{0}$ ,  $\Phi_O = \Phi_n$ , the negative samples are drawn from a distribution  $P(\mathbf{x}) \sim N(\cdot | \mathbf{m}, \Phi_n)$ , and  $\mathbf{W}$  is calculated by solving for  $\mathbf{S}_n \mathbf{w} = \tilde{\lambda} \mathbf{S}_p \mathbf{w}$ , i.e., we obtain the class discrimination definition of CSDA. The Spectral Regression-based solutions of CSDA in [1], [12], [19] and the low-rank regression solution of [13] use the same class discrimination criterion and, thus, correspond to the same setting of PCSDA. Since the discrimination criterion used in CSDA is a special case of the proposed probabilistic model, all the above-mentioned methods can be extended to perform classification using  $g(\cdot)$  in (27). Jointly optimizing the data projection and the classification rule can be considered a big advantage.

**E. CLASS-SPECIFIC VS. CONTRASTIVE LEARNING**

As can be seen from Eqs. (1), (2) and (3), the Class-specific Discriminant Analysis criterion has similarities with Contrastive Learning based on triplet loss [39], which is defined for multi-class problems. That is, in its generic form, it optimizes for class discrimination by using three samples, i.e. the sample of interest, a sample belonging to the same class with the sample of interest (defining the same-label pair), and a sample belonging to a different class (defining the different-label pair). Thus, it can be modeled by a graph-based criterion, as was shown in [10]. The class-specific learning approaches are defined on binary problems where the negative class is formed data coming from multiple classes, but are assigned the same (negative) label. All the class-specific discriminant analysis approaches (including the proposed one) are defined using variance criteria. They optimize for class discrimination by using the mean of the positive class, a positive sample, and a negative sample. Thus, they can be defined as graph-based methods having the Laplacian matrices  $\mathbf{L}_n$  and  $\mathbf{L}_l$  in Eqs. (30) and (31).

The adoption of class-specific learning has two advantages compared to contrastive learning when class-specific subspace learning is considered:

- Subspace learning based on triplet loss optimizes for two objectives:
  - 1) Given a positive vector of interest, it should be mapped close to another positive vector (same-label pair), and far away from a negative vector (different-label pair).
  - 2) Given a negative vector of interest, it should be mapped close to another negative vector

(same-label pair), and far away from a positive vector (different-label pair).

While objective 1 is desirable in class-specific problems, optimizing for objective 2 can be destructive. Optimizing for both objectives is good for multi-class problems when assuming class unimodality, like in the case of LDA.

- Class-specific learning defined by a variance-based criterion can lead to fast solutions. This is due to that the graph Laplacian matrices  $\mathbf{L}_n$  and  $\mathbf{L}_l$  do not need to be calculated and stored in the memory. Instead, their eigenvectors can be efficiently obtained and highly reduce the memory and computational cost. This is described in Section III-C.

**F. NON-LINEAR PCSDA**

In the above analysis we considered the linear class-specific subspace learning case. In order to non-linearly map  $\mathbf{x}_i \in \mathbb{R}^D$  to  $\mathbf{z}_i \in \mathbb{R}^d$  traditional kernel-based learning methods perform a non-linear mapping of the input space  $\mathbb{R}^D$  to the feature space  $\mathcal{F}$  using a function  $\phi(\cdot)$ , i.e.  $\mathbf{x}_i \in \mathbb{R}^D \rightarrow \phi(\mathbf{x}_i) \in \mathcal{F}$ . Then, linear class-specific projections are defined by using the training data in  $\mathcal{F}$ . Since the dimensionality of  $\mathcal{F}$  is arbitrary (virtually infinite), the data representations in  $\mathcal{F}$  cannot be calculated. Traditional kernel-based learning methods address this issue by exploiting the Representer theorem and the non-linear mapping is implicitly performed using the kernel function encoding dot products in the feature space, i.e.  $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$  [35].

As has been shown in [23] the effective dimensionality of the kernel space  $\mathcal{F}$  is at most equal to  $L = \min(D, N)$  and, thus, an explicit non-linear mapping  $\mathbf{x}_i \in \mathbb{R}^D \rightarrow \Phi_i \in \mathbb{R}^L$  can be calculated such that  $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \Phi_i^T \Phi_j$ . This is achieved by using  $\Phi = \Sigma^{\frac{1}{2}} \mathbf{U}^T$ , where  $\mathbf{U}$  and  $\Sigma$  contain the eigenvectors and eigenvalues of the kernel matrix  $\mathbf{K} \in \mathbb{R}^{N \times N}$  [24]. Thus, extension of PCSDA to the non-linear (kernel) case can be readily obtained by applying the above-described linear PCSDA on the vectors  $\Phi_i, i = 1, \dots, N$ . Here we need to mention that, since the proposed method exploits kernel-based data representations, it inherits drawbacks of kernel methods related to their high computational and memory costs when applied to large-scale problems. For the cases where the size of training set is prohibitive for applying kernel-based discriminant learning, the Nyström-based kernel subspace learning method of [11] or nonlinear data mappings based on randomized features, as proposed in [34], can be used. We should also note that the application of  $K$ -Means using in  $\mathbb{R}^L$  corresponds to the application of kernel  $K$ -Means in the original space  $\mathbb{R}^D$ .

**G. TIME COMPLEXITY**

The time complexity of PCSDA is as follows [9]:

- $K$ -Means application for determining the subclasses of the negative class having time complexity of  $O(\eta KN_n D)$ , where  $\eta$  is the number of iterations until convergence,

**TABLE 1. Datasets information.**

Dataset	$D$	#Samples	#Classes
Jaffe [27]	1200	210	7
Kanade [21]	1200	245	7
BU [41]	1200	700	7
YALE [26]	1200	2432	38
AR [28]	1200	2600	100
15 scenes [25]	512	4485	15
OptDigits [6]	54	5620	10
Caltech101 [33]	512	9145	102

- Calculation of the scatter matrices in (20) and solution of the generalized eigen-analysis problem having a time complexity of  $O(D^3 + D^2 N)$ ,

Keeping the high-order terms, the overall time complexity is of  $O(D^3 + D^2 N)$ .

Considering the kernel-based version of PCSDA, its time complexity is as follows:

- Kernel matrix  $\mathbf{K}$  calculation having a time complexity of  $O(DN^2)$ .
- Application of the method [23], involving the eigen-decomposition of  $\mathbf{K}$ , having a time complexity of  $O(N^3)$ .
- $K$ -Means application for determining the negative class subclasses having complexity of the order of  $O(\eta KN_n N)$ .
- Calculation of the scatter matrices in (20) and solution of the generalized eigen-analysis problem having a time complexity of  $O(N^3 + 2 N^2)$

Thus, the time complexity of the kernel version of PCSDA is  $O(N^3)$ , which is the case of a generic kernel-based subspace learning method. As noted in subsection III-F, for applying nonlinear PCSDA on big datasets, the approximate kernel subspace learning method in [11] or nonlinear data mappings based on randomized features, as proposed in [34], can be used leading to  $L \ll N$  and highly reducing the overall time complexity to  $O(L^3 + LN^2)$ .

Finally, considering the Spectral Regression-based solution described in Section III-C, the time complexity of a generic solution is the same as those described above, but efficient solutions are possible by using efficient decomposition methods, like Cholesky decomposition as in [13].

**IV. EXPERIMENTS**

In this Section we provide experimental results illustrating the performance of the proposed PCSDA method. We used eight datasets in our experiments, details of which are illustrated in Table 1. For BU, Jaffe, Kanade, YALE and AR facial image datasets we used the vectorized pixel intensity values for representing the images. For the 15 scenes and Caltech101 datasets we used deep features generated by average pooling over spatial dimension of the last convolution layer of VGG network [36] trained on ILSVRC2012 database. For optDigits dataset we used the raw features provided by the UCI repository [6].

**TABLE 2. Performance on retrieval problems.**

	RR	SVR	PCA	LDA	PCSDA-1	PCSDA-K
Jaffe	0.9024 (0.0935)	0.8954 (0.1179)	0.3010 (0.1383)	0.9052 (0.0990)	0.9099 (0.0953)	0.7591 (0.1728)
Kanade	0.6523 (0.2635)	0.6525 (0.2625)	0.2930 (0.1139)	0.6336 (0.2476)	0.6621 (0.2573)	0.4196 (0.2629)
BU	0.6848 (0.2279)	0.6976 (0.2346)	0.2903 (0.1455)	0.6491 (0.2099)	0.6840 (0.1940)	0.6520 (0.1902)
YALE	0.9874 (0.0250)	0.9883 (0.0231)	0.5580 (0.0410)	0.9864 (0.0251)	0.9864 (0.0251)	0.9911 (0.0189)
AR	0.9990 (0.0057)	0.9976 (0.0109)	0.8642 (0.0113)	0.9988 (0.0061)	0.9986 (0.0071)	0.9959 (0.0194)
15 scenes	0.9340 (0.0409)	0.9403 (0.0387)	0.6161 (0.1805)	0.9160 (0.0587)	0.9162 (0.0602)	0.9127 (0.0549)
OptDigits	0.9968 (0.0036)	0.9969 (0.0044)	0.8337 (0.1352)	0.9974 (0.0033)	0.9970 (0.0034)	0.9945 (0.0064)
Caltech101	0.8708 (0.1383)	0.8792 (0.1301)	0.2239 (0.2000)	0.8485 (0.1513)	0.8467 (0.1514)	0.8523 (0.1486)

**TABLE 3. Performance on classification problems.**

	RR	SVM	PCA+SVM	LDA+NCC	PCSDA-1	PCSDA-K
Jaffe	0.6750 (0.1032)	0.6714 (0.1225)	0.2809 (0.0542)	0.7049 (0.1299)	0.6261 (0.1262)	0.5242 (0.1249)
Kanade	0.4001 (0.1520)	0.4122 (0.1271)	0.2597 (0.0464)	0.5194 (0.2677)	0.4546 (0.2427)	0.3861 (0.1894)
BU	0.4463 (0.0819)	0.4562 (0.1391)	0.2592 (0.0180)	0.5632 (0.1706)	0.5731 (0.1288)	0.5431 (0.1793)
YALE	0.9369 (0.0090)	0.9396 (0.0851)	0.2715 (0.0851)	0.9517 (0.0208)	0.9409 (0.0214)	0.9611 (0.0222)
AR	0.9664 (0.0352)	0.9683 (0.0343)	0.7346 (0.0394)	0.9760 (0.0452)	0.9674 (0.0481)	0.9516 (0.0629)
15 scenes	0.8997 (0.0854)	0.9005 (0.0517)	0.5365 (0.1507)	0.8993 (0.0539)	0.8573 (0.0654)	0.8130 (0.1275)
OptDigits	0.9875 (0.0900)	0.9881 (0.0082)	0.8985 (0.0706)	0.9899 (0.0077)	0.9569 (0.0141)	0.9826 (0.0074)
Caltech101	0.8130 (0.0971)	0.8140 (0.0807)	0.6042 (0.0394)	0.8075 (0.1448)	0.7605 (0.1829)	0.7663 (0.1955)

On each experiment, we formed class-specific ranking and classification problems for each class using the class data as positive samples and the data of the remaining classes as negative samples. In all the experiments the data is non-linearly mapped to the subspace of the kernel space (as discussed in Section III-F). We used the RBF kernel function and set the value of  $\sigma$  equal to the mean pair-wise distance value between the positive training vectors. For the small datasets, we used the method in [24] by keeping the eigenvectors corresponding to all positive eigenvalues, while for the large datasets we used the method in [11] by setting the dimensionality of the resulted kernel subspace to  $L = 1000$ . On each class-specific problem, we ran five experiments by randomly selecting 70% of the positive and negative classes for training and the rest 30% for testing and measure the average performance value. The hyper-parameter values of all methods have been optimized by applying five-fold cross-validation on the training data. Specifically, for the methods performing subspace learning, the dimensionality of the obtained subspace was selected within the range of [1], [25] and for Support Vector Machine (SVM), Support Vector Regression (SVR) and Ridge Regression (RR) the regularization parameter value was selected in the range of  $10^r$ ,  $r = -3, \dots, 3$ . We used the publicly available software implementations from [8] for the support vector methods. We tested two variants of the proposed PCSDA, one in which the number of negative subclasses is fixed to  $K = 1$  (noted as PCSDA-1) and one in which the number of negative subclasses is automatically selected during the training process following the five-fold cross validation process from the set  $K = \{5, 10, 15, 20\}$  (noted as PCSDA-K). We opt to automatically determine the number of negative subclasses (and not set  $K$  equal to the number of the classes of the multi-class classification problem forming the negative class) to appropriately address

possible issues of class multi-modality. Indeed, by following such an approach, negative subclasses can be formed by samples belonging to more than one classes of the original multi-class classification problem, and multiple negative subclasses can be formed by samples belonging to the same class of the original multi-class classification problem. We should note here that the case  $K = 1$  is a subcase of the generic PCSDA-K.

For evaluating the performance of the methods in a ranking setting, we use the mean Average Precision (mAP) metric. For evaluating the performance of the methods in a classification setting, we use the  $f_1$  score defined as  $f_1 = 2 \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$ . Performance obtained for each method in a ranking and a classification setting is shown in Tables 2 and 3, respectively. The values of the Tables correspond to the mean performance value and the corresponding standard deviation over all class-specific problems of each dataset.

As can be seen in Table 2 for the ranking problems, the use of an unsupervised subspace learning method (PCA) leads to low performance. The use of regression models (RR and SVR) achieves high performance in most of the datasets, except Kanade and BU. The case is similar for the supervised subspace learning methods (LDA and PCSDA). It is interesting to see that in these two datasets PCSDA outperforms LDA by a margin of around 3%, while for the rest datasets their performance is similar. Focusing on PCSDA-1 and PCSDA-K, we can see that in most cases the use of one negative subclass seems to provide (slightly) better results, except for the Yale and Caltech101 datasets. We speculate that this is due to the homogeneity of the remaining datasets, however, we can see that the use of multiple subclasses can enhance performance for heterogeneous data.

Table 3 illustrates the performance of the methods in classification problems. Here we need to note that for performing classification using a subspace-based method (PCA and LDA) one needs to perform a two-step process, i.e. data projection to the subspace followed determined by the method followed by the application of a classification method. We combine PCA with SVM and LDA with Nearest Class Centroid (NCC) classifiers. The choice of LDA+NCC is based on the fact that the LDA optimization problem exploits as a class separability criterion that of NCC. Despite the fact that PCSDA is a subspace learning method, its probabilistic formulation leads to a classification rule (as detailed in Section III-B), and thus there is no need to perform a classification method for PCSDA. We can again see that the use of unsupervised subspace learning combined with SVM leads to low performance. Subspace learning-based classification leads to good performance, outperforming regression-based classification and SVM in most cases. Interestingly, the use of multiple negative subclasses leads to an improvement on Caltech101, Yale and OptDigits datasets of sizes 0.5%, 2% and 2.5%, respectively.

## V. CONCLUSIONS

In this paper we proposed a probabilistic model for class-specific discriminant subspace learning that is able to incorporate subclass information of the negative class in the optimization criterion. The adoption of a probability-based optimization criterion for class-specific discriminant subspace learning leads to a classifier defined on the data representations in the discriminant subspace. We showed that the parameters of the probabilistic model can be obtained by applying an efficient Spectral Regression-based process. Moreover, we showed that the proposed probabilistic model includes as special cases existing class-specific discriminant analysis methods. Experimental results illustrated the performance of the proposed model, in comparison with that of related methods, in both verification and classification problems. Interesting extensions of the proposed approach would be to devise class-specific criteria for Subspace Clustering [30], [32] and multi-view based subspace learning [3], [31].

## ACKNOWLEDGEMENT

This work has received partial funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant agreement No 871449 (OpenDR). This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

## REFERENCES

- [1] S. R. Arashloo and J. Kittler, "Class-specific kernel fusion of multiple descriptors for face verification using multiscale binarised statistical image features," *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 12, pp. 2100–2109, Dec. 2014.
- [2] D. Cai, X. He, and J. Han, "Spectral regression for efficient regularized subspace learning," in *Proc. IEEE 11th Int. Conf. Comput. Vis.*, Oct. 2007, pp. 1–7.
- [3] G. Cao, A. Iosifidis, K. Chen, and M. Gabbouj, "Generalized multi-view embedding for visual recognition and cross-modal retrieval," *IEEE Trans. Cybern.*, vol. 48, no. 9, pp. 2542–2555, Sep. 2018.
- [4] M. Cao, C. Chen, X. Hu, and S. Peng, "Towards fast and kernelized orthogonal discriminant analysis on person re-identification," *Pattern Recognit.*, vol. 94, pp. 218–229, Oct. 2019.
- [5] X.-W. Chen and T. Huang, "Facial expression recognition: A clustering-based approach," *Pattern Recognit. Lett.*, vol. 24, nos. 9–10, pp. 1295–1302, Jun. 2003.
- [6] D. Dua and C. Graff, "UCI machine learning repository," School Inf. Comput. Sci., Univ. California, Irvine, Irvine, CA, USA, Tech. Rep., 2017. [Online]. Available: [https://archive.ics.uci.edu/ml/citation\\_policy.html](https://archive.ics.uci.edu/ml/citation_policy.html)
- [7] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2nd ed. Hoboken, NJ, USA: Wiley, 2000.
- [8] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Jun. 2008.
- [9] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD, USA: Johns Hopkins Univ. Press, 1996.
- [10] L. H. Morsing, O. A. I. Sheikh-Omar, and A. Iosifidis, "Supervised domain adaptation: A graph embedding perspective and a rectified experimental protocol," 2020, *arXiv:2004.11262*. [Online]. Available: <http://arxiv.org/abs/2004.11262>
- [11] A. Iosifidis and M. Gabbouj, "Nyström-based approximate kernel subspace learning," *Pattern Recognit.*, vol. 57, pp. 190–197, Sep. 2016.
- [12] A. Iosifidis and M. Gabbouj, "Scaling up class-specific kernel discriminant analysis for large-scale face verification," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 11, pp. 2453–2465, Nov. 2016.
- [13] A. Iosifidis and M. Gabbouj, "Class-specific kernel discriminant analysis revisited: Further analysis and extensions," *IEEE Trans. Cybern.*, vol. 47, no. 12, pp. 4485–4496, Dec. 2017.
- [14] A. Iosifidis, M. Gabbouj, and P. Pekki, "Class-specific nonlinear projections using class-specific kernel spaces," in *Proc. IEEE Int. Conf. Big Data Sci. Eng.*, Aug. 2015, pp. 1–8.
- [15] A. Iosifidis, A. Tefas, and I. Pitas, "Activity-based person identification using fuzzy representation and discriminant learning," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 530–542, Apr. 2012.
- [16] A. Iosifidis, A. Tefas, and I. Pitas, "On the optimal class representation in linear discriminant analysis," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 9, pp. 1491–1497, Sep. 2013.
- [17] A. Iosifidis, A. Tefas, and I. Pitas, "Representative class vector clustering-based discriminant analysis," in *Proc. 9th Int. Conf. Intell. Inf. Hiding Multimedia Signal Process.*, Oct. 2013, pp. 526–529.
- [18] A. Iosifidis, A. Tefas, and I. Pitas, "Kernel reference discriminant analysis," *Pattern Recognit. Lett.*, vol. 49, pp. 85–91, Nov. 2014.
- [19] A. Iosifidis, A. Tefas, and I. Pitas, "Class-specific reference discriminant analysis with application in human behavior analysis," *IEEE Trans. Human-Machine Syst.*, vol. 45, no. 3, pp. 315–326, Jun. 2015.
- [20] Y. Jia, F. Nie, and C. Zhang, "Trace ratio problem revisited," *IEEE Trans. Neural Netw.*, vol. 20, no. 4, pp. 729–735, Apr. 2009.
- [21] T. Kanade, J. F. Cohn, and Y. Tian, "Comprehensive database for facial expression analysis," in *Proc. 4th IEEE Int. Conf. Automat. Face Gesture Recognit.*, Mar. 2000, pp. 46–53.
- [22] Y. P. Kittler, J. Li, and J. Matas, "Face verification using client specific Fisher faces," *Statist. Directions, Shapes Images*, pp. 63–66, 2000.
- [23] N. Kwak, "Nonlinear projection trick in kernel methods: An alternative to the kernel trick," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 12, pp. 2113–2119, Dec. 2013.
- [24] N. Kwak, "Implementing kernel methods incrementally by incremental nonlinear projection trick," *IEEE Trans. Cybern.*, vol. 47, no. 11, pp. 4003–4009, Nov. 2017.
- [25] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2006, pp. 2169–2178.
- [26] K.-C. Lee, J. Ho, and D. J. Kriegman, "Acquiring linear subspaces for face recognition under variable lighting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 5, pp. 684–698, May 2005.
- [27] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, "Coding facial expressions with Gabor wavelets," in *Proc. 3rd IEEE Int. Conf. Automat. Face Gesture Recognit.*, Apr. 1998, pp. 200–205.

- [28] A. M. Martinez and A. C. Kak, "PCA versus LDA," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 2, pp. 228–233, Feb. 2001.
- [29] A. M. Mood, F. A. Graybill, and D. C. Boes, *Introduction to the Theory of Statistics*, 3rd ed. New York, NY, USA: McGraw-Hill, 1974.
- [30] X. Peng, J. Feng, S. Xiao, W.-Y. Yau, J. T. Zhou, and S. Yang, "Structured autoencoders for subspace clustering," *IEEE Trans. Image Process.*, vol. 27, no. 10, pp. 5075–5086, Oct. 2018.
- [31] X. Peng, Z. Huang, J. Lv, H. Zhu, and J. T. Zhou, "Comic: Multi-view clustering without parameter selection," in *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97, 2019, pp. 5092–5101.
- [32] X. Peng, Z. Yu, Z. Yi, and H. Tang, "Constructing the L2-graph for robust subspace learning and subspace clustering," *IEEE Trans. Cybern.*, vol. 47, no. 4, pp. 1053–1066, Apr. 2017.
- [33] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 413–420.
- [34] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 1177–1184.
- [35] B. Scholkopf and A. J. Smola, *Learning With Kernels*. Cambridge, MA, USA: MIT Press, 2001.
- [36] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: <http://arxiv.org/abs/1409.1556>
- [37] L. S. Souza, B. B. Gatto, J.-H. Xue, and K. Fukui, "Enhanced Grassmann discriminant analysis with randomized time warping for motion recognition," *Pattern Recognit.*, vol. 97, Jan. 2020, Art. no. 107028.
- [38] D. T. Tran, M. Gabbouj, and A. Iosifidis, "Multilinear class-specific discriminant analysis," *Pattern Recognit. Lett.*, vol. 100, pp. 131–136, Dec. 2017.
- [39] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *J. Mach. Learn. Res.*, vol. 10, pp. 207–244, Feb. 2009.
- [40] H. Ye, Y. Li, C. Chen, and Z. Zhang, "Fast Fisher discriminant analysis with randomized algorithms," *Pattern Recognit.*, vol. 72, pp. 82–92, Dec. 2017.
- [41] L. Yin, X. Wei, Y. Sun, J. Wang, and M. J. Rosato, "A 3D facial expression database for facial behavior research," in *Proc. 7th Int. Conf. Automat. Face Gesture Recognit. (FGR)*, 2006, pp. 211–216.
- [42] W. Yu and C. Zhao, "Online fault diagnosis in industrial processes using multimodel exponential discriminant analysis algorithm," *IEEE Trans. Control Syst. Technol.*, vol. 27, no. 3, pp. 1317–1325, May 2019, doi: 10.1109/TCST.2017.2789188.
- [43] S. Zafeiriou, G. Tzimiropoulos, M. Petrou, and T. Stathaki, "Regularized kernel discriminant analysis with a robust kernel for face recognition and verification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 3, pp. 526–534, Mar. 2012.
- [44] M. Zhu and A. M. Martinez, "Subclass discriminant analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 8, pp. 1274–1286, Aug. 2006.



**ALEXANDROS IOSIFIDIS** (Senior Member, IEEE) received the B.Sc. degree in electrical and computer engineering and the M.Sc. degree with a specialization in mechatronics from the Democritus University of Thrace, Greece, in 2008 and 2010, respectively, and the Ph.D. degree in computer science from the Aristotle University of Thessaloniki, Greece, in 2014. Before he joined Aarhus University, Denmark, he held Postdoctoral Researcher positions with the Aristotle University of Thessaloniki and Tampere University of Technology, Finland, where he was an Academy of Finland Postdoctoral Research Fellow. He is currently an Associate Professor of Machine Learning with the Department of Engineering, Aarhus University. He has contributed to more than twenty R&D projects financed by EU, Finnish, as well as Danish funding agencies and companies. He has coauthored 75 articles in international journals and 91 papers in international conferences, proposing novel machine learning techniques and their application in a variety of problems. His research interests focus on topics of neural networks and statistical machine learning finding applications in computer vision, financial modeling, and graph mining problems. He has served as an Officer of the Finnish IEEE Signal Processing-Circuits and Systems Chapter (2016–2018). He is currently a member of the EURASIP Technical Area Committee on Visual Information Processing, and serves as an Area/Associate Editor for *Neurocomputing*, *Signal Processing: Image Communications*, IEEE Access, and *BMC Bioinformatics journals*.

•••

# Probabilistic Class-Specific Discriminant Analysis: Mathematical Analysis

Alexandros Iosifidis

## Abstract

This document provides the mathematical analysis supporting the paper ‘‘Probabilistic Class-Specific Discriminant Analysis’’.

## Definitions

Probabilistic Class-Specific Discriminant Analysis defines a subspace  $\mathbb{R}^d$ ,  $d \leq D$  of a feature space  $\mathbb{R}^D$  such that the positive class is optimally discriminated from the negative class, based on the assumption that the negative class is formed by multiple subclasses having the same cardinality. We show how to relax this assumptions in the next subsection.

Let us denote by  $\mathbf{x} \in \mathbb{R}^D$  a random variable, realizations of which correspond to samples of the positive and negative classes in our problem. PCSDA assumes there exists a positive class  $c_p$  following a multivariate Gaussian distribution defined by the mean vector  $\mathbf{m} \in \mathbb{R}^D$  and the covariance matrix  $\Phi_p \in \mathbb{R}^{D \times D}$ , i.e.:

$$P(\mathbf{x}|c_p) \sim N(\mathbf{x}|\mathbf{m}, \Phi_p). \quad (1)$$

Since the negative class  $c_n$  is formed by samples belonging to multiple classes (which are not distinguished by the labeling information available during training), PCSDA assumes that the negative class forms subclasses, each of which follows a multivariate Gaussian distribution in  $\mathbb{R}^D$ . Let  $\mathbf{y} \in \mathbb{R}^D$  be a random variable a realization of which corresponds to the mean vector of a negative subclass. Since in class-specific learning we are interested in maximizing the scatter of the negative data from the positive class (represented by  $\mathbf{m}$ ), we model the distribution of  $\mathbf{y}$  with respect to  $\mathbf{m}$  as a multivariate Gaussian distribution:

$$P(\mathbf{y}) \sim N(\mathbf{y}|\mathbf{m}, \Phi_n), \quad (2)$$

where  $\Phi_n$  is the corresponding covariance matrix expressing the scatter of the negative subclasses with respect to the positive class mean  $\mathbf{m}$ . Then, the distribution of the samples from the negative subclasses with respect to the positive class is expressed by the following multi-modal distribution:

$$P(\mathbf{x}|c_n) = \int N(\mathbf{y}|\mathbf{m}, \Phi_n) N(\mathbf{x}|\mathbf{y}, \Phi_w) d\mathbf{y}, \quad (3)$$

where  $\Phi_w$  is the (common) covariance matrix negative subclasses.

## Training phase

Given a set of positive i.i.d. samples  $\mathcal{S}_p = \{\mathbf{x}_1, \dots, \mathbf{x}_{N_p}\}$  the probability of correct assignment under our model is:

$$P(\mathcal{S}_p|c_p) = \prod_{i=1}^{N_p} P(\mathbf{x}_i|c_p). \quad (4)$$

Let us assume that negative class is formed by samples  $\mathbf{x}_i$ ,  $i = N_p + 1, \dots, N$  belonging to  $K$  subclasses, i.e.  $\mathcal{S}_n = \{\mathcal{S}_1, \dots, \mathcal{S}_K\}$ , each having a cardinality of  $M = N_n/K$ . The probability of assigning each of the negative samples to the corresponding subclass is given by:

$$P(\mathcal{S}_k|c_n) = \int N(\mathbf{y}|\mathbf{m}, \Phi_n) \prod_{\mathbf{x}_i \in \mathcal{S}_k} P(\mathbf{x}_i|\mathbf{y}, \Phi_w) d\mathbf{y}. \quad (5)$$

Without loss of generality, we assume that  $\mathbf{m} = \mathbf{0}$  (this can always be done by setting  $\mathbf{x}_i - \mathbf{m} \rightarrow \mathbf{x}_i$ ). Then, (4) and (5) become:

$$P(\mathcal{S}_p|c_p) = \prod_{i=1}^{N_p} P(\mathbf{x}_i) = \frac{1}{(2\pi)^{\frac{N_p D}{2}} |\Phi_p|^{\frac{N_p}{2}}} \exp\left(-\frac{1}{2} \sum_{i=1}^{N_p} \mathbf{x}_i^T \Phi_p^{-1} \mathbf{x}_i\right) = \frac{1}{(2\pi)^{\frac{N_p D}{2}} |\Phi_p|^{\frac{N_p}{2}}} \exp\left(-\frac{1}{2} \text{Tr}(\Phi_p^{-1} \mathbf{S}_p)\right), \quad (6)$$

where  $\mathbf{S}_p = \sum_{i=1}^{N_p} \mathbf{x}_i \mathbf{x}_i^T$  is the scatter matrix calculated using the positive samples with respect to the positive class mean vector  $\mathbf{m} = \mathbf{0}$ . Also (5) becomes:

$$\begin{aligned}
P(\mathcal{S}_k|c_n) &= \int N(\mathbf{y}|\mathbf{0}, \Phi_n) \prod_{\mathbf{x}_i \in \mathcal{S}_k} P(\mathbf{x}_i|\mathbf{y}, \Phi_w) d\mathbf{y} \\
&= \int \frac{1}{(2\pi)^{\frac{(M+1)D}{2}} |\Phi_n|^{\frac{1}{2}} |\Phi_w|^{\frac{M}{2}}} \exp\left(-\frac{1}{2} \mathbf{y}^T \Phi_n^{-1} \mathbf{y} - \frac{1}{2} \sum_{\mathbf{x}_i \in \mathcal{S}_k} (\mathbf{x}_i - \mathbf{y})^T \Phi_w^{-1} (\mathbf{x}_i - \mathbf{y})\right) d\mathbf{y} \\
&= \frac{1}{(2\pi)^{\frac{(M+1)D}{2}} |\Phi_n|^{\frac{1}{2}} |\Phi_w|^{\frac{M}{2}}} \exp\left(-\frac{1}{2} \mathbf{x}_i^T \Phi_w^{-1} \mathbf{x}_i\right) \int \exp\left(-\frac{1}{2} \mathbf{y}^T (\Phi_n^{-1} + M \Phi_w^{-1}) \mathbf{y} + \sum_{\mathbf{x}_i \in \mathcal{S}_k} \mathbf{x}_i^T \Phi_w^{-1} \mathbf{y}\right) d\mathbf{y} \\
&= \frac{1}{(2\pi)^{\frac{MD}{2}} |\Phi_n|^{\frac{1}{2}} |\Phi_w|^{\frac{M}{2}} |\Phi_n^{-1} + M \Phi_w^{-1}|^{\frac{1}{2}}} \\
&\quad \exp\left(-\frac{1}{2} \sum_{\mathbf{x}_i \in \mathcal{S}_k} \mathbf{x}_i^T \Phi_w^{-1} \mathbf{x}_i + \frac{1}{2} \left(\sum_{\mathbf{x}_i \in \mathcal{S}_k} \mathbf{x}_i^T \Phi_w^{-1}\right) (\Phi_n^{-1} + M \Phi_w^{-1})^{-1} \left(\Phi_w^{-1} \sum_{\mathbf{x}_i \in \mathcal{S}_k} \mathbf{x}_i\right)\right). \tag{7}
\end{aligned}$$

We split (7) in three terms:

$$\begin{aligned}
\frac{1}{(2\pi)^{\frac{MD}{2}} |\Phi_n|^{\frac{1}{2}} |\Phi_w|^{\frac{M}{2}} |\Phi_n^{-1} + M \Phi_w^{-1}|^{\frac{1}{2}}} &= \frac{1}{(2\pi)^{\frac{MD}{2}} |\Phi_n|^{\frac{1}{2}} |\Phi_w|^{\frac{M}{2}} |\Phi_n^{-1} (\Phi_n + \frac{1}{M} \Phi_w) \frac{1}{M} \Phi_w^{-1}|^{\frac{1}{2}}} \\
&= \frac{1}{(2\pi)^{\frac{MD}{2}} |\Phi_n|^{\frac{1}{2}} |\Phi_w|^{\frac{M}{2}} |\Phi_n|^{-\frac{1}{2}} M^{-\frac{D}{2}} |\Phi_w|^{-\frac{1}{2}} |\Phi_n + \frac{1}{M} \Phi_w|^{\frac{1}{2}}} \\
&= \frac{1}{M^{-\frac{D}{2}} (2\pi)^{\frac{MD}{2}} |\Phi_w|^{\frac{M-1}{2}} |\Phi_n + \frac{1}{M} \Phi_w|^{\frac{1}{2}}} \tag{8}
\end{aligned}$$

$$\begin{aligned}
\left(\sum_{\mathbf{x}_i \in \mathcal{S}_k} \mathbf{x}_i^T \Phi_w^{-1}\right) (\Phi_n^{-1} + M \Phi_w^{-1})^{-1} \left(\Phi_w^{-1} \sum_{\mathbf{x}_i \in \mathcal{S}_k} \mathbf{x}_i\right) &= M^2 \bar{\mathbf{x}}_k^T \Phi_w^{-1} (\Phi_n^{-1} + M \Phi_w^{-1})^{-1} \bar{\mathbf{x}}_k \\
&= M^2 \bar{\mathbf{x}}_k^T \Phi_w^{-1} \left(\frac{1}{M} \Phi_w - \frac{1}{M} \Phi_w (\Phi_n + \frac{1}{M} \Phi_w)^{-1} \Phi_w \frac{1}{M}\right) \Phi_w^{-1} \bar{\mathbf{x}}_k \\
&= M \bar{\mathbf{x}}_k^T \Phi_w^{-1} \Phi_w \Phi_w^{-1} \bar{\mathbf{x}}_k - \bar{\mathbf{x}}_k^T \Phi_w^{-1} \Phi_w (\Phi_n + \frac{1}{M} \Phi_w)^{-1} \Phi_w \Phi_w^{-1} \bar{\mathbf{x}}_k \\
&= M \bar{\mathbf{x}}_k^T \Phi_w^{-1} \bar{\mathbf{x}}_k - \bar{\mathbf{x}}_k^T (\Phi_n + \frac{1}{M} \Phi_w)^{-1} \bar{\mathbf{x}}_k \tag{9}
\end{aligned}$$

where  $\bar{\mathbf{x}}_k = \frac{1}{M} \sum_{\mathbf{x}_i \in \mathcal{S}_k} \mathbf{x}_i$  is the mean vector of the  $k$ -th negative subclass.

$$\begin{aligned}
-\frac{1}{2} \sum_{\mathbf{x}_i \in \mathcal{S}_k} \mathbf{x}_i^T \Phi_w^{-1} \mathbf{x}_i + \frac{1}{2} \left(\sum_{\mathbf{x}_i \in \mathcal{S}_k} \mathbf{x}_i^T \Phi_w^{-1}\right) (\Phi_n^{-1} + M \Phi_w^{-1})^{-1} \left(\Phi_w^{-1} \sum_{\mathbf{x}_i \in \mathcal{S}_k} \mathbf{x}_i\right) &= \\
-\frac{1}{2} \left(\sum_{\mathbf{x}_i \in \mathcal{S}_k} \mathbf{x}_i^T \Phi_w^{-1} \mathbf{x}_i - M \bar{\mathbf{x}}_k^T \Phi_w^{-1} \bar{\mathbf{x}}_k + \bar{\mathbf{x}}_k^T (\Phi_n + \frac{1}{M} \Phi_w)^{-1} \bar{\mathbf{x}}_k\right) &= \\
-\frac{1}{2} \left(\sum_{\mathbf{x}_i \in \mathcal{S}_k} (\mathbf{x}_i^T \Phi_w^{-1} \mathbf{x}_i - \bar{\mathbf{x}}_k^T \Phi_w^{-1} \bar{\mathbf{x}}_k) + \bar{\mathbf{x}}_k^T (\Phi_n + \frac{1}{M} \Phi_w)^{-1} \bar{\mathbf{x}}_k\right) &= \\
-\frac{1}{2} \left(\sum_{\mathbf{x}_i \in \mathcal{S}_k} Tr(\Phi_w^{-1} (\mathbf{x}_i - \bar{\mathbf{x}}_k) (\mathbf{x}_i - \bar{\mathbf{x}}_k)^T) + Tr((\Phi_n + \frac{1}{M} \Phi_w)^{-1} \bar{\mathbf{x}}_k \bar{\mathbf{x}}_k^T)\right) &= \\
-\frac{1}{2} \left(Tr\left(\Phi_w^{-1} \sum_{\mathbf{x}_i \in \mathcal{S}_k} (\mathbf{x}_i - \bar{\mathbf{x}}_k) (\mathbf{x}_i - \bar{\mathbf{x}}_k)^T\right) + Tr((\Phi_n + \frac{1}{M} \Phi_w)^{-1} \bar{\mathbf{x}}_k \bar{\mathbf{x}}_k^T)\right) &= \\
-\frac{1}{2} \left(Tr(\Phi_w^{-1} \mathbf{S}_w^{(k)}) + Tr((\Phi_n + \frac{1}{M} \Phi_w)^{-1} \mathbf{S}_n^{(k)})\right), \tag{10}
\end{aligned}$$

where  $\mathbf{S}_w^{(k)} = \sum_{\mathbf{x}_i \in \mathcal{S}_k} (\mathbf{x}_i - \bar{\mathbf{x}}_k) (\mathbf{x}_i - \bar{\mathbf{x}}_k)^T$  is the within-subclass scatter matrix of the  $k$ -th negative subclass and  $\mathbf{S}_n^{(k)} = \bar{\mathbf{x}}_k \bar{\mathbf{x}}_k^T$  is the scatter of the  $k$ -th negative subclass with respect to  $\mathbf{m} = \mathbf{0}$ .

Combining the above we get:

$$P(\mathcal{S}_k|c_n) = \frac{1}{M^{-\frac{D}{2}} (2\pi)^{\frac{MD}{2}} |\Phi_w|^{\frac{M-1}{2}} |\Phi_n + \frac{1}{M}\Phi_w|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\text{Tr}(\Phi_w^{-1}\mathbf{S}_w^{(k)})\right) \exp\left(-\frac{1}{2}\text{Tr}\left((\Phi_n + \frac{1}{M}\Phi_w)^{-1}\mathbf{S}_n^{(k)}\right)\right) \quad (11)$$

Since the assignment of the negative samples to subclasses is not provided by the labels used during training, we define them by applying a clustering method (e.g.  $K$ -Means) on the negative class vectors.

The parameters of the proposed PCSDA are the covariance matrices  $\Phi_p$ ,  $\Phi_n$  and  $\Phi_w$  defining the data generation processes for the positive and negative classes. These parameters are estimated by maximizing the (log) probability of correct assignment of the vectors  $\mathbf{x}_i$ ,  $i = 1, \dots, N$ :

$$L = \ln P(\mathcal{S}_p|c_p) + \ln P(\mathcal{S}_n|c_n), \quad (12)$$

where

$$\ln P(\mathcal{S}_p|c_p) = -\frac{N_p D}{2} \ln(2\pi) - \frac{N_p}{2} \ln |\Phi_p| - \frac{1}{2} \text{Tr}(\Phi_p^{-1} \mathbf{S}_p) \quad (13)$$

and

$$\begin{aligned} \ln P(\mathcal{S}_n|c_n) &= \sum_{k=1}^K \ln P(\mathcal{S}_k|c_n) \\ &= -\frac{D}{2} \sum_{k=1}^K \ln M - \frac{N_n D}{2} \ln(2\pi) - \frac{N_n - K}{2} \ln |\Phi_w| - \frac{K}{2} \ln |\Phi_n + \frac{1}{M}\Phi_w| \\ &\quad - \frac{1}{2} \sum_{k=1}^K \text{Tr}(\Phi_w^{-1} \mathbf{S}_w^{(k)}) - \frac{1}{2} \sum_{k=1}^K \text{Tr}\left((\Phi_n + \frac{1}{M}\Phi_w)^{-1} \mathbf{S}_n^{(k)}\right) \\ &= -\frac{D}{2} \sum_{k=1}^K \ln M - \frac{N_n D}{2} \ln(2\pi) - \frac{N_n - K}{2} \ln |\Phi_w| - \frac{K}{2} \ln |\Phi_n + \frac{1}{M}\Phi_w| \\ &\quad - \frac{1}{2} \text{Tr}(\Phi_w^{-1} \mathbf{S}_w) - \frac{1}{2} \text{Tr}\left((\Phi_n + \frac{1}{M}\Phi_w)^{-1} \mathbf{S}_n\right), \end{aligned} \quad (14)$$

where  $\mathbf{S}_n$  and  $\mathbf{S}_w$  are the total scatter matrices of the negative sub-classes, i.e.,  $\mathbf{S}_w = \sum_{k=1}^K \mathbf{S}_w^{(k)}$  and  $\mathbf{S}_n = \sum_{k=1}^K \mathbf{S}_n^{(k)}$ .

By substituting (13) and (14) in (12) the optimization problem takes the form:

$$\begin{aligned} L &= -\frac{D}{2} \sum_{k=1}^K \ln M - \frac{(N_p + N_n)D}{2} \ln(2\pi) - \frac{N_p}{2} \ln |\Phi_p| - \frac{1}{2} \text{Tr}(\Phi_p^{-1} \mathbf{S}_p) \\ &\quad - \frac{N_n - K}{2} \ln |\Phi_w| - \frac{K}{2} \ln |\Phi_n + \frac{1}{M}\Phi_w| - \frac{1}{2} \text{Tr}(\Phi_w^{-1} \mathbf{S}_w) - \frac{1}{2} \text{Tr}\left((\Phi_n + \frac{1}{M}\Phi_w)^{-1} \mathbf{S}_n\right). \end{aligned} \quad (15)$$

Setting the derivative of  $L$  w.r.t.  $\Phi_p$  to zero we get:

$$\frac{\partial L}{\partial \Phi_p} = \mathbf{0} \Rightarrow -\frac{N_p}{2} \Phi_p^{-1} + \frac{1}{2} \Phi_p^{-1} \mathbf{S}_p \Phi_p^{-1} = \mathbf{0} \Rightarrow \frac{1}{2} (\Phi_p^{-1} \mathbf{S}_p - N_p \mathbf{I}) \Phi_p^{-1} = \mathbf{0} \Rightarrow \Phi_p^{-1} \mathbf{S}_p = N_p \mathbf{I} \Rightarrow \Phi_p = \frac{1}{N_p} \mathbf{S}_p \quad (16)$$

Setting the derivative of  $L$  w.r.t.  $\Phi_n$  to zero we get:

$$\begin{aligned} \frac{\partial L}{\partial \Phi_n} = \mathbf{0} &\Rightarrow \frac{\partial}{\partial \Phi_n} \left( -\frac{K}{2} \ln |\Phi_n + \frac{1}{M}\Phi_w| - \frac{1}{2} \text{Tr}\left((\Phi_n + \frac{1}{M}\Phi_w)^{-1} \mathbf{S}_n\right) \right) = \mathbf{0} \\ &\Rightarrow \frac{\partial \mathbf{Q}}{\partial \Phi_n} \frac{\partial}{\partial \mathbf{Q}} \left( -\frac{K}{2} \ln |\mathbf{Q}| - \frac{1}{2} \text{Tr}(\mathbf{Q}^{-1} \mathbf{S}_n) \right) = \mathbf{0} \\ &\Rightarrow \left( -\frac{K}{2} \mathbf{Q}^{-1} + \frac{1}{2} \mathbf{Q}^{-1} \mathbf{S}_n \mathbf{Q}^{-1} \right) \mathbf{I} = \mathbf{0} \\ &\Rightarrow \left( -\frac{K}{2} (\Phi_n + \frac{1}{M}\Phi_w)^{-1} + \frac{1}{2} (\Phi_n + \frac{1}{M}\Phi_w)^{-1} \mathbf{S}_n (\Phi_n + \frac{1}{M}\Phi_w)^{-1} \right) \mathbf{I} = \mathbf{0} \\ &\Rightarrow \frac{1}{2} \left( (\Phi_n + \frac{1}{M}\Phi_w)^{-1} \mathbf{S}_n - K \mathbf{I} \right) (\Phi_n + \frac{1}{M}\Phi_w)^{-1} = \mathbf{0} \\ &\Rightarrow (\Phi_n + \frac{1}{M}\Phi_w)^{-1} \mathbf{S}_n - K \mathbf{I} = \mathbf{0} \Rightarrow (\Phi_n + \frac{1}{M}\Phi_w)^{-1} \mathbf{S}_n = K \mathbf{I} \\ &\Rightarrow \Phi_n + \frac{1}{M}\Phi_w = \frac{1}{K} \mathbf{S}_n \end{aligned} \quad (17)$$

Setting the derivative of  $L$  w.r.t.  $\Phi_w$  to zero we get:

$$\frac{\partial L}{\partial \Phi_w} = \mathbf{0} \Rightarrow \frac{\partial}{\partial \Phi_w} \left( -\frac{N_n - K}{2} \ln |\Phi_w| - \frac{K}{2} \ln |\Phi_n + \frac{1}{M} \Phi_w| - \frac{1}{2} \text{Tr}(\Phi_w^{-1} \mathbf{S}_w) - \frac{1}{2} \text{Tr} \left( (\Phi_n + \frac{1}{M} \Phi_w)^{-1} \mathbf{S}_n \right) \right) = \mathbf{0} \quad (18)$$

We split (18) in four terms:

$$\frac{\partial}{\partial \Phi_w} \left( \frac{(M-1)}{2} \ln |\Phi_w| \right) = \frac{(M-1)}{2} \Phi_w^{-1} \quad (19)$$

$$\frac{\partial}{\partial \Phi_w} \left( \frac{K}{2} \ln |\Phi_n + \frac{1}{M} \Phi_w| \right) = \frac{\partial \mathbf{Q}}{\partial \Phi_w} \frac{\partial}{\partial \mathbf{Q}} \left( \frac{K}{2} |\mathbf{Q}| \right) = \frac{K}{2} \mathbf{Q}^{-1} \frac{\partial \mathbf{Q}}{\partial \Phi_w} = \frac{K}{2M} (\Phi_n + \frac{1}{M} \Phi_w)^{-1} \quad (20)$$

$$\frac{\partial}{\partial \Phi_w} \left( \frac{1}{2} \text{Tr}(\Phi_w^{-1} \mathbf{S}_w) \right) = -\frac{1}{2} \Phi_w^{-1} \mathbf{S}_w \Phi_w^{-1} \quad (21)$$

$$\frac{\partial}{\partial \Phi_w} \left( \frac{1}{2} \text{Tr} \left( (\Phi_n + \frac{1}{M} \Phi_w)^{-1} \mathbf{S}_n \right) \right) = \frac{\partial \mathbf{Q}}{\partial \Phi_w} \frac{\partial}{\partial \mathbf{Q}} \left( \frac{1}{2} \text{Tr}(\mathbf{Q}^{-1} \mathbf{S}_n) \right) = -\frac{1}{2M} (\Phi_n + \frac{1}{M} \Phi_w)^{-1} \mathbf{S}_n (\Phi_n + \frac{1}{M} \Phi_w)^{-1} \quad (22)$$

Combining the above we get:

$$\frac{\partial L}{\partial \Phi_w} = \mathbf{0} \Rightarrow -(N_n - K) \Phi_w^{-1} - \frac{K}{M} (\Phi_n + \frac{1}{M} \Phi_w)^{-1} + \Phi_w^{-1} \mathbf{S}_w \Phi_w^{-1} + \frac{1}{M} (\Phi_n + \frac{1}{M} \Phi_w)^{-1} \mathbf{S}_n (\Phi_n + \frac{1}{M} \Phi_w)^{-1} = \mathbf{0} \quad (23)$$

Using (17) we get:

$$\begin{aligned} \frac{\partial L}{\partial \Phi_w} = \mathbf{0} &\Rightarrow -(N_n - K) \Phi_w^{-1} - \frac{K^2}{M} \mathbf{S}_n^{-1} + \Phi_w^{-1} \mathbf{S}_w \Phi_w^{-1} + \frac{K^2}{M} \mathbf{S}_n^{-1} \mathbf{S}_n \mathbf{S}_n^{-1} = \mathbf{0} \\ &\Rightarrow -(N_n - K) \Phi_w^{-1} + \Phi_w^{-1} \mathbf{S}_w \Phi_w^{-1} = \mathbf{0} \\ &\Rightarrow (\Phi_w^{-1} \mathbf{S}_w - (N_n - K) \mathbf{I}) \Phi_w^{-1} = \mathbf{0} \\ &\Rightarrow \Phi_w^{-1} \mathbf{S}_w = (N_n - K) \mathbf{I} \Rightarrow \mathbf{S}_w = (N_n - 1) \Phi_w \\ &\Rightarrow \Phi_w = \frac{1}{N_n - K} \mathbf{S}_w \end{aligned} \quad (24)$$

Note that when  $N_n = K$  (24) takes an indeterminate form (i.e. 0/0). However, in this case each negative subclass is formed by one sample and by definition  $\Phi_w = \mathbf{0}$ . This is discussed more in the next subsection.

Combining (25) and (24) we get:

$$\Phi_n = \frac{1}{K} \mathbf{S}_n - \frac{1}{M(N_n - K)} \mathbf{S}_w. \quad (25)$$

By combining (24) and (25) we can define the following matrix:

$$\Phi_O = \Phi_n + \Phi_w = \frac{1}{K} \mathbf{S}_n + \frac{1}{N_n} \mathbf{S}_w, \quad (26)$$

Thus, as can be seen from (16), (24), (25) and (26), the parameters of PCSDA can be calculated using  $\mathbf{S}_p$ ,  $\mathbf{S}_n$  and  $\mathbf{S}_w$  defined on the training vectors  $\mathbf{x}_i$ ,  $i = 1, \dots, N$ .

Using  $\mathbf{S}_p$ ,  $\mathbf{S}_n$  and  $\mathbf{S}_w$  a data transformation  $\mathbf{W} \in \mathbb{R}^{D \times D}$  can be obtained by optimizing for:

$$\mathbf{S}_n \mathbf{w} = \lambda (\mathbf{S}_p + \mathbf{S}_w) \mathbf{w}. \quad (27)$$

Problem (27) corresponds to the generalized eigen-analysis problem of the matrices  $\mathbf{S}_n$  and  $\mathbf{S}_I = \mathbf{S}_p + \mathbf{S}_w$ . Since the rank of  $\mathbf{S}_n$  is  $K$ , the dimensionality of the obtained subspace is restricted to  $d \leq \min(D, K)$ . Eigenvectors in PCSDA define a data transformation that minimizes the intra-class variance of the positive class and the intra-cluster variance of the negative subclasses, while at the same time maps the means of the negative subclasses as far as possible from the positive class mean. Since the above-described property is optimized by treating (27) as maximization problems, the eigenvectors forming  $\mathbf{W}$  are sorted in a decreasing order of the corresponding eigenvalues.

After obtaining  $\mathbf{W}$ , the intra-class and out-of-class covariance matrices of the transformed data  $\mathbf{z}_i = \mathbf{W}^T \mathbf{x}_i$  are given by:

$$\tilde{\Phi}_p = \mathbf{W}^T \Phi_p \mathbf{W} \quad \text{and} \quad \tilde{\Phi}_O = \mathbf{W}^T \Phi_O \mathbf{W}. \quad (28)$$

In the above we set the assumption that the number of samples forming the negative subclasses is equal. This assumption can be relaxed following one of the following approaches. After assigning all negative samples to subclasses and calculating the negative subclass distributions, one can sample  $M$  vectors from each distribution. Alternatively, one can calculate the total within-subclass matrix of the negative class by  $\mathbf{S}_w = \sum_{k=1}^K \mathbf{S}_w^{(k)}$ . The latter approach is commonly used in multi-class discriminant analysis variants. Note that for the model's parameters calculation, only the overall scatter matrices  $\mathbf{S}_p$ ,  $\mathbf{S}_w$  and  $\mathbf{S}_n$  are used.

### Spectral Regression-based solution of PCSDA

We further show in the following that PCSDA can be efficiently solved by following a spectral regression based process. Let us denote by  $\mathbf{1}_p \in \mathbb{R}^N$  a vector having ones in the elements corresponding to the positive training vectors and zeros elsewhere. In the same manner, we define the vector  $\mathbf{1}_k \in \mathbb{R}^N$  as a vector having ones in the elements corresponding to the negative samples belonging to the  $k$ -th subclass and zeros elsewhere. We also define by  $\mathbf{J}_p \in \mathbb{R}^{N \times N}$  and  $\mathbf{J}_k \in \mathbb{R}^{N \times N}$  the diagonal matrices having in their diagonal the vectors  $\mathbf{1}_p$  and  $\mathbf{1}_k$ , respectively. Then,  $\mathbf{S}_n$  and  $\mathbf{S}_I$  can be expressed as:

$$\mathbf{S}_n = \sum_{k=1}^K (\bar{\mathbf{x}}_k - \mathbf{m})(\bar{\mathbf{x}}_k - \mathbf{m})^T = \mathbf{X}\mathbf{L}_n\mathbf{X}^T, \quad (29)$$

$$\mathbf{S}_I = \mathbf{S}_p + \mathbf{S}_w = \mathbf{X}\mathbf{L}_I\mathbf{X}^T \quad (30)$$

where

$$\mathbf{L}_n = \sum_{k=1}^K \left( \frac{1}{N_k^2} \mathbf{1}_k \mathbf{1}_k^T - \frac{1}{N_k N_p} (\mathbf{1}_k \mathbf{1}_p^T + \mathbf{1}_p \mathbf{1}_k^T) + \frac{1}{N_p^2} \mathbf{1}_p \mathbf{1}_p^T \right) \quad (31)$$

$$\mathbf{L}_I = \left( \mathbf{J}_p - \frac{1}{N_p} \mathbf{1}_p \mathbf{1}_p^T \right) + \sum_{k=1}^K \left( \mathbf{J}_k - \frac{1}{N_k} \mathbf{1}_k \mathbf{1}_k^T \right). \quad (32)$$

Substituting (29) and (30) in (27) and setting  $\mathbf{v} = \mathbf{X}^T \mathbf{w}$ :

$$\mathbf{L}_n \mathbf{v} = \lambda \mathbf{L}_I \mathbf{v}. \quad (33)$$

Thus, the vectors  $\mathbf{v}$  are the eigenvectors of the matrix  $\mathbf{L} = \tilde{\mathbf{L}}_w^{-1} \mathbf{L}_n$ , where  $\tilde{\mathbf{L}}_w$  is a regularized version of  $\mathbf{L}_w$ , i.e.  $\tilde{\mathbf{L}}_w = \mathbf{L}_w + \epsilon \mathbf{I}$  where  $\epsilon > 0$ . It can be shown that the matrix  $\mathbf{L}$  is a block matrix formed by blocks the elements of which are indicated by the class labels of the positive data and the cluster labels of the negative data. The top  $K$  eigenvectors of it are also formed by blocks indicated by the same labels. After the determination of the matrix  $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_K]$ , the solution of (27) is given by  $\mathbf{W} = (\mathbf{X}^T)^\dagger \mathbf{V}$ , where  $(\mathbf{X}^T)^\dagger$  is the pseudo-inverse of  $\mathbf{X}^T$ .

### Test phase

After the estimation of the model's parameters, a new sample represented by the vector  $\mathbf{x}^*$  can be evaluated. The posterior probabilities of the positive and negative classes are given by:

$$P(c_p | \mathbf{x}^*) = \frac{p(\mathbf{x}^* | c_p) P(c_p)}{p(\mathbf{x}^*)} \quad \text{and} \quad P(c_n | \mathbf{x}^*) = \frac{p(\mathbf{x}^* | c_n) P(c_n)}{p(\mathbf{x}^*)}. \quad (34)$$

The a priori class probabilities  $P(c_p)$  and  $P(c_n)$  can be calculated by the proportion of the positive and negative samples in the training phase, i.e.  $P(c_p) = N_p/N$  and  $P(c_n) = N_n/N$ . Depending on the problem at hand, it may be sometimes preferable to consider equiprobable classes, i.e.  $P(c_p) = P(c_n) = 1/2$ , leading to the maximum likelihood classification case. The class-conditional probabilities of the transformed sample  $\mathbf{z}^* = \mathbf{W}^T \mathbf{x}^*$  are given by:

$$\begin{aligned} p(\mathbf{x}^* | c_n) &= \int \frac{1}{(2\pi)^D |\Phi_n|^{1/2} |\Phi_w|^{1/2}} \exp \left( -\frac{1}{2} \mathbf{y}^T \Phi_n^{-1} \mathbf{y} - \frac{1}{2} (\mathbf{x}^* - \mathbf{y})^T \Phi_w^{-1} (\mathbf{x}^* - \mathbf{y}) \right) d\mathbf{y} \\ &= \frac{1}{(2\pi)^D |\Phi_n|^{1/2} |\Phi_w|^{1/2}} \exp \left( -\frac{1}{2} \mathbf{x}^{*T} \Phi_w^{-1} \mathbf{x}^* \right) \int \exp \left( -\frac{1}{2} \mathbf{y}^T (\Phi_n^{-1} + \Phi_w^{-1}) \mathbf{y} + \mathbf{x}^{*T} \Phi_w^{-1} \mathbf{y} \right) d\mathbf{y} \\ &= \frac{(2\pi)^{\frac{D}{2}}}{(2\pi)^D |\Phi_n|^{1/2} |\Phi_w|^{1/2} |\Phi_n^{-1} + \Phi_w^{-1}|^{1/2}} \exp \left( -\frac{1}{2} \mathbf{x}^{*T} \Phi_w^{-1} \mathbf{x}^* + \frac{1}{2} \mathbf{x}^{*T} \Phi_w^{-1} (\Phi_n^{-1} + \Phi_w^{-1})^{-1} \Phi_w^{-1} \mathbf{x}^* \right) \\ &= \frac{1}{(2\pi)^{\frac{D}{2}} |\Phi_n + \Phi_w|^{1/2}} \exp \left( -\frac{1}{2} \mathbf{x}^{*T} \Phi_w^{-1} \mathbf{x}^* + \frac{1}{2} \mathbf{x}^{*T} \Phi_w^{-1} \mathbf{x}^* - \frac{1}{2} \mathbf{x}^{*T} (\Phi_n + \Phi_w)^{-1} \mathbf{x}^* \right) \\ &= \frac{1}{(2\pi)^{\frac{D}{2}} |\Phi_O|^{1/2}} \exp \left( -\frac{1}{2} \mathbf{x}^{*T} \Phi_O^{-1} \mathbf{x}^* \right) = \frac{1}{(2\pi)^{\frac{D}{2}} |\tilde{\Phi}_O|^{1/2}} \exp \left( -\frac{1}{2} \mathbf{z}^{*T} \tilde{\Phi}_O^{-1} \mathbf{z}^* \right) \end{aligned} \quad (35)$$

and

$$p(\mathbf{x}^* | c_p) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Phi_p|^{1/2}} \exp \left( -\frac{1}{2} \mathbf{x}^{*T} \Phi_p^{-1} \mathbf{x}^* \right) = \frac{1}{(2\pi)^{\frac{D}{2}} |\tilde{\Phi}_p|^{1/2}} \exp \left( -\frac{1}{2} \mathbf{z}^{*T} \tilde{\Phi}_p^{-1} \mathbf{z}^* \right). \quad (36)$$

In the above, we used the orthogonal property of the matrix  $\mathbf{W}$ . In the case where we want  $\mathbf{z}^* \in \mathbb{R}^d$ ,  $d < D$ , we keep the dimensions of  $\mathbf{z}^*$  corresponding to the first  $d$  columns of  $\mathbf{W}$ .

Combining (34)-(35) the ratio of class posterior probabilities is equal to:

$$\lambda(\mathbf{z}^*) = \frac{P(c_p) |\tilde{\Phi}_O|^{\frac{1}{2}} \exp\left(-\frac{1}{2} \mathbf{z}^{*T} \tilde{\Phi}_p^{-1} \mathbf{z}^*\right)}{P(c_n) |\tilde{\Phi}_p|^{\frac{1}{2}} \exp\left(-\frac{1}{2} \mathbf{z}^{*T} \tilde{\Phi}_O^{-1} \mathbf{z}^*\right)}. \quad (37)$$

$\lambda(\mathbf{z}^*)$  can be used to classify  $\mathbf{z}^*$  to the positive class when  $\lambda(\mathbf{z}^*) > 1$  and to the negative class otherwise. Alternatively, (37) can be also used to define the classification rule:

$$g(\mathbf{z}^*) = \ln P(c_p) - \ln P(c_n) + \frac{1}{2} \ln |\tilde{\Phi}_O| - \frac{1}{2} \ln |\tilde{\Phi}_p| - \frac{1}{2} \mathbf{z}^{*T} \tilde{\Phi}_p^{-1} \mathbf{z}^* + \frac{1}{2} \mathbf{z}^{*T} \tilde{\Phi}_O^{-1} \mathbf{z}^* \quad (38)$$

classifying  $\mathbf{z}^*$  to the positive class if  $g(\mathbf{z}^*) > 0$  and to the negative class otherwise.

In class-specific ranking settings one can follow the process applied when using the standard CSDA approach. First, test vectors  $\mathbf{x}_j^*$  are mapped to the discriminant subspace  $\mathbf{z}_j^* = \mathbf{W}^T \mathbf{x}_j^*$  and  $\mathbf{z}_j^*$ ,  $i = 1, \dots, N$  are obtained. Then,  $\mathbf{z}_j^*$ 's are ordered based on their distance w.r.t. the positive class mean  $d_j = \|\mathbf{z}_j^* - \boldsymbol{\mu}\|_2$ , where  $\boldsymbol{\mu}$  is the mean of positive training samples in  $\mathbb{R}^d$ .

## **7.16 Deepbots: A Webots-based Deep Reinforcement Learning Framework for Robotics**

The appended paper follows.

# Deepbots: A Webots-based Deep Reinforcement Learning Framework for Robotics

M. Kirtas, K. Tsampazis, N. Passalis, and A. Tefas

Artificial Intelligence and Information Analysis Lab  
Department of Informatics, Aristotle University of Thessaloniki, Greece  
{`eakirtas`, `tsampaka`, `passalis`, `tefas`}@`csd.ath.gr`

**Abstract.** Deep Reinforcement Learning (DRL) is increasingly used to train robots to perform complex and delicate tasks, while the development of realistic simulators contributes to the acceleration of research on DRL for robotics. However, it is still not straightforward to employ such simulators in the typical DRL pipeline, since their steep learning curve and the enormous amount of development required to interface with DRL methods significantly restrict their use by researchers. To overcome these limitations, in this work we present an open-source framework that combines an established interface used by DRL researchers, the OpenAI Gym interface, with the state-of-the-art Webots robot simulator in order to provide a standardized way to employ DRL in various robotics scenarios. Deepbots aims to enable researchers to easily develop DRL methods in Webots by handling all the low-level details and reducing the required development effort. The effectiveness of the proposed framework is demonstrated through code examples, as well as using three use cases of varying difficulty.

**Keywords:** Deep Reinforcement Learning · Simulation Environment · Webots · Deepbots

## 1 Introduction

Reinforcement Learning (RL) is a domain of Machine Learning, and one of the three basic paradigms alongside supervised and unsupervised learning. RL employs agents that *learn* by simultaneously *exploring* their environment and *exploiting* the already acquired knowledge to solve the task at hand. The learning process is guided by a reward function, which typically expresses how close the agent is to reaching the desired target behavior. In recent years, Deep Learning (DL) [8] was further combined with RL to form the field of Deep Reinforcement Learning (DRL) [17], where powerful DL models were used to solve challenging RL problems.

DRL is also increasingly used to train robots to perform complex and delicate tasks. Despite the potential of DRL on robotics, such approaches usually require an enormous amount of time to sufficiently explore the environment and manage to solve the task, often suffering from low sample efficiency [20].

Furthermore, during the initial stages of training, the agents take actions at random, potentially endangering the robot’s hardware. To circumvent these restrictions, researchers usually first run training sessions on realistic simulators, such as Gazebo [11], and OpenRAVE [4], where the simulation can run at accelerated speeds and with no danger, only later to transfer the trained agents on physical robots. However, this poses additional challenges [5], due to the fact that simulated environments provide a varying degree of realism, so it is not always possible for the agent to observe and act exactly as it did during training in the real world. This led to the development of more realistic simulators, which further reduce the gap between the simulation and the real world, such as Webots [15], and Actin [1]. It is worth noting that these simulators not only simulate the physical properties of an environment and provide a photorealistic representation of the world, but also provide an easily parameterizable environment, which can be adjusted according to the needs of every different real life scenarios.

Even though the aforementioned simulators provide powerful tools for developing and validating various robotics applications, it is not straightforward to use them for developing DRL methods, which typically operate over a higher level of abstraction that hides low-level details, such as how the actual control commands are processed by the robots. This limits their usefulness for developing DRL methods, since their steep learning curve and the enormous amount of development required to interface with DRL methods, considerably restricts their use by DRL researchers.

The main contribution of this work is to provide an open-source framework that can overcome the aforementioned limitations, supplying a DRL interface that is easy for the DRL research community to use. More specifically, the developed framework, called “deepbots”, combines the well known OpenAI Gym [3] interface with the Webots simulator in order to establish a standard way to employ DRL in real case robotics scenarios. Deepbots aims to enable researchers to use RL in Webots and it has been created mostly for educational and research purposes. In essence, deepbots acts as a middle-ware between Webots and the DRL algorithms, exposing a Gym style interface with multiple levels of abstraction. The framework uses design patterns to achieve high code readability and reusability, allowing to easily incorporate it in most research pipelines. The aforementioned features come as an easy-to-install Python package that allows developers to efficiently implement environments that can be utilized by researchers or students to use their algorithms in realistic benchmarking. At the same time, deepbots provides ready-to-use standardized environments for well-known problems. Finally, the developed framework provides some extra tools for monitoring, e.g., tensorboard logging and plotting, allowing to directly observe the training progress. Deepbots is available at <https://github.com/aidudezz/deepbots>.

The paper is structured as follows. First, Section 2 provides a brief overview of existing tools and simulators that are typically used for training DRL algorithms and highlights the need for providing a standardized DRL framework over the simulators to lower the barrier for accessing these tools by DRL researchers.

Then, a detailed description of deepbots is provided in Section 3, while a set of already implemented examples, along with results achieved by well-established baseline RL algorithms, are provided in Section 4. Finally, Section 5 concludes this paper.

## 2 Related Work

First, the well-established OpenAI Gym toolkit, as well as the Webots simulator, are briefly introduced. Then, a number of related frameworks are also discussed and compared to the proposed deepbots framework.

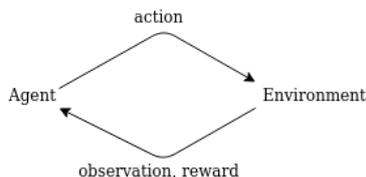


Fig. 1: OpenAI Gym interface

The OpenAI Gym, or simply “Gym”, framework has been established as the standard interface between the actual simulator and RL algorithm [3]. According to the OpenAI Gym documentation, the framework follows the classic “agent-environment loop”, as shown in Fig. 1, and it defines a set of environments. An environment for each step receives an action from the agent and returns a new observation and reward for the action. This procedure is repeated and separated in an episodic format. Except that, Gym standardizes a collection of testing environments for RL benchmarking. Even though OpenAI Gym is an easy-to-use tool to demonstrate the capabilities of RL in practice, it comes only with toy, unrealistic and difficult to extend scenarios. It needs several external dependencies to build more complex environments, like the MuJoCo simulator [21], which is a proprietary piece of software, which barriers its use and ability to be extended.

As RL problems become more and more sophisticated, researchers have to come up with even more complicated simulations. Self-driving cars, multi-drone scenarios, and other tasks with many more degrees of freedom synthesize the new big picture of RL research. Consequently, that leads to the need of even more accurate and realistic simulations, such as Webots [16], which is a free and open-source 3D robot simulator. Webots provides customizable environments, the ability to create robots from scratch, as well as high fidelity simulations with realistic graphics and is also Robot Operating System (ROS) compliant. It comes preloaded with several well-known robots, e.g., E-puck [7], iCub [14], etc. Robots can be wheeled or legged and use actuators like robotic arms, etc. An array of sensors is also provided, e.g., lidars, radars, proximity sensors, light sensors, touch sensors, GPS, accelerometers, cameras, etc. These capabilities allow it to cover a wide range of different applications. Robots are programmed via controllers that can be written in several languages (C, C++, Python, Java and MATLAB). However, even though Webots can be used for DRL, it comes

with a set of limitations. The mechanisms which are used to run the different scripts are not friendly for those with DRL background, requiring a significant development overhead for supporting DRL algorithms, while there is no standardization regarding the way DRL methods are developed and interface with Webots.

Note that there is also an increasing number of works that attempt to formalize and facilitate the usage of RL in robotic simulators. However, none of these works target the state-of-the-art Webots simulator. For example, Gym-Ignition [6] is a work which aims to expose an OpenAI Gym interface to create reproducible robot environments for RL research. The framework has been designed for the Gazebo simulator and provides interconnection to external third party software, multiple physics and rendering engines, distributed simulation capabilities and it is ROS compliant. Other than that, [22] extends the Gym interface with ROS compliance and it uses the Gazebo simulator as well. The latest version of this work [13] is compatible with ROS 2 and is extended and applied in more real world oriented examples. All of these works are limited by the low quality graphics provided by the Gazebo simulator, rendering them less useful for DRL algorithms that rely on visual input. Finally, Isaac Gym [9] is a powerful software development toolkit providing photorealistic rendering, parallelization and is packed as a unified framework for DRL and robotics. However, its closed source nature can render it difficult to use, especially on scenarios that deviate from its original use cases. To the best of our knowledge this is the first work which provide a generic OpenAI Gym interface for Webots, standardizing the way DRL algorithms interface with Webots and provide easy access to a state-of-the-art simulator.

### 3 Deepbots

Deepbots follows the same agent-environment loop as the OpenAI Gym framework, with the only difference being that the agent, which is responsible for choosing an action, runs on the supervisor and the observations are acquired by the robot. This master-minion protocol is not problem-specific and thus has the advantage of generalization, due to the fact that it can be used in more than one examples. That makes it easier to construct various use cases and utilize them as benchmarks. In this way, the deepbots framework acts as a wrapper, meaning that it wraps up and hides certain operations from the users, so that they are able to focus on the DRL task, rather than handling all the technical simulator-specific details. At the same time, deepbots also enriches the training pipeline with live monitoring features, which helps researchers get early observations about the fundamental parts of the training process. All these features contribute into providing a powerful DRL-oriented abstraction over Webots, allowing researchers to quickly model different use cases and simulation environments, as well as employ them to develop sophisticated DRL algorithms.

Before describing deepbots in detail, it is useful to briefly review the way Webots handles various simulation tasks. Webots represents scenes with a tree

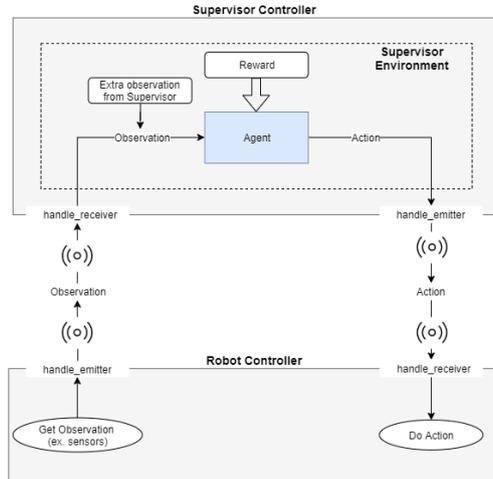


Fig. 2: Deepbots supervisor-controller communication

structure in which the root node is the world and its children nodes are the different items in the world. Consequently, a robot should be a node under the root node which contains a controller. Controllers are scripts responsible for the node's functionality. A robot for example has a controller in order to read values from sensors and act accordingly. For RL purposes, it is necessary to include a special supervisor node which has full knowledge of the world and it can get information for and modify every other node. For example, the supervisor can be used to move items in the world or measure distances between a robot and a target.

With respect to the aforementioned logic, deepbots gives the ability to easily construct DRL tasks with minimal effort. The basic structure of deepbots communication scheme is depicted in Fig. 2, where the supervisor controller is the script of the supervisor and the robot controller is the script of the robot. The communication between supervisor and robot is achieved via emitters/receivers, which broadcast and receive messages respectively. Without loss of generality, the supervisor is a node without mass or any physical properties in the simulation. For example, in a real case scenario, a supervisor could be a laptop which transmits actions to the robot, but without interacting with the actual scene. Furthermore, the emitter and the receiver could be any possible device, either cable or wireless, properly set up for this task.

Deepbots works as follows: first of all the simulator has to be reset in the initial state. On the one hand, the robot collects the first set of observations and by using its emitter sends the information to the supervisor node. On the other hand, the supervisor receives the observations with its receiver component and in turn passes them to the agent. In this step, if needed, the supervisor can augment the observation with extra information, e.g., Euclidean distances with

respect to some ground truth objects, which are unavailable to the robot and its sensors. Except for the observation, the supervisor can pass the previous action, reward and/or any other state information to the agent. It should be mentioned that deepbots is not bound to any DL framework and the agent can be written in any Python-based DL framework, e.g., PyTorch, TensorFlow, Numpy, etc. After that, the agent produces an action which is going to be transmitted to the robot via the emitter. Finally, the robot has to perform the action which was received, with its actuators. The aforementioned procedure is performed iteratively until the robot achieves its objective or for a certain number of epochs/episodes, or whatever condition is needed by the use case.

---

```

1 class FindTargetSupervisor(SupervisorEmitterReceiver):
2     def get_observation(self):...
3     def get_reward(self, action):...
4     def is_done(self):...
5     def reset(self):...
6     def step(self, action):...
7
8     env = FindTargetSupervisor()
9     env = TensorboardLogger(env)
10    agent = DDPG(...)
11    for i in range(EPOCHS):
12        done = False
13        score = 0
14        obs = env.reset()
15        while not done:
16            act = agent.choose_action(obs)
17            obs, reward, done, info = env.step(act)
18            agent.remember(obs, action, reward, done)
19            agent.learn()
20            score += reward

```

---

Code Example 1.1: Supervisor controller code example

In order to implement an agent, the user has to implement two scripts at each side of the communication channel and the framework handles the details. On the supervisor side, the user has to create a Gym environment with the well known abstract methods and train/evaluate the DRL model, as shown in Code Example 1.1. While on the other side, a simple script has to be written for reading values from sensors and translating messages to the actual values needed by the actuators. A typical script for this task is shown in Code Example 1.2. The deepbots framework runs all the essential operations needed by the simulation, executes the step function and handles the communication between the supervisor and the robot controller in the background. In addition, by following the framework workflow, cleaner code is achieved, while the agent logic is separated from the actuator manipulation and it is closer to the physical cases. Furthermore, the framework logic is similar to ROS and can be integrated with it with minimal effort.

```

1 class FindTargetRobot(RobotEmitterReceiver):
2     def create_message(self):
3         message = []
4         for rangefinder in self.rangefinders:
5             message.append(rangefinder.value())
6         return message
7     def use_message_data(self, message):
8         gas = float(message[1])
9         wheel = float(message[0])
10        ...
11 robot_controller = FindTargetRobot()
12 robot_controller.run()

```

Code Example 1.2: Robot controller code example

As the deepbots framework mostly aims to be a user-friendly tool for educational and research purposes, it has different levels of abstraction. An overview of the abstraction level class diagram of deepbots is provided in Fig. 3. For example, users can choose if they would use JSON emitters and receivers or if they want to go on with an implementation from scratch. At the top of the abstraction hierarchy is the *SupervisorEnv*, which is the OpenAI Gym abstract class. Below that level is the actual implementation which resolves the communication between the supervisor and the robot. Similarly, the robot has also different levels of abstraction. A user can choose among certain types of message formats to transmit actions and observations. Extra features can be added to the framework as decorator classes by implementing the OpenAI Gym interface, as demonstrated in line 9 of Code Example 1.1. This design pattern could be used to stack different controls, monitoring and other functionalities.

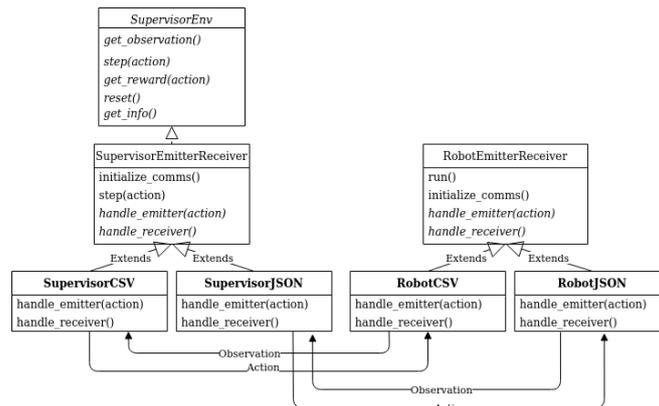


Fig. 3: Abstraction level class diagram

## 4 Example Environments

Deepbots contains a collection of ready-to-use environments, which showcase uses of the framework in toy or complicated examples. On the one hand, the community can contribute new environments and use cases to enrich the existing collection. On the other hand, this collection can be used by researchers to benchmark RL algorithms in Webots. Three environments of varying complexity are presented in this Section.

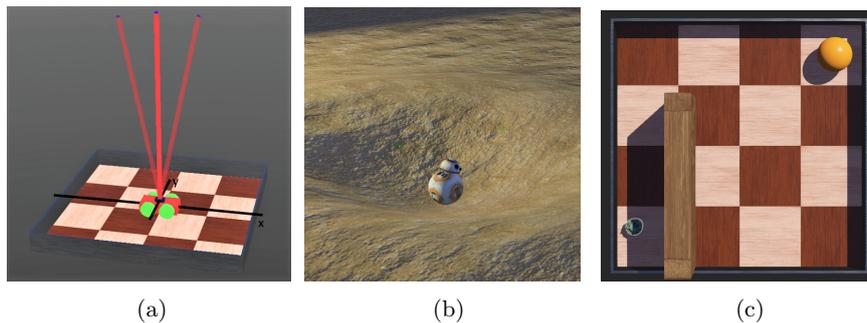


Fig. 4: (a) CartPole: the x axis is the cart motion axis, the y axis is the pole rotation axis, (b) PitEscape: the robot inside the pit, (c) FindBall: the robot searching for the yellow ball

### 4.1 CartPole

The CartPole example is based on the problem described in [2] and adapted to Webots. In the world exists an arena, and a small four wheeled cart that has a long pole connected to it by a free hinge, as shown in Fig. 4. The hinge contains a sensor to measure the angle the pole has off vertical. The pole acts as an inverted pendulum and the goal is to keep it vertical by moving the cart forward and backward. This task is tackled with the discrete version of the Proximal Policy Optimization (PPO) RL algorithm [19].

The observation contains the cart position and velocity on the x axis, the pole angle off vertical and the pole velocity at its tip. The action space is discrete containing two possible actions for each time step, move forward or move backward. For every step taken, the agent is rewarded with +1 including the termination step. Each episode is terminated after a maximum 200 steps or earlier if the pole has fallen  $\pm 15$  degrees off vertical or the cart has moved more than  $\pm 0.39$  meters on the x axis. To consider the task solved, the agent has to achieve an average score of over 195.0 in 100 consecutive episodes.

The learning curve using the PPO algorithm, as well as the average action probability over the training process are depicted in Fig. 5. The actor and critic consist of small two-layered neural networks with 10 ReLU neurons on each layer and the agent was able to solve the problem after running for a simulated time of about 2.5 hours.

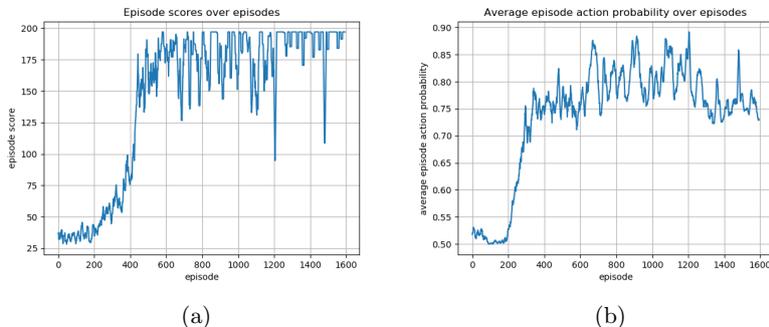


Fig. 5: CartPole: (a) reward accumulated for each episode, and (b) average probability of selected actions per episode

## 4.2 Pit Escape

The Pit Escape example that can be seen in Figure 4 is a problem taken from *robotbenchmark*<sup>1</sup>.

The Pit Escape world comprises of a pit with a radius of 2.7, where inside it lies a BB-8 robot with a spherical body that acts as a wheel [16]. The robot contains a pitch and a yaw motor and can move by rolling forward or backward (pitch) or by turning left and right (yaw). The task is to escape from the pit, but its motors are not powerful enough to escape by just moving forward, so it needs to move in some other way. This task is also tackled with the discrete version of the PPO RL algorithm [19].

This problem is very similar to the Mountain Car one [18], but in three dimensions and has more complex observation and action spaces. The robot contains a gyroscope and an accelerometer which provide the observation. Thus, the observation contains the robot orientation and acceleration in the x, y, z axes, i.e., a total of 6 values. The action space is discrete containing 4 possible actions for each time step. With each action the robot can set its motor speeds to their maximum or minimum values. Each episode lasts 60 seconds and the reward function is based on a metric  $M$ :

$$M = \begin{cases} 0.5 \frac{d}{R} & d < R \\ 0.5 + 0.5 \frac{T-t}{T} & d > R \end{cases}, \quad (1)$$

where  $d$  is the maximum distance achieved from the center of the pit until now in the episode,  $R$  is the radius of the pit,  $T$  is the maximum time allowed per episode (60 seconds), and  $t$  is the time until now in the episode.  $M$  only changes when a higher distance from the center is achieved during the episode. For each time step, based on the change between the previous step and current step metrics, the reward  $R_i$  for step  $i$  is calculated as  $R_i = M_{old} - M$ , where  $M_{old}$  is the previous step metric and  $M$  is the current step metric. An episode terminates

<sup>1</sup> robotbenchmark, <https://robotbenchmark.net>

after 60 seconds or if the robot has escaped the pit, which is calculated by the distance between the robot and the pit center.

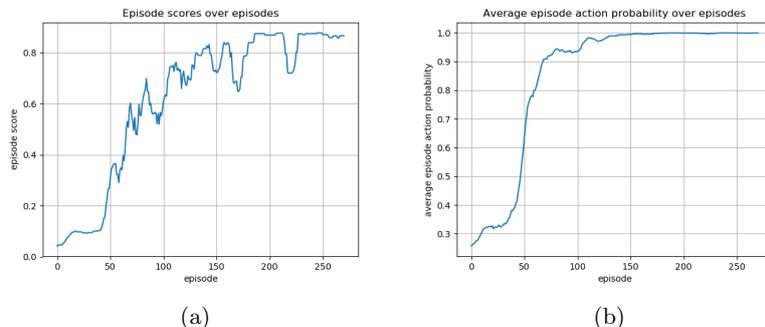


Fig. 6: Pit Escape: (a) reward accumulated for each episode, and (b) average probability of selected actions per episode

Two-layered networks with 60 ReLU neurons on each layer were used for the actor and critic models, while the learning curves are provided in Fig. 6. The agent achieved an average episode score of over 0.8 after training for a simulated time of about 3 hours.

### 4.3 Find the ball & avoid obstacles

The last example is a typical find target and avoid obstacles task with a simple world configuration. For this task the E-puck robot is used [7], which is a compact mobile robot developed by GCTronic and EPFL and is included in Webots. The world configuration contains an obstacle and a target ball. Different world configurations with incremental difficulty have been used in the training sessions for better generalization. It has been observed that the convergence of training algorithms can be improved by incrementing the difficulty of the problems [10]. The E-puck robot uses 8 IR proximity distance sensors and it has two motors for moving. The agent, apart from the distance sensor values, also receives the Euclidean distance and angle from the target. Consequently, the observation the agent gets is an one-dimensional vector with 10 values. On the other hand, the actuators are motors, which means that the outputs of the agent are two values controlling the forward/backward movement and left/right turning respectively (referred to as gas and wheel).

In order to deal with the continuous action space problem, the Deep Deterministic Policy Gradient (DDPG) algorithm was used to tackle this task [12]. The architecture of the models is described in Fig. 7. The reward function used for training the agent is calculated as:

$$R = \begin{cases} -10 & s > T_{steps} \\ +10 & d < T_{distance} \\ -1 & crashed \\ \frac{1}{d} - \frac{T_{steps}}{s} & otherwise \end{cases}, \quad (2)$$

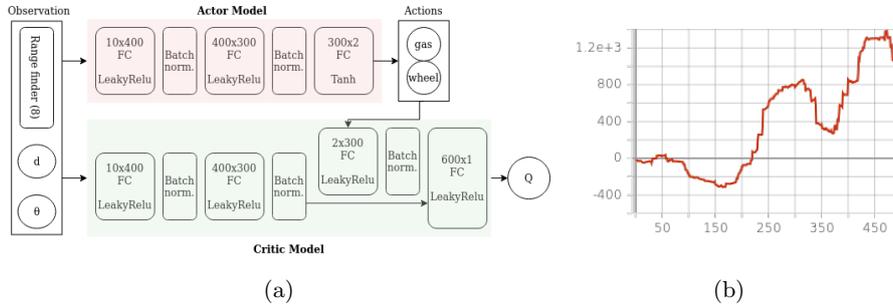


Fig. 7: FindBall: (a) DDPG models architecture, and (b) reward accumulated for each episode

where  $s$  is the current step,  $T_{steps}$  the maximum allowed steps,  $d$  the current distance from target and  $T_{distance}$  the minimum distance between the robot and the target which is considered as reaching the goal. This reward function takes into account both the distance from the target and the number of steps elapsed, while when the robot crashes on an obstacle or does not find the target after certain steps, it provides a negative reward and the episode is terminated.

The agent has been trained for 500 episodes and the accumulated reward is presented in Fig. 7. The training session lasted for about 1 hour of wall clock time and about 3 hours of simulated time. Although the agent solved the problem, it fails to generalize in more complicated scenes, highlighting the challenging nature of this baseline, that can be used for benchmarking future DRL algorithms.

## 5 Conclusions

Even though there have been attempts to formalize the use of RL in robotic simulators, none of them targets the state-of-the-art simulator Webots. The deepbots framework comes to fill that gap for anyone who wants to apply RL and DRL in a high fidelity simulator. Deepbots provides a standardized way to apply RL on Webots, by focusing only on parts that are important for the task at hand. Deepbots can fit a high variety of use cases, both research and educational, and can be extended by the community due to its open-source nature. Together with Webots, it provides a test bed for algorithm research and task solving with RL, as well as a practical platform for students to experiment with and learn about RL and robotics.

**Acknowledgments** This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR). This publication reflects the authors’ views only. The European Commission is not responsible for any use that may be made of the information it contains.

## References

1. Actin: <https://www.energid.com/actin>, industrial-Grade Software for Advanced Robotic Solutions
2. Barto, A., Sutton, R., Anderson, C.: Neuron like elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, Cybernetics* **13**, 834–846 (01 1970)
3. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym (2016)
4. Diankov, R.: Automated Construction of Robotic Manipulation Programs. Ph.D. thesis, Carnegie Mellon University, Robotics Institute (August 2010), [http://www.programmingvision.com/rosen\\_diankov\\_thesis.pdf](http://www.programmingvision.com/rosen_diankov_thesis.pdf)
5. Dulac-Arnold, G., Mankowitz, D.J., Hester, T.: Challenges of real-world reinforcement learning. *CoRR* **abs/1904.12901** (2019), <http://arxiv.org/abs/1904.12901>
6. Ferigo, D., Traversaro, S., Metta, G., Pucci, D.: Gym-ignition: Reproducible robotic simulations for reinforcement learning (2019)
7. Gonçalves, P., Torres, P., Alves, C., Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.C., Floreano, D., Martinoli, A.: The e-puck, a robot designed for education in engineering. *Proc. Conf. on Autonomous Robot Systems and Competitions* **1** (01 2009)
8. Goodfellow, I.G., Bengio, Y., Courville, A.C.: Deep learning. *Nature* **521**, 436–444 (2015)
9. Gym, I.: <https://www.nvidia.com/en-in/deep-learning-ai/industries/robotics>
10. Hacoen, G., Weinshall, D.: On the power of curriculum learning in training deep networks (2019)
11. Koenig, N., Howard, A.: Design and use paradigms for gazebo, an open-source multi-robot simulator. In: *Proc. 2004 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. vol. 3, pp. 2149–2154 (2004)
12. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning (2015)
13. Lopez, N.G., Nuin, Y.L.E., Moral, E.B., Juan, L.U.S., Rueda, A.S., Vilches, V.M., Kojcev, R.: gym-gazebo2, a toolkit for reinforcement learning using ros 2 and gazebo (2019)
14. Metta, G., Sandini, G., Vernon, D., Natale, L., Nori, F.: The icub humanoid robot: An open platform for research in embodied cognition. *Performance Metrics for Intelligent Systems (PerMIS) Workshop* (01 2008)
15. Michel, O.: Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems* **1**(1), 39–42 (2004)
16. Michel, O.: Webotstm: Professional mobile robot simulation. *Int. Journal of Advanced Robotic Systems* **1** (03 2004)
17. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning (2013)
18. Moore, A.: Efficient memory-based learning for robot control (06 2002)
19. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms (2017)
20. Steckelmacher, D., Plisnier, H., Roijers, D.M., Nowé, A.: Sample-efficient model-free reinforcement learning with off-policy critics (2019)
21. Todorov, E., Erez, T., Tassa, Y.: Mujoco: A physics engine for model-based control. pp. 5026–5033 (10 2012)
22. Zamora, I., Lopez, N.G., Vilches, V.M., Cordero, A.H.: Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo (2016)

## **7.17 Speech Command Recognition in Computationally Constrained Environments with a Quadratic Self-organized Operational Layer**

The appended paper follows.

# SPEECH COMMAND RECOGNITION IN COMPUTATIONALLY CONSTRAINED ENVIRONMENTS WITH A QUADRATIC SELF-ORGANIZED OPERATIONAL LAYER

Mohammad Soltanian\*      Junaid Malik<sup>†</sup>      Jenni Raitoharju<sup>‡</sup>  
Alexandros Iosifidis<sup>§</sup>      Serkan Kiranyaz<sup>¶</sup>      Moncef Gabbouj<sup>||</sup>

\*<sup>†||</sup> Department of Computing Sciences, Tampere University, Finland

<sup>‡</sup>Programme for Environmental Information, Finnish Environment Institute, Jyväskylä, Finland

<sup>§</sup>Department of Electrical and Computer Engineering, Aarhus University, Denmark

<sup>¶</sup>Electrical Engineering Department, Qatar University, Qatar

## ABSTRACT

Automatic classification of speech commands has revolutionized human computer interactions in robotic applications. However, employed recognition models usually follow the methodology of deep learning with complicated networks which are memory and energy hungry. So, there is a need to either squeeze these complicated models or use more efficient light-weight models in order to be able to implement the resulting classifiers on embedded devices. In this paper, we pick the second approach and propose a network layer to enhance the speech command recognition capability of a lightweight network and demonstrate the result via experiments. The employed method borrows the ideas of Taylor expansion and quadratic forms to construct a better representation of features in both input and hidden layers. This richer representation results in recognition accuracy improvement as shown by extensive experiments on Google speech commands (GSC) and synthetic speech commands (SSC) datasets.

*Index Terms*— Light-weight model, Taylor expansion, quadratic forms, self-organized neural network

## 1. INTRODUCTION

Automatic Speech recognition (ASR) is the art and science of automatic identification of speech. Speech commands recognition (SCR) is a subcategory of ASR in which the machine identifies short spoken commands. It is a key technology which enables machines to understand human commands. Possible applications include smart-phones with mobile assistants, robots which follow human spoken instructions, and smart home assistants, like Amazon Echo and Google Home. These kinds of applications are increasingly affecting human machine interactions. There are also many use cases for this technology in both medicine and education, e. g., for blind or handicapped people [1, 2]. As many of these applications rely on small sized devices which have limited computational capacities, fast and lightweight implementations are needed to work smoothly in real time [3].

Classical SCR approaches rely on extracting discriminative features from the raw speech. Frequency domain features like fast Fourier transform are among the most widely used features [4, 5],

to convert the time domain speech into frequency space. After feature extraction, an acoustic model such as hidden Markov model (HMM) was classically used to represent the sequence of words of phonemes [6]. Combination of neural networks and HMM-based approaches have also been in use since more than 20 years ago [7, 8].

Recently, deep learning-based approaches have dominated, in terms of performance accuracy, over conventional machine learning methods for many different applications [9–11]. Neural networks like long short term memory (LSTM) have raised the accuracy of natural language processing systems beyond the traditional approaches [12, 13]. Also, the neural network based approaches have dominated the whole SCR history and have been accepted as the state of the art tool by the speech recognition society [14–17]. In the meanwhile, several different neural network architectures like CNNs, LSTMs, and GRUs have been successfully applied to SCR [15–17]. Direct application of CNNs on the raw audio is also explored in [18–20]. This is while the low computational and memory capacities of the embedded devices are still serious barriers to fully exploit the benefits of highly accurate complicated deep networks.

To tackle the problem of computational complexity on embedded devices, there are usually two general approaches. One can squeeze the state of the art deep models to reduce the computational costs without severely degrading the performance accuracy [21, 22]. Alternatively, more efficient deep networks and methods can be employed from the beginning in order to achieve better accuracy rates with lower computational costs [23, 24]. The second approach especially becomes important when training examples are scarce or the input signals are of low quality or resolution. Using either of these ways, one can implement a reasonable classifier on energy constrained embedded devices.

There are many studies focusing on squeezing the complicated networks or building lightweight networks from the beginning. Progressive operational perceptrons (POPs) [25] try to generalize the conventional multilayer perceptrons (MLPs) using generalized operational perceptrons (GOPs). GOPs have a distinct set of operators, to mimic the synaptic connections of biological neurons, which enables them to better learn the hidden semantics of data. PyGOP [26] is a library for implementation of GOPs in both CPU and GPU. Authors in [27] accelerate the learning process in POPs employing the concept of memory and information from previously learned layers. The idea of biologically inspired neurons is further explored in [28], where each neuron in the network can have its distinct set of operators, which results in a compact network.

In [29], a generic neural network layer structure employing a

This work received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR). This publication reflects the authors’ views only. The European Commission is not responsible for any use that may be made of the information it contains.

multi-linear projection is proposed, which requires several times less memory compared to traditional CNNs, while achieving better performance. Authors in [15] used deep networks with dilated convolutions to achieve a high accuracy with lower number of parameters. SincNet Kernels [19] at the first convolutional layer of a CNN is another method to shrink the CNN by lowering the number of learnable parameters. This method is used for speaker recognition in [18] and later for phoneme recognition [19]. Depth-wise separable convolutions (DSCconvs) have also been introduced firstly in image processing field [30,31] and later to SCR and keyword spotting [15] and machine translation [32], in order to reduce the amount of required computations.

In this paper, we propose a new network layer to enhance the speech command recognition capability using a lightweight network. To this end, we use a novel combination of self organized operational neural networks (SelfONNs) [24] and quadratic forms kernels [33]. The experiments on Google speech commands (GSC) [34] and synthetic speech commands (SSC) datasets [35] show the effectiveness of the proposed approach for recognition of speech commands especially in lightweight networks.

To the best of our knowledge, we are the first to propose a combination of SelfONNs and quadratic kernels to make a new network layer type to boost the performance of a lightweight CNN. We are also the first to apply concepts of SelfONN and quadratic kernels on the task of SCR.

In the following, we describe the most relevant work in Section II. The proposed quadratic SelfONN approach to the problem of SCR is given in Section III, and the experiments are given in Section IV. Finally, the paper is concluded in Section V.

## 2. RELATED WORK

### 2.1. Self Organized Operational Neural Networks

Operational Neural Networks (ONNs) [23] are heterogeneous networks with a generalized neuron model that include a dictionary of nodal operators (instead of just an ordinary convolution) to boost the performance of the conventional CNNs. However, they need a greedy search to find the optimal operators which result in the best accuracy. The performance is also highly dependent on the selection of the initial operator set dictionary. In order to resolve these issues, the authors in [24] have recently proposed SelfONNs with generative neurons which are able to adapt the nodal operators of each neuron during the training phase. This removes the need for a fixed operator set and greedy search within that library to find the optimal set of operators. The authors show, via extensive experiments, the superiority of SelfONNs over ONNs in four applications: image synthesis, image denoising, face segmentation, and image transformation.

The main concept behind SelfONN is to use Taylor series expansion of the nodal function at each layer instead of the applying the ordinary convolution. An ordinary convolutional layer computes the convolution of the input with the layer weights as the following:

$$\mathbf{Y} = \mathbf{X} \circledast \mathbf{W} + \mathbf{b}, \quad (1)$$

where  $\mathbf{X}$ ,  $\mathbf{W}$ ,  $\mathbf{b}$ , and  $\mathbf{Y}$ , are input, weight, bias, and output tensors, respectively.

In ONNs, however, this is generalized to the following form:

$$\mathbf{Y}_{\text{ONN}} = \Psi(\mathbf{X}, \mathbf{W}) + \mathbf{b}, \quad (2)$$

where  $\Psi$  is an arbitrary nodal operator and may be a combination of different functions. If we set  $\Psi$  function equal to the regular convo-

lution operator ( $\circledast$ ), the ONN layer transforms back to the ordinary convolutional layer.

In SelfONNs, instead of using a specific function  $\Psi$  as the nodal operator, a suitable function can be approximated with a truncated Taylor series expansion to resolve the above mentioned shortcomings of the ONN. So, the SelfONN layer is defined as:

$$\begin{aligned} \mathbf{Y}_{\text{SelfONN}} &= \mathbf{W}_{T_0} + (\mathbf{X} - \mathbf{A}) \circledast \mathbf{W}_{T_1} + (\mathbf{X} - \mathbf{A})^2 \circledast \mathbf{W}_{T_2} + \\ &\dots + (\mathbf{X} - \mathbf{A})^K \circledast \mathbf{W}_{T_K} + \mathbf{b}_S \\ &= \sum_{i=0}^K (\mathbf{X} - \mathbf{A})^i \circledast \mathbf{W}_{T_i} + \mathbf{b}_S, \end{aligned} \quad (3)$$

where  $\mathbf{A}$  is the point around which  $\Psi$  is expanded.  $\mathbf{W}_{T_i} = \frac{1}{i!} \frac{\partial \Psi}{\partial \mathbf{X}^i}$  is the  $i$ 'th order partial derivative of  $\Psi$  with respect to the input, and  $K$  controls the order of approximating polynomial.  $\mathbf{b}_S$  is the bias term associated with the SelfONN layer. It is argued in [24] that the bias term ( $\mathbf{b}_S$ ) and the DC term in Taylor expansion ( $\mathbf{W}_{T_0}$ ) can be merged into a single term. In practice, the input is assumed to be centered around zero ( $\mathbf{A} = 0$ ). As the goal is not to estimate any particular function  $\Psi$ , but to learn the most suitable nodal function,  $\mathbf{W}_{T_i}$ s are learned via a training process (each  $\mathbf{W}_{T_i}$  is a learnable weight which can be learned via back propagation).

### 2.2. Quadratic-form Kernels

The core idea behind the network layer based on the quadratic form expansion [33] is that it generalizes the linear convolution by taking into account the cross correlation between the input elements within the receptive field of the layer kernel. In other words, in addition to ordinary convolution between the input and the weight, a second-order convolution between the input and an expanded weight tensor is computed which improves the final performance in terms of classification accuracy. The layer based on quadratic form expansion is therefore expressed as:

$$\mathbf{Y}_{\text{Quadratic}} = \mathbf{X}^H \circledast \mathbf{W}_{Q_2} \circledast \mathbf{X} + \mathbf{X} \circledast \mathbf{W}_{Q_1} + \mathbf{b}_Q \quad (4)$$

in which  $\mathbf{W}_{Q_1}$  and  $\mathbf{W}_{Q_2}$  are weight tensors,  $\mathbf{b}_Q$  is the bias term, and  $()^H$  is the Hermitian operator. Equation (4) is the same as (1) except for the added quadratic term  $\mathbf{X}^H \circledast \mathbf{W}_{Q_2} \circledast \mathbf{X}$ . This quadratic term improves the SCR accuracy, as shown in the simulations, mainly because it models second-order dependencies of the input features.

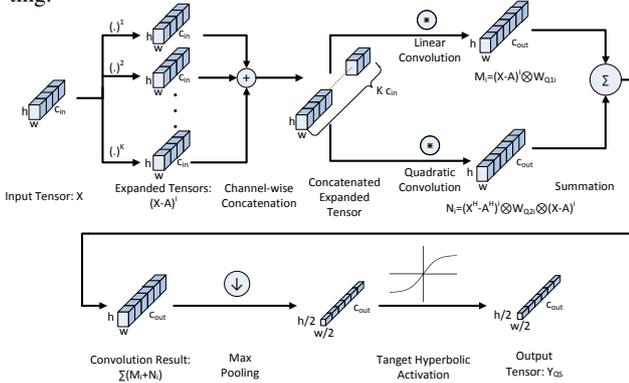
## 3. PROPOSED QUADRATIC SELF ORGANIZED OPERATIONAL NEURAL NETWORK METHOD

The concepts of SelfONN and quadratic convolution have not yet been applied to SCR problem. We are the first who use these methods in computationally constraint environments in order to boost the SCR accuracy. Additionally, we combine these two layer types into a unique layer which inherits advantages of both SelfONN (by a generalized convolution) and quadratic kernels (by modeling cross correlation of input feature points). To do so, we merge the Taylor series expansion in (3) with the quadratic-form convolution in (4) as the following:

$$\begin{aligned} \mathbf{Y}_{QS} &= \sum_{i=0}^K (\mathbf{X}^H - \mathbf{A}^H)^i \circledast \mathbf{W}_{Q_{2i}} \circledast (\mathbf{X} - \mathbf{A})^i + \\ &\sum_{i=0}^K (\mathbf{X} - \mathbf{A})^i \circledast \mathbf{W}_{Q_{1i}} + \mathbf{b}_Q, \end{aligned} \quad (5)$$

in which  $Y_{QS}$ ,  $W_{Q_{1_i}}$ ,  $W_{Q_{2_i}}$ , and  $b_Q$  are the output feature map, linear and quadratic weight tensors, and the bias term, respectively.

The flow diagram of the proposed layer is shown in Figure 1. The nodal operator based on the Taylor series expansion proposed in [24] is firstly applied to the input. The resulting feature map is fed to the quadratic-form convolution. Regular max pooling and tangent hyperbolic activation are then applied to the output. Dropout is also used after activation function in the training phase to avoid overfitting.



**Fig. 1.** Flow diagram of the proposed quadratic-SelfONN based layer: The spectrogram, MFCC, or any feature maps are fed as the input. The input is then expanded as a power series, and the resulting feature maps are concatenated. Afterwards, linear and quadratic convolutions are applied and the results are summed together. Finally, max pooling and tangent hyperbolic activation function are applied.

## 4. EXPERIMENTS

### 4.1. Datasets

The proposed model is evaluated on Google Speech Commands (GSC) [34] and synthetic speech commands (SSC) [35] datasets.

GSC is the most widely used benchmark for SCR systems. The training set of it consists of more than 60,000 audio files categorized into 32 directories with the folder names being the label of the audio clips. A subset of the dataset which is commonly used, e.g., in Kaggle competition, contains "on", "off", "yes", "no", "left", "right", "up", "down", "stop", and "go" commands. This is the subset we use for our evaluations in the current work. Each audio file in GSC is a voice clip with length of almost 1 second. Every audio file is in the form of 16-bit little-endian PCM-encoded WAVE file at a sampling frequency of 16000 Hz.

SSC is automatically generated and is actually text-to-speech counterpart for the GSC. It contains two versions, the normal version and very noisy version, each of which consists of 41849 audio files categorized into 30 classes. Since GSC is a low noise dataset, the very noisy version of SSC is employed in the experiments to assess the performance of the proposed method on noisy data as well. The same subset of classes as for GSC is used SSC in our experiments. In contrary to GSC, the audio files in SSC are not segmented into different train, test, and validation subsets. So, we randomly partition each category into train (80%), test (10%), and validation (10%).

The accuracy of the proposed method against the baseline is evaluated based on the number of correctly predicted labels in the test set divided by the total number of test samples, for both GSC and SSC datasets.

### 4.2. Data Pre-processing

The 1D raw input audio is converted to a 2D signal before being fed to the network. Actually, each input speech wave is segmented to 30 milliseconds windows with 10 milliseconds overlap. Then, Mel-frequency cepstral coefficients (MFCCs) are extracted from each window. Since the configuration is aimed to be used in computationally constrained environments, only 20 cepstral coefficients are kept at each window. Although 40 or more cepstral coefficients are usually used in research papers like [15, 16], it is widely believed that 15 or more cepstral coefficients will preserve the essential information in audio signals for classification applications. The resulting single channel 2D MFCC  $\in \mathbb{R}^{20 \times 51}$  is finally normalized to lie between -1 and 1.

### 4.3. Network Structure

For comparison of the proposed layer with the baseline convolutional layer in terms of accuracy of final speech commands classification, we need to use the layers in a deep structure which is end-to-end trainable. Since we aim to use the resulting network in computationally constrained environments, like embedded devices in robotic applications, we use a lightweight basic model for the evaluations. Indeed, the model could not dominate the complex state-of-the-art networks, but it is quite enough for a proof of concept, i.e., showing superiority of the proposed layer over the baseline convolutional layer. So, we choose a structure which is similar to LeNet-1 [36]. So, the original LeNet-1 has two convolutional and one fully connected layers. The network employed here has the same structure, but the convolutional layers are replaced with the proposed quadratic SelfONN layer. The codes will also be made freely available on-line upon publication of the paper.

After each convolutional layer a max pooling layer and a Tangent hyperbolic activation are used. Dropout is also used to prevent over-fitting. The proposed layer and the resulting network are implemented in PyTorch v1.4.0.

### 4.4. Hyper-parameters

For training of the network, the accuracy over validation set is computed at each epoch and the training proceeds if both following conditions are met: the epoch number is less than the maximum allowable epochs (100 in our simulations) and the accuracy over validation set is not less than the maximum achieved validation accuracy for the last 10 epochs. The mini-batch size is set to 50. The kernel size is set to  $3 \times 3$ . Padding of size 2 is also applied to the all sides of the input of the convolutional layers. The number of neurons (or channels) in two convolutional layers and one fully connected layer equals to 20, 20, and 10, respectively. Kernel size is fixed to  $3 \times 3$ , and dilation and stride are set to 1.

The values for kernel size and stride are determined based on cross validation on the training set using the CNN comprised of the baseline convolutional layers. Stochastic gradient descent with momentum = 0.9 and learning rate = 0.01 is used for network training.

### 4.5. Experimental Results

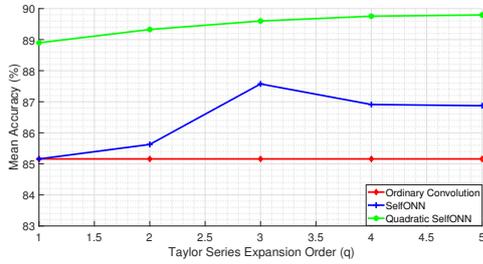
Figures 2 and 3 show the comparison of accuracy versus maximum Taylor series power ( $q$ ) over the test set between convolutional, SelfONN, and proposed layers for GSC and SSC datasets, respectively. As shown, the network comprised of the proposed layer outperforms the network with convolutional or SelfONN layers in terms of test accuracy over both datasets. For SelfONN, the values of  $q = 3$  and

**Table 1.** Performance comparison of the proposed layer with ordinary convolutional and SelfONN layers over GSC and SSC datasets.

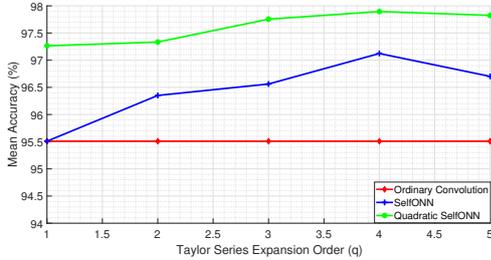
Dataset	Ordinary convolution	SelfONN	Proposed
GSC	85.2	87.6	89.8
SSC	95.5	97.1	97.9

$q = 4$  result in the best accuracy for GSC and SSC datasets, respectively. First, the accuracy of the proposed layer increases with increase in  $q$ , but the accuracy is saturated when  $q$  further increases.

As could be inferred from Figures 2 and 3, which is also summarized in Table 1, the accuracy gains for the proposed layer over the baseline convolutional layer are nearly 4.6% and 2.2% for GSC and SSC datasets, respectively. Similarly, the accuracy gains of the proposed layer over SelfONN layer are nearly 2.4% and 0.8% for GSC and SSC datasets, respectively.



**Fig. 2.** Comparison of the convolutional, SelfONN, and proposed layer in terms of test accuracy for SCR task over GSC dataset.



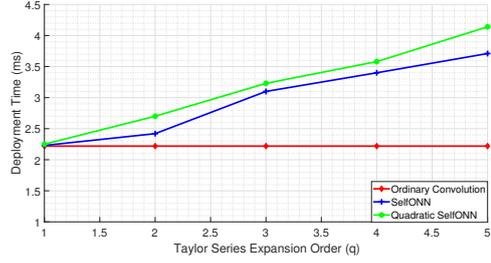
**Fig. 3.** Comparison of the convolutional, SelfONN, and proposed layer in terms of test accuracy for SCR task over SSC dataset.

Figure 4 evaluates the proposed layer based on the deployment complexity. To this end, the deployment time for a single 1 second speech command is computed by averaging the deployment time over the whole test set. The simulations and learning process are run on a machine with a Nvidia GeForce GTX 1080.

As can be seen, the deployment times of the proposed layer are higher than that of the convolution layer. However, the deployment time of the proposed layer is quite close to that of SelfONN.

#### 4.6. Comparison to the State-of-the-art

According to leader board section of Kaggle challenge on GSC dataset [37], the first team has achieved an accuracy around 91%.



**Fig. 4.** Comparison of the convolutional, SelfONN, and the proposed layers in terms of deployment time on a Nvidia GeForce GTX 1080 for SCR task over GSC dataset.

However, they probably used much more complex networks or ensemble of classifiers. They also used the original wave signals with full resolution spectrograms or higher number of cepstral coefficients. Since, the purpose of our work is the proof of concept of the proposed layer in computationally limited environments, we just used a very light-weight network, with small resolution inputs and compared the accuracy with the baseline convolutional layers. Even with these limitations, the achieved accuracy is quite high. However, it is expected that employing the proposed layer in deeper networks will result in higher accuracy values. A similar discussion also holds for SSC dataset.

## 5. CONCLUSION

An extension of the ordinary convolutional layer is proposed which is highly efficient especially at computationally constrained environments. It is especially appealing for robotic applications, where the resource limitations will not allow to use highly deep structures. The performance gap is promising with the proposed layer for embedded SCR applications. Future work will involve using the proposed layer in more complex networks in combination with network squeezing methods like network pruning, distillation, and sparsification of the fully connected layers.

## 6. REFERENCES

- [1] Shinichiro Okada, Yoshiaki Tanaba, Hideyuki Yamauchi, and Shoichi Sato, "Single-surgeon thoracoscopic surgery with a voice-controlled robot," *The Lancet*, vol. 351, no. 9111, pp. 1249, 1998.
- [2] Samir A. Elsagheer Mohamed, Allam Shehata Hassanin, and Mohamed Tahar Ben Othman, "Educational system for the holy quran and its sciences for blind and handicapped people based on google speech api," *Journal of Software Engineering and Applications*, vol. 7, no. 3, pp. 18–30, 2014.
- [3] Roman A. Solovyev, Alexandr A. Kalinin, Alexander G. Kustov, Dmitry V. Telpukhov, and Vladimir S. Ruhlov, "FPGA implementation of convolutional neural networks with fixed-point calculations," *arXiv preprint arXiv:1808.09945*, 2018.
- [4] K.-F. Lee and H.-W. Hon, "Speaker-independent phone recognition using hidden Markov models," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 11, pp. 1641–1648, 1989.

- [5] Urmila Shrawankar and Vilas M. Thakare, “Techniques for feature extraction in speech recognition system: A comparative study,” *arXiv preprint arXiv:1305.1145*, 2013.
- [6] Dan Jurafsky and James H. Martin, *Speech and Language Processing. Vol. 3*, Pearson London London, 2014.
- [7] Herve A. Bourlard and Nelson Morgan, *Connectionist Speech Recognition: A Hybrid Approach*, vol. 247, Springer Science & Business Media, 2012.
- [8] James L. McClelland and Jeffrey L. Elman, “The TRACE model of speech perception,” *Cognitive psychology*, vol. 18, no. 1, pp. 1–86, 1986.
- [9] Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdelrahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, and Tara N. Sainath, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.
- [10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*, MIT press, 2016.
- [12] Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins, “Learning to forget: Continual prediction with LSTM,” in *9th International Conference on Artificial Neural Networks*. 1999, pp. 850–855, IET.
- [13] Klaus Greff, Rupesh K. Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber, “LSTM: A search space odyssey,” *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.
- [14] Guoguo Chen, Carolina Parada, and Georg Heigold, “Small-footprint keyword spotting using deep neural networks,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2014, pp. 4087–4091, IEEE.
- [15] Yundong Zhang, Naveen Suda, Liangzhen Lai, and Vikas Chandra, “Hello edge: Keyword spotting on microcontrollers,” *arXiv preprint arXiv:1711.07128*, 2017.
- [16] Raphael Tang and Jimmy Lin, “Deep residual learning for small-footprint keyword spotting,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5484–5488.
- [17] Seungwoo Choi, Seokjun Seo, Beomjun Shin, Hyeonmin Byun, Martin Kersner, Beomsu Kim, Dongyoung Kim, and Sungjoo Ha, “Temporal convolution for real-time keyword spotting on mobile devices,” *arXiv preprint arXiv:1904.03814*, 2019.
- [18] Mirco Ravanelli and Yoshua Bengio, “Speaker recognition from raw waveform with sincnet,” in *IEEE Spoken Language Technology Workshop (SLT)*, 2018, pp. 1021–1028.
- [19] Mirco Ravanelli and Yoshua Bengio, “Interpretable convolutional filters with sincnet,” *arXiv preprint arXiv:1811.09725*, 2018.
- [20] Neil Zeghidour, Nicolas Usunier, Gabriel Synnaeve, Ronan Collobert, and Emmanuel Dupoux, “End-to-end speech recognition from the raw waveform,” *arXiv preprint arXiv:1806.07098*, 2018.
- [21] Justice Amoh and Kofi M. Odame, “An Optimized Recurrent Unit for Ultra-Low-Power Keyword Spotting,” in *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*. June 2019, vol. 3 of 2, pp. 1–17, Association for Computing Machinery.
- [22] Théodore Bluche, Maël Primet, and Thibault Gisselbrecht, “Small-Footprint Open-Vocabulary Keyword Spotting with Quantized LSTM Networks,” *arXiv:2002.10851*, Feb. 2020.
- [23] Serkan Kiranyaz, Turker Ince, Alexandros Iosifidis, and Moncef Gabbouj, “Operational Neural Networks,” *arXiv:1902.11106*, Oct. 2019.
- [24] Serkan Kiranyaz, Junaid Malik, Habib Ben Abdallah, Turker Ince, Alexandros Iosifidis, and Moncef Gabbouj, “Self-Organized Operational Neural Networks with Generative Neurons,” *arXiv preprint arXiv:2004.11778*, 2020.
- [25] Serkan Kiranyaz, Turker Ince, Alexandros Iosifidis, and Moncef Gabbouj, “Progressive operational perceptrons,” *Neurocomputing*, vol. 224, pp. 142–154, 2017.
- [26] Dat Thanh Tran, Serkan Kiranyaz, Moncef Gabbouj, and Alexandros Iosifidis, “PyGOP: A Python library for Generalized Operational Perceptron algorithms,” *Knowledge-Based Systems*, vol. 182, pp. 104801, 2019.
- [27] Dat Thanh Tran, Serkan Kiranyaz, Moncef Gabbouj, and Alexandros Iosifidis, “Progressive Operational Perceptrons with Memory,” *Neurocomputing*, vol. 379, pp. 172–181, 2020.
- [28] Dat Thanh Tran, Serkan Kiranyaz, Moncef Gabbouj, and Alexandros Iosifidis, “Heterogeneous multilayer generalized operational perceptron,” *IEEE transactions on neural networks and learning systems*, vol. 31, no. 3, pp. 710–724, 2019.
- [29] Dat Thanh Tran, Alexandros Iosifidis, and Moncef Gabbouj, “Improving efficiency in convolutional neural networks with multilinear filters,” *Neural Networks*, vol. 105, pp. 328–339, 2018.
- [30] François Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1251–1258.
- [31] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.
- [32] Lukasz Kaiser, Aidan N. Gomez, and Francois Chollet, “Depthwise separable convolutions for neural machine translation,” *arXiv preprint arXiv:1706.03059*, 2017.
- [33] Georgios Zoupourlis, Alexandros Doumanoglou, Nicholas Vretos, and Petros Daras, “Non-linear convolution filters for CNN-based learning,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4761–4769.
- [34] Pete Warden, “Speech commands: A dataset for limited-vocabulary speech recognition,” *arXiv preprint arXiv:1804.03209*, 2018.
- [35] “Synthetic Speech Commands Dataset,” 2017.
- [36] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [37] “TensorFlow Speech Recognition Challenge,” .