OpenDR —
Open Deep Learning Toolkit for Robotics

Project Start Date: 01.01.2020
Duration: 48 months
Lead contractor: Aristotle University of Thessaloniki

Deliverable D3.2 Second report on deep human centric active perception and cognition

Date of delivery: 31 December 2021

Contributing Partners: TAU, AUTH, AU

Version: v3.0

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

Title	**D3.2 Second report on deep human centric active perception and cognition**
Project	**OpenDR** (ICT-10-2019-2020 RIA)
Nature	Report
Dissemination Level:	**PU**blic
Authors	Negar Heidari (AU), Lukas Hedegaard Morsing (AU), Illia Oleksiienko (AU), Alexandros Iosifidis (AU), Moncef Gabbouj (TAU), Jenni Raitoharju (TAU), Anton Muravev (TAU), Kateryna Chumachenko (TAU), Dat Thanh Tran (TAU), Joonas Roos (TAU), Anastasios Tefas (AUTH), Nikos Nikolaidis (AUTH), Nikolaos Passalis (AUTH), Paraskevi Nousi (AUTH), Angelos Nalmpantis (AUTH), Pavlos Tosidis (AUTH), Charalampos Symeonidis (AUTH), Efstratios Kakaletsis (AUTH), Theodoros Manousis (AUTH), Konstantinos Tsampazis (AUTH)
Lead Beneficiary	TAU (Tampere University)
WP	3
Doc ID:	OPENDR_D3.2.pdf

Document History

Version	Date	Reason of change
v1.0	24/9/2021	Deliverable structure template ready
v2.0	26/11/2021	Contributions from partners finalized
v3.0	17/12/2021	Final version ready
v4.0	23/12/2021	Final version to be submitted

Contents

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. 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*, *Task 3.4–Deep speech and biosignals analysis and recognition*, and *Task 3.5–Multi-modal human centric perception and cognition*. 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 second year of the project in the five 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;
- Multi-modal human centric perception and cognition.

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

AUTH and TAU 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 OpenDR partners is briefly summarized below.

First, AUTH worked on models that support adaptive computational graphs allowing for easily adapting the computations to the available resources by selecting the most appropriate computational path (Section 2.1). However, such models are typically used in classification settings, e.g., using early exits, despite the fact that DL models often aim at extracting representations (metric learning), e.g., for face recognition. In this work, AUTH provided a metric learning oriented early exit methodology for DL models. As it was demonstrated, employing early exits in metric learning scenarios poses unique challenges compared to existing methodologies for classification-oriented early exits. To this end, the Bag-of-Features model was employed, building upon classification-oriented early exits, to efficiently extract compact representations from any layer of a DL model that is then combined with an efficient linear regressor to match the final representation of the model (without having to feedforward the whole computational graph). The proposed method is agile and can be directly used with any pre-trained DL model, while also being end-to-end differentiable, allowing for further fine-tuning the models towards having multiple early exits. The effectiveness of the proposed method was demonstrated using five face verification/recognition datasets.

In addition, AUTH also proposed a novel DNN-based Non-Maximum Suppression (NMS) method (Section 2.3) for the person detection task. NMS is a final refinement step incorporated to almost every visual object detection framework, assigned the duty of merging/filtering any spatially overlapping detected Regions-of-Interest (ROIs), i.e., bounding boxes, which correspond to the same visible object on an image. Person detection remains a challenging task for most NMS methods, facing difficulties in identifying individuals within a crowd, due to various levels of occlusions. Our method, named Seq2Seq NMMS, reformulates NMS for object detection as a sequence-to-sequence problem and proposes a novel DNN-based architecture, which relies on the Scaled Dot-Product Attention mechanism, in order to classify a ROI as "correct" or as "potentially suppressed". In addition, a new, GPU-based, fast and efficient method for extracting appearance-based ROI representations is proposed and employed on the overall pipeline. Seq2Seq-NMS was evaluated on three separate datasets, achieving favourable results compared to state-of-the-art NMS techniques.

Also, AUTH continued working on active vision approaches for human-centric perception (Section 2.2), exploiting the ability of robots to interact with their environment in order to better sense their surroundings. This key ability of robots is often ignored when developing Deep Learning (DL) models since they are usually trained using static datasets. As a result, this limits the ability of robots to perceive the environment in challenging scenarios, e.g., under occlusions. On the other hand, integrating perception and action in tightly coupled systems, typically known as active perception, holds the credentials for deploying DL-enabled robots in such conditions, leading to more robust agents that can solve challenging tasks in real-world scenarios more accurately. AUTH investigated whether active perception approaches can be employed and integrated into robotic systems in order to improve the face recognition accuracy, as well as study the effect of such an approach on the computational requirements. To this end, AUTH proposed a DRL-based control approach for training agents that are able to identify and focus on task-relevant objects, i.e., humans, as well as issue the appropriate control commands accordingly to acquire better results. Through the conducted experimental evaluation it was demonstrated that the proposed method leads to significant improvements in face recognition over the rest of the evaluated approaches, improving the distance from which a human can be recognized, as well as providing a much wider angle-invariance.

Also, AUTH proposed a novel approach for active face recognition using synthesized facial views (Section 2.4), which allows for exploiting photorealistic facial view rendering to aid the robot in deciding how to move in order to capture better facial views to use for face recognition. More precisely, AUTH investigated an active vision algorithm that aims to direct a robot to move in a position that will allow it to capture a good (in terms of face recognition accuracy) facial view of a person. The method utilizes a state-of-the-art facial view synthesis method that renders novel views based on a single facial image and an up-to-date Face Recognizer.

In addition, AUTH utilized a data generation method capable to generate realistic data, for the training and testing of methods for human centric perception tasks. An extensive experimental evaluation was performed, presented in Section 2.5, which validated that the generated data are visually realistic enough and that they can successfully bridge the gap between the virtual and real world data, by enabling deep learning methods trained on the synthetic and real data to achieve highly accurate results on both.

AUTH also worked towards evaluating the impact of domain shifts for object detection models trained on well known datasets (Section 2.6). More specifically, a dataset collected in the context of the agricultural use case using the Robotti robotic platform was employed to evaluate the accuracy of human detectors trained on existing datasets, identifying important limitations that these detectors face on such scenarios and highlighting the need for using domain adaptation and knowledge transfer approaches to ensure that they will perform as expected in such scenarios.

TAU worked towards improving the robustness of Multilinear Compressive Learning (MCL), which is an efficient signal acquisition and learning paradigm for multidimensional signals proposed by OpenDR researchers in previous year. The level of signal compression affects the detection or classification performance of a MCL model, with higher compression rates often associated with lower inference accuracy. However, higher compression rates are more amenable to a wider range of applications, especially those that require low operating bandwidth and minimal energy consumption. A novel optimization scheme was proposed (Section 2.7) that improves the robustness of the MCL model with respect to the size of the compressed signal, and enables practical implementation of adaptive compressive signal acquisition and inference systems.

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

AU has contributed novel methods for accelerating the online inference of both video- and skeleton-based human activity recognition networks. A brief of this work is given below.

Extensive benchmarks of the current best in class lightweight video recognition methods have been performed using various computational devices (Section 3.1.4). Here, only the smallest models (with severely reduced accuracy) were able to perform in real-time on an embedded GPU platform. Due to the convolutional architecture of these state-of-the-art lightweight models, and their requirement that the input be a spatio-temporal clip, a considerable computational redundancy was observed due to overlapping clips for frame-by-frame predictions.

To alleviate this redundancy, AU researchers have proposed a new class of convolution, the *continual* convolution, which can perform identical computations to those in a regular convolution while considering only one time-step at a time. Existing convolutional architectures and weights can thus be reused and accelerated through a continual inference mode. This principle was used to reduce the per-prediction floating point operations of 3D CNNs for video-based human action recognition by an order of magnitude (Section 3.1) as well as by two orders of magnitude in Spatio-Temporal Graph Convolutional Neural Networks (ST-GCNs) for skeleton-based action recognition (Section 3.2). Due to a local caching mechanism within each continual convolution (an operation, which common computational devices are not optimised for), the throughput was increased by approx $6\times$ for 3D CNNs and $18\times$ for ST-GCNs. Given these remarkable speed-ups, it is our hope that the contribution of AU will lead to the utilisation of higher accuracy models and/or enable the use of computationally constrained hardware in embedded applications and robotics.

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

AU developed a novel methodology and a corresponding tool for facial expression recognition based on videos or image sequences. The work conducted by AU is briefly summarized below.

The spatio-temporal features of a facial expression depicted in a video or image sequence can be effectively represented by the localized facial landmarks which are invariant to illumination variations, face appearance and scale. Therefore, a facial expression video can be modeled as a spatio-temporal graph, representing the dynamic motions of the key facial parts for each specific facial expression, and this graph can be employed by a landmark-based facial expression recognition method. AU had previously proposed and developed Spatio-Temporal BiLinear Networks (ST-BLN) [51] for skeleton-based human action recognition, which have been integrated into the OpenDR framework. The GCN-based methods defined for this task can also be extended for landmark-based facial expression recognition, as these tasks share similar ideas on processing a sequence of graph structured data instead of videos/image sequences. Accordingly, AU investigated the application of spatio-temporal GCN-based methods on landmark-based facial expression recognition, and proposed the Progressive ST-BLN (PST-BLN) method [52] which inherits the advantage of ST-BLN to learn graph structures at each layer of the network and automatically learns an optimized data-dependent network structure while simultaneously training the model parameters. The proposed method is evaluated on three widely-used facial expression recognition datasets and the experimental results indicate that it has achieved comparable performance to more complex state-of-the-art methods, while it also captures the model's uncertainty using Monte Carlo Dropout technique [154].

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

AUTH worked towards developing a deep high-order Gaussian filtering method for time series analysis (Section 5.1), including, but not limited to biosignal analysis, such as electrocardiograms. To this end, a discrete approximation of a Gaussian filter was appropriately constructed to support training the resulting layer in an end-to-end fashion through backpropagation. To further improve the performance of the proposed method, AUTH also proposed extracting multiple features using high-order derivatives of the Gaussian function by convolving the derivatives of the Gaussian filters with the input signal, leading to a smooth high-order representation of the input. The proposed series of layers are lightweight, since they consist only of a few trainable parameters, and can significantly improve time series classification, as demonstrated in the experimental evaluation.

TAU worked on developing new attention methods for Neural Bag of Features (NBoF) formulation for time series analysis (Section 5.3). The main focus was on the classification of biosignals under the task of the atrial fibrillation detection from the ECG data. To this end, self-attention based extensions to the previously-developed codebook and temporal attention mechanisms of NBoF were formulated. In addition, a codebook-temporal self-attention block was proposed as a way to mitigate the limitations of conventional attention approaches that lie in the independence of their learnt feature representations from each other, even if applied jointly. The proposed methodologies can be incorporated into any Bag of Features architecture. Preliminary results show the superiority of the proposed approaches. TAU also worked on extending the capabilities of the standard speech command recognition models by incorporating 1-dimensional SelfONN layers in place of traditional convolutions (Section 5.2). SelfONN is a more general and flexible operation that is capable of learning a wider range of behaviors, increasing the model capacity and power without modifying its depth.

1.5 Multi-modal human centric perception and cognition (T3.5)

TAU has contributed to the task by developing an attention-based methodology (Section 6.1) that can be used in CNNs for fusing information from different modalities in two-stream architectures [86]. Attention mechanisms in CNNs aim to explicitly identify and highlight relevant regions of the image and pass the attended representation to further layers of the network, generally allowing to achieve improved performance. At the same time, it can be argued that explicit learning of the parts of the image relevant to the given task is generally more challenging than learning which parts of the image are less relevant and, thus, should be ignored. Following this intuition, TAU researchers proposed a learning mechanism aiming to explicitly learn irrelevant information in the scene and suppress it in the produced representation, keeping only relevant attributes, hence learning the attention implicitly. The resulting implicit attention can be incorporated into existing attention mechanism and the developed methodology was evaluated using two state-of-the-art formulations, namely, SE and CBAM. The approach was evaluated both as a standalone plug-in attention module, as well as a part of multimodal fusion attention-based framework, evaluated for the task of action recognition from RGB+skeleton data on NTU-RGBD dataset.

In addition, TAU has worked towards multi-modal gesture recognition from RGB-D data and has developed an early-fusion based gesture recognition method from image data (Section 6.2). Early fusion is utilized due to a range of benefits that it is associated with, such as possibility of transfer learning from larger datasets, as well as little computational overhead. In the toolkit, several widely-used architecture backbones have been added, including lightweight architectures such as MobileNet. The model thus developed has been evaluated and validated on a 16-gesture dataset.

1.6 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*

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

AUTH developed an adaptive inference approach for deep representation learning tasks, such as face recognition (Section 2.1), which allows for meeting the strict speed and latency requirements for many robotics applications by allowing for adapting the computational requirements to the available resources on-the-fly. Furthermore, AUTH developed a deep high-order Gaussian filtering method (Section 5.1) for time series analysis, such as biosignals, that can allow for achieving higher analysis accuracy, while using smaller and faster architectures. AUTH also developed a DNN-based NMS method (Section 2.3) capable of improving the performance of person detection methods, especially in the case where humans appear in crowded areas, while maintaining relatively fast inference times compared other DNN methods. AUTH also worked towards evaluating the impact of domain shifts for object detection models trained on well known datasets (Section 2.6), identifying critical limitations and training approaches to mitigate approaches that could increase the robotic autonomy in the field. Finally, AUTH evaluated the use of mixed image data for training DNN methods for human centric perception tasks (Section 2.5) in order to validate whether they improve the performance of such methods on real-world settings as well on simulation environments.

AU proposed and developed two methodologies for efficient continual human activity recognition (Section 3.1 and Section 3.2), which allow to reduce the number of computations compared to the standard approach of sliding window-based classification. Moreover, AU proposed and developed an efficient facial expression recognition method based on facial landmarks and the PSTBLN network (Section 4.1) which also provides a measurement of the model's uncertainty in its recognition result.

TAU proposed a method for improving robustness of Multilinear Compressive Learning with respect to the size of the compressed signal and enables practical implementation of adaptive signal acquisition and inference systems (Section 2.7), which can be used for lightweight on-robot light-weight neural processing and offloading computations to more powerful processor. Moreover, TAU is developing self-attention extensions to NBoF attention approaches that would allow to improve robustness of the such lightweight models in time-series analysis in general, and biosignal classification in particular (Section 5.3). The developed methods are architecture-independent and can be utilized to improve robustness of lightweight NBoF models. TAU is also working on improving the performance of speech command recognition models with novel 1D SelfONN layers (Section 5.2), which are lightweight neural networks offering higher flexibility and improved learning capacity without losing the efficient inference properties of the original structures. In addition, TAU developed an attention-based methodology for Convolutional Neural Networks (Section 6.1) that can be used in a variety of computer vision tasks both as a standalone module, as well as for multimodal fusion in multi-stream CNN architectures, including lightweight ones. The method was evaluated for image classification and multimodal human action recognition tasks.

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

TAU developed and integrated an early fusion based method for multimodal hand gesture recognition from RGBD data (Section 6.2), enabling efficient human-robot interaction. The method achieves 38.1 FPS on XavierAGX.

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

AUTH proposed a deep reinforcement learning-based active face recognition methodology, going beyond existing rule-based approaches for active vision (Section 2.2). AUTH also proposed a novel approach for active face recognition using synthesized facial views providing a more effective way to perform active perception with simulating the effects of various control actions and exploiting photorealistic facial view rendering (Section 2.4).

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

2.1 Adaptive Inference for Face Recognition leveraging Deep Metric Learning-enabled Early Exits

2.1.1 Introduction, objectives and summary of state of the art

DL models are becoming increasingly powerful, following the continuous improvements in dedicated hardware accelerators, such as Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs) [161], which allowed for training and deploying deeper and more complicated models. However, in many applications, such as robotics, we are often still limited to using less powerful hardware, due to a number of limitations, ranging from energy and power constraints to constrained physical form factors. As a result, numerous methods have been proposed to allow for developing more *lightweight* DL models that will be deployed in such devices, while meeting critical application-specific requirements, such as low latency and real-time operation.

These methods include quantization [46], for reducing the number of bits spent for each of the parameters of the model, pruning methods [96], that discard parts of the model that are not critical for its operation, models that are lightweight by design [56, 173], as well as knowledge distillation approaches [54, 118], which aim to transfer the knowledge from a larger and more complex neural network into a smaller and faster one. These approaches led to more lightweight models that could operate faster in many embedded and mobile devices. However, most of these methods are not capable of adapting to varying computational loads. In other words, the inference time is constant regardless the environmental conditions, e.g., the difficulty of each sample, the load of the system, etc.

This is a critical limiting factor in a number of embedded applications, where the load dynamically varies according to the environmental conditions. For example, for a face recognition application the time needed for face recognition depends on the number of faces that appear in a given frame. As a result, even through a model might operate in real time for a specific number of faces, e.g., 2-3 faces, this might not be the case when a larger number of people appears in a given frame. Therefore, in such cases, we need models that can effectively adapt to the current conditions, providing faster (and possibly less accurate) predictions when the load is higher in order to satisfy the processing time limitations of a given application. In this way, the models can provide accurate answers, exploiting all the available processing time, while still meeting the requirements of each application when the load is higher.

These limitations can be addressed by using models that support *adaptive computational graphs*, such as [5, 115, 149]. These approaches work by altering the number of computations in order to keep the load within certain limits. This is usually achieved by using multiple paths over the computation graph of the model. Among the most straightforward ways to achieve this is by using *early exits* [5, 115, 149]. By placing such early *classification* layers at various intermediate layers of the network we can early stop the computation whenever it is deemed appropriate (e.g., when the computational budget is spent or when the network is already confident enough regarding the provided prediction), obtaining an estimation for the representation that would be extracted from the final output layer of the network.

Even though early exits provided a very powerful tool to address the aforementioned limitations, its use is currently limited to classification settings [5, 115, 149]. However, in many cases, deep learning models aim at extracting representations (metric learning) [140], instead of directly predicting the class to which the input sample belongs to. Perhaps the most well known example of such metric learning task is face recognition [109] and content-based information retrieval [82]. To the best of our knowledge there has been no attempt to use early exits in such scenarios, such as metric learning-based face verification. Even though it could be argued that using early exits in such scenarios could be deemed redundant, since we can directly extract a representation from any layer of a DL model without any modification, we demonstrate that this naive approach has significant limitations and that we can achieve higher accuracy by employing appropriately designed and trained early exit layers.

The main contribution of this work is to provide a metric learning-oriented early exit methodology for DL models. As we experimentally demonstrate, employing early exits in metric learning scenarios pose unique challenges compared to existing methodologies for classification-oriented early exits. To this end, in this work we leverage the Bag-of-Features model to efficiently extract compact representations from any layer of a neural network. Then, an additional small linear regressor is used to regress the final output of the model at selected points of its computational graph. In this way, a representation that can be used in place of the final representation can be readily extracted from an early exit. This provides significant advantages over existing metric learning approaches, which would require keeping a separate database for the representations extracted from each exit layer, increasing the space required for using any additional layer as an early exit and reducing the accuracy of the resulting models, as we demonstrate in the experimental evaluation. The proposed method is agile and can be directly used with any pre-trained metric learning DL model, while it is end-to-end differentiable, allowing for further fine-tuning the models towards having multiple early exits. The effectiveness of the proposed method is demonstrated using five face verification/recognition datasets, including DL models trained on the large-scale MS-Celeb-1M dataset [45] and evaluated using a wide range of datasets, as well as experiments conducted on two embedded platforms typically used in robotics applications.

A summary of this work is provided hereafter. The corresponding publication is listed below, and can be found in Appendix 8.1:

1. [120] N. Passalis, and A. Tefas, "*Adaptive Inference for Face Recognition leveraging Deep Metric Learning-enabled Early Exits*", European Signal Processing Conference (EUSIPCO) 2021

2.1.2 Description of work performed so far

For the rest of this section, we assume that we employ a network that has already been trained to perform a specific metric learning task [109, 17], and we will focus on training the early exits on top of the representations $\mathbf{y}^{(i)}$ extracted at specific points of its computational graph. Early exits typically employ an additional estimator, fitted on top of the representation extracted at various points of the computation graph of the model, to predict the final output of the model. Classification-based early exits are trained to directly solve the original classification task of the network [149, 115, 5]. However, for metric-learning oriented network this approach cannot be employed, since even if we use the original loss used for training the network, e.g., the contrastive loss [17], we will not learn representations in the same space as the one formed by the last layer of the network. As a result, the representations extracted by the early exits would not be useful for performing queries in a database that consists of representations extracted from the final layer of the network.

To overcome this limitation, in this work we proposed using a distillation inspired approach [54], i.e., to train the early exits in order to mimic the output of each layer. Perhaps the most straightforward approach to ensure that the features extracted by the early exits, denoted by $g^{(i)}_{\mathbf{W}_i}(\cdot)$, will reside in the same space as the final representation of the network $\mathbf{y}$ is to minimize the quadratic divergence between these two representations. Therefore, early exit estimators are trained in order to minimize the following loss:

$$\mathscr{L}_i = \frac{1}{N}\sum_{j=1}^{N} ||\mathbf{y}_j - g^{(i)}_{\mathbf{W}_i}(\mathbf{y}^{(i)}_j)||_2^2, \tag{1}$$

where i denotes the early exit that we are training, $||\cdot||_2$ denotes the l_2 norm of a vector and the notation $\mathbf{y}_j$ is used to refer to the representation extracted when the j-th sample is fed into the network.

Usually, early exits employ a feature aggregation approach to reduce the dimensionality of the extracted feature maps, e.g., Global Average Pooling, that is then followed by a fully connected layer. However, naive feature aggregation approaches, such as global average/max pooling, have been shown to discard useful information [116]. Therefore, in this work we employ a Bag-of-Features (BoF)-based aggregation layer in order to reduce the dimensionality of the extracted feature maps and extract a compact summary representation that can be further adapted towards the task at hand [117]. This histogram representation extracted using BoF provides a summary of the concepts that appear in the corresponding features. By appropriately tuning the codewords we can *focus* the representation on different concepts. For example, using *k-means* to learn the codebook leads to a generic representation that can be used for any task, while finetuning the whole layer using gradient descent allows for learning task-specific codewords (provided that the BoF layer is part of a network trained for a specific task). Finally, this histogram representation is fed into a linear layer that projects the histogram into the desired space. In the case of metric-learning networks this would be the space formed by the output layer of the network. Then, early exits can be trivially trained using gradient descent, minimizing the loss provided in (1).

2.1.3 Performance evaluation

The proposed method was evaluated using the MS-Celeb-1M [45], Labeled Faces in the Wild (LFW) [62, 61], Cross Pose LFW (CPLFW) [177], Cross Age LFW (CALFW) [178], and VGGFace2 [14] datasets. More specifically, we follow a standard face verification setup [1], where the models are trained on the MS-Celeb-1M dataset and evaluated on the remaining four datasets, i.e., LFW, CPLFW, CALFW and VGGFace2. All images used for the conduncted experiments were resized to 112×112 pixels. For the evaluation procedure we randomly sample 6,000 image pairs that either correspond to face images of the same person or to face images of different persons (equally distributed among the two cases). A face pair is considered to belong to the same person when the distance between the corresponding embeddings is lower than a certain threshold. This threshold is selected to maximize the face verification accuracy on a validation dataset. As a result, we report the average 10-fold cross validation accuracy for all the conducted experiments, i.e., the threshold is selected according to the validation split and the accuracy is reported on the corresponding test set. For all the conducted experiments we used a ResNet-50 network [48], where inverted residual blocks were employed for improving its efficiency [130]. The early exits were placed after the 1st, 2nd and 3nd residual block. The dimensionality of the feature vectors extracted from each of these blocks is 128, 256 and 256 respectively, while the dimensionality of the final representation of the network is 512.

Three different methods were evaluated along with the proposed one. For the first one we directly extracted the feature vectors from each early exit, we employed global average pooling and then we queried the database using these representations. This approach is called "*raw*", since it relies on directly using the raw feature vectors, as they are extracted from the network. Even though the dimensionality of these feature vectors is lower, this method requires keeping a separate database with the feature vectors extracted from each additional early exit, significantly increasing the storage requirements. Next, we evaluated a linear regressor (denoted by "LR") that was trained to directly regress the output representation of the network based on the (average) pooled representation extracted from each early exit. The same approach was also repeated using the Bag-of-Feature model, where we used 512 codewords for building the codebook (using the k-means algorithms and the feature vectors extracted from each early exit for building the codebook). This method is denoted as "BoF". Note that both the LR and BoF methods can be regarded as a simplified (ablated) version of the proposed one, since, to the best of our knowledge, neither has been proposed in the literature for constructing early exits. Despite this, they consist a strong baseline, as we demonstrate in Table 1. Finally, we evaluated the proposed method using again $N_K = 512$ codewords (in order to be directly comparable with the BoF baseline). More details regarding the experimental setup are provided in Appendix 8.1.

The experimental evaluation is provided in Table 1. The four evaluated methods are compared on four different datasets using the three different added early exits. In all the cases, using a subsequent early exit increases the obtained verification accuracy as expected. Furthermore, just using a linear regressor (LR) to regress the final representation of the network leads to a significant increase over directly using the raw representation. Indeed, in some cases (e.g., CALFW) the accuracy increases by over 25%. Then, using the BoF model further increases the performance, while employing the proposed method leads to the overall best accuracy in all the evaluated cases. It is worth noting that in some cases, the verification performance is very close to the actual performance of the final output of the network, as reported in the full paper provided in the appendix.

2.2 Learning active vision control policies for face recognition using deep reinforcement learning

2.2.1 Introduction and objectives

There have been several recent attempts to integrate active perception principles into DL models [106, 119]. Most of them focused on robotics tasks, where they attempt to appropriately manipulate a camera and/or robot in order to improve the accuracy of the models. However, training DL models for such tasks is not trivial, since most datasets used for training DL models do not provide the appropriate data and/or annotations that can be exploited in active perception scenarios. Indeed, active perception requires an agent that can *interact* with its environment and acquire an improved view of the world. To overcome this limitation, existing methods either employ simple handcrafted rules for implementing active perception feedback [106], or use multi-view datasets to simulate some of the effects of active perception feedback [119]. However, due to the lack of appropriate datasets, such methods are still usually trained with simplistic rules, e.g., to predict if moving left/right will increase/decrease the confidence on correctly recognizing a person [119]. Another closely related line of work employs Deep Rein-

Table 1: Face verification accuracy on four different datasets using three different early exits. The mean and standard deviation of the 10-fold cross-validation accuracy is reported.

Method	**Exit 1**	**Exit 2**	**Exit 3**
Dataset: LFW			
Raw	68.60 ± 2.25	82.33 ± 1.25	92.80 ± 1.57
LR	75.05 ± 2.00	88.98 ± 1.42	94.15 ± 0.69
BoF	79.93 ± 1.59	92.33 ± 1.22	92.58 ± 1.23
Proposed	$\mathbf{80.98 \pm 2.64}$	$\mathbf{92.70 \pm 1.28}$	$\mathbf{96.58 \pm 0.64}$
Dataset: CPLFW			
Raw	52.75 ± 1.95	51.98 ± 1.74	63.83 ± 2.38
LR	66.80 ± 1.71	75.57 ± 2.04	83.02 ± 1.36
BoF	68.52 ± 1.79	79.98 ± 2.16	81.13 ± 1.05
Proposed	$\mathbf{68.98 \pm 1.18}$	$\mathbf{80.05 \pm 1.70}$	$\mathbf{84.20 \pm 1.56}$
Dataset: CALFW			
Raw	55.00 ± 1.62	61.72 ± 1.50	68.10 ± 1.98
LR	70.28 ± 1.51	82.18 ± 1.89	87.35 ± 1.18
BoF	71.80 ± 1.18	84.55 ± 1.06	84.93 ± 1.51
Proposed	$\mathbf{73.65 \pm 1.82}$	$\mathbf{84.78 \pm 1.50}$	$\mathbf{89.37 \pm 1.37}$
Dataset: VGGFace2			
Raw	56.88 ± 2.07	64.24 ± 1.12	78.88 ± 1.43
LR	67.10 ± 1.74	78.50 ± 1.85	84.16 ± 1.33
BoF	68.86 ± 1.84	81.56 ± 2.69	83.50 ± 1.55
Proposed	$\mathbf{71.34 \pm 2.04}$	$\mathbf{82.32 \pm 2.32}$	$\mathbf{88.10 \pm 1.43}$

forcement Learning (DRL) algorithms to perform a specific control task [103, 132, 104], e.g., acquire a frontal view of a person [153]. Despite the effectiveness of DRL approaches in these robotics tasks, applying them on challenging computer vision tasks typically require realistic simulation environments and/or appropriate training methods, e.g., *sim2real* approaches [129]. At the same time, the lengthy training time of DRL methods further limits their applications in robotics. As a result, despite its enormous potential for developing active perception approaches, its application faces significant obstacles.

The main contribution of this work is to propose a DRL-based active perception approach integrated with state-of-the-art DL-based face recognition models. More specifically, our goal is to investigate whether active perception approaches can be employed and integrated into robotic systems in order to improve face recognition results, as well as study the effect of such an approach on the computational requirements. To this end, we propose a DRL-based control approach for training agents that are able to identify and focus on task-relevant objects, i.e., humans, as well as issue the appropriate control commands accordingly to acquire better results. To train and evaluate the proposed method we developed a simulation environment using the Webots simulator [101] and generated several 3D human models using the MakeHuman software [7]. The proposed method aims to control a drone, equipped with a camera, in order to improve face recognition results over existing baseline and rule-based active perception approaches. Indeed, as the experimental results demonstrate, the proposed method managed to lead to significant improvements in face recognition over the rest of the evaluated approaches, improving the distance from which a human can be recognized, as well as providing a much wider angle-invariance. Indeed, the trained agents showed an emergent behavior that can resemble those of humans, e.g., move closer or around a person in order to more confidently identify it. At the same time, it is demonstrated that the proposed method can also lead to computational savings, under certain conditions.

2.2.2 Summary of state of the art

Face recognition research in the past years has made tremendous leaps. From traditional approaches that represent faces with hand-crafted features extracted from an image [170], to modern deep learning approaches that automatically learn the distinctive features of a face when trained on massive datasets [23, 158, 93]. The face recognition pipeline of such approaches typically consists of four stages: a) face detection and cropping, b) (optionally) face alignment, c) feature extraction, and d) classification/verification. The two first stages are often considered as preprocessing stages. A face recognition model requires an input image that is carefully cropped and aligned. Then, this prepocessed image is fed into a DL model that extracts a discriminative feature vector. Finally, this vector is compared to a set of feature vectors of people of interest [23, 158, 93], performing the final classification or verification task. The method proposed in this work is orthogonal to these approaches, since it can be readily combined with any face recognition model and further increase its accuracy. Indeed, as demonstrated in the conduncted experiments, the proposed method can be readily combined with a state-of-the-art DL-based face recognition system and increase its accuracy by integrating it into a active vision pipeline.

This work is also closely related to active perception approaches. According to Bajscy [6], an actively perceiving agent is one which can, among others, appropriately control its mechanical components in order to enable the best sensing of the surroundings, as well as select the best viewpoint to achieve the task in hand. However, there are only a few recent approaches to active face recognition using DL [106, 119]. An active face recognition system that employs a DL model to extract the facial features and a controller module to act based on the results of the DL model was proposed in [106]. The controller module works as a rule-based controller that selects the most appropriate action based on the face recognition confidence and predefined thresholds for each action. A fully end-to-end trainable DL-based approach was also proposed in [119], where a DL model was trained to output both the face feature embeddings, as well as a suggested action. The network was trained on a small dataset containing facial images at various pans and tilts, providing a proof-of-concept demonstration for a DL-based pipeline for active face recognition. Also, this approach cannot fully exploit the potential of active perception, since it only considered 1-step actions for training the control branch of the DL model.

The proposed method goes beyond these approaches by employing a powerful RL-based formulation that is both end-to-end trainable and does not make any assumption regarding the control policy. In this way, more advanced policies can be discovered without introducing any strong prior, using handcrafted rules either for training or inference. However, at the same time, the proposed method requires a realistic simulation environment for training, since the control module cannot be trained using the existing static datasets. To overcome this limitation, in this work we employed the realistic Webots simulator, along with 3D human models generated using the MakeHuman software. Furthermore, both of the aforementioned works require the use of a face detector to appropriately crop the face image before feeding it to the face recognition module. On the other hand, the proposed method allows for significantly reducing the computational requirements by working independently of the face recognition model. In this way, a lightweight DL model is used for performing control and the heavy face recognition pipeline (face detection and recognition) is only employed when deemed appropriate.

2.2.3 Description of work performed so far

Let $\mathbf{x} \in \mathbb{R}^{W \times H \times C}$ be an image that contains a face to be recognized, where W, H and C are the width, height and number of channels of the corresponding image. As described before, face recognition algorithms require to first employ a face detection model to detect and crop the bounding box that encloses each face. Therefore, let

$$\mathbf{x}_p = f_p(\mathbf{x}) \in \mathbb{R}^{W_p \times H_p \times C_p}, \tag{2}$$

be the cropped face image, where the notation $f_p(\cdot)$ is used to refer to the face detector and prepossessing pipeline employed to crop the image and W_p, H_p, and C_p are the width, height and number of channels of the cropped image. Most recent deep face recognition methods, e.g., [93], aim at learning an appropriate model $\mathbf{y} = f_r(\mathbf{x}_p) \in \mathbb{R}^D$ that will extract a discriminative identify-oriented representation from each face image, where D denotes the dimensionality of the embedding space used for representing the input face images.

Different loss functions have been proposed to train the face recognition model f_r to extract discriminative embeddings. In this work, we employ the *Additive Angular Margin Loss* [23], 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 are as far as possible. After training the model $\mathbf{y} = f_r(\mathbf{x}_p)$, the identity of a person depicted in an image $\mathbf{x}_p$ can be obtained simply by calculating the Euclidean distance between the feature vector of that image and the feature vectors 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,

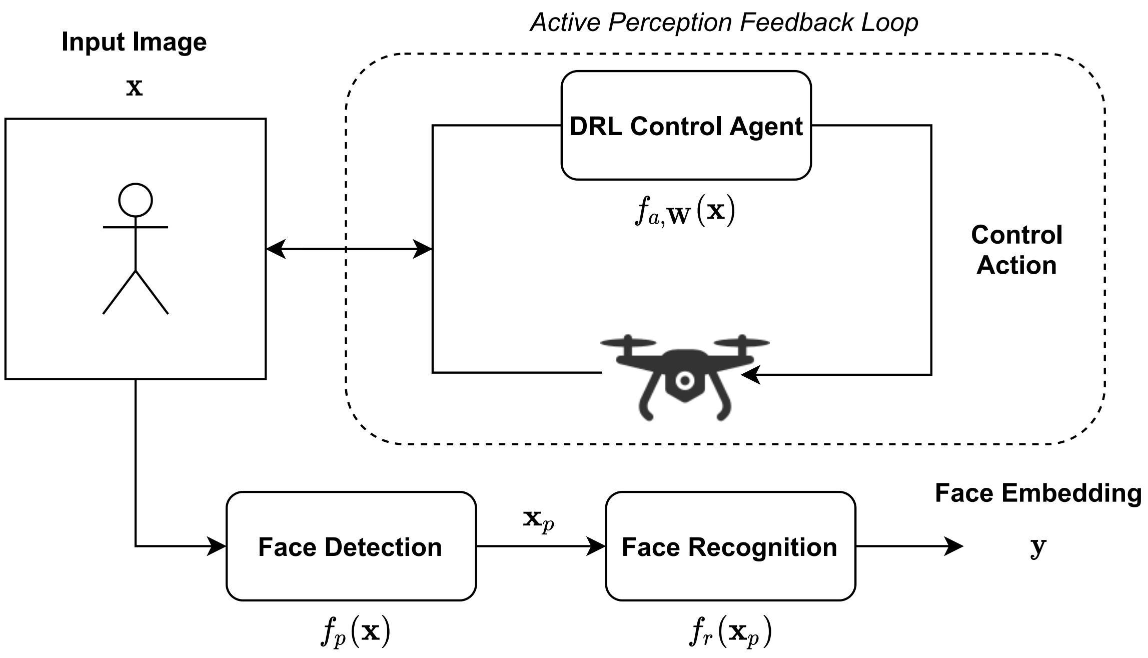


Figure 1: Proposed Active Perception Approach: A DRL agent is employed to issue control commands in order to acquire the most appropriate view for improve face recognition accuracy.

during inference the identity l of a person appearing in a novel image $\mathbf{x}$ is obtained as $l = l_k$, where

$$k = \arg\min_i ||f(\mathbf{x}_i) - f(\mathbf{x})||_2 \ (\forall (\mathbf{x}_i, l_i) \in \mathscr{X}_d). \tag{3}$$

The proposed method aims to learn an agent that can appropriately control a robot in order to re-acquire an input image $\mathbf{x}$ in which the depicted person can be more confidently identified, as shown in Fig. 1. To this end, another model $f_{a,\mathbf{W}}(\mathbf{x})$ is introduced, where $\mathbf{W}$ denotes the trainable parameters of the model. This model is responsible for controlling the position and orientation of the robot in order to recognize the human in the scene with the greatest confidence possible. Five possible actions are supported by this model:

1. *stay*, where the robot does not move and initiates the face recognition pipeline,
2. *move forward/backward*, where the robot moves forward/backward, and
3. *move left/right*, where the robot rotates and translates its position on a predefined arc either on the left or right.

All the actions translate into discrete actions in the simulation environment, e.g., moving forward/backward moves the agent 0.1m to the corresponding direction. Note that the face recognition pipeline is only employed when the agent issues the *stay* command. This can significantly reduce the computationally complexity of the employed pipeline, since both the face detection and recognition models run only when the control agent is confident enough that the depicted person can be indeed recognized. This is in contrast with other active vision approaches that require all models to run simultaneously, e.g., [119].

The proposed agent is trained using DRL. More specifically, the Proximal Policy Optimization (PPO) algorithm was used [132]. The reward used for training the DRL agent was defined based on the face recognition confidence of a pretrained face recognition model. If the person was not correctly identified, the agent received a reward of 0. Therefore, after identifying the embedding of the most similar person (k) in the database according to (3), the reward at time-step t can be defined as:

$$r_t = \begin{cases} c & \text{if } \|\mathbf{y}-\mathbf{y}_k\|_2 < a \\ 0 & \text{otherwise} \end{cases}, \tag{4}$$

where $\mathbf{y}_k = f(\mathbf{x}_k)$, c is the face recognition confidence, and a is a cut-off value for recognizing a person, i.e., if the Euclidean distance is larger than a, then we assume that the person has not been recognized. The face recognition confidence is calculated simply as the negative of the normalized Euclidean distance between the current embedding vector and the embedding vector of the most similar person in the database:

$$c = 1 - \frac{\|\mathbf{y}-\mathbf{y}_l\|_2}{a}. \tag{5}$$

Note that c is bounded between 0 and 1, since the Euclidean distance cannot exceed the value of a, due to the used cut-off threshold.

A deep convolutional neural network, receiving input images of 400×300 pixels, was used to implement the policy, i.e., $f_{a,\mathbf{W}}(\mathbf{x})$, as well as estimate the advantage value. A lightweight DL model was used to this end. The architecture of the model was the following: 2 convolutional layers with 16 (8×8) and 32 (4×4) filters respectively utilizing the *ReLU* activation function, one fully connected layer of 256 neurons and two output layers. The first one was responsible for providing the policy function $f_{a,\mathbf{W}}(\mathbf{x})$ function. This layer was composed of the same number of neurons as the number of available actions and employed the *softmax* function to provide the final action probabilities. The other one was used for implementing the critic function and was composed of one output neuron providing the current advantage. To constrain the advantage values the *tanh* activation function was employed for this branch.

Each training episode lasts 1.000 steps and the agent initially starts at a random position around the human model, which also facing at different directions in each episode. The simulation world consists of a square room, a human model at the center of it and a drone robot, which the DRL agent is controlling. After each time-step the agent must decide whether or not its position and orientation must be adjusted. For training the network we employed the Adam optimization algorithm with a learning rate of 0.0003, while a total for 10.000.000 steps where performed during the training.

2.2.4 Performance evaluation

The proposed method was evaluated under two different setups. On the first setup, the agent was trained to select one of the first three actions (“stay”, “move left” and “move right”). In this setup, the human was always initialized to be in front of the drone and correctly centered. The aim of this ablated setup was to evaluate the ability of the agent to control the movement in just one axis in order to increase the face recognition model’s confidence. In the second setup, the agent was allowed to select any of the available control actions, evaluating the ability of the proposed method to perform more complicated sequences of actions in order to improve face recognition accuracy.

The proposed method was also compared to two baselines. First, a face recognition pipeline was employed to evaluate the ability of existing approaches to detect and recognize humans at

Table 2: Evaluation for controlling one axis (Setup 1). Face recognition confidence is reported. A value of zero is used when a person is not correctly recognized.

Distance	**Static**	**Rule-based**	**Proposed**
1m	0.76	0.77	**0.78**
2m	0.55	0.77	**0.78**
3m	0.43	**0.78**	0.77
4m	0.19	**0.76**	**0.76**
5m	0	**0.77**	**0.77**
6m	0	0.63	**0.75**
7m	0	0	**0.76**
10m	0	0	**0.71**
15m	0	0	**0.6**8
20m	0	0	**0.48**

In this setup the drone is initialized at a distance of 20m, which decreases by 1m in every evaluation episode. We report the average face recognition confidence reached for 4 different human models at each distance.

different distances. This setup was called "static" in the conducted experiments. Then, we also employed an active perception enabled agent that uses rules. The rule-based agent employed a face detector to detect if a face exists in the scene. If a face is found it outputs the appropriate control commands to center it to its field of view based on the detected bounding box and then moves forward based on the face recognition model's confidence, until it reaches the maximum confidence. This method is called "rule-based" in the conducted experiments. For all methods we used ArcFace [23] for the face recognition and RetinaFace [24] for the face detection. Also, the database of known identities consists of one feature vector extracted from cropped frontal face images of 5 different human models that were used for the conducted experiments.

The experimental results for the first setup are reported in Table 2. In this setup, the drone was positioned at various distances in front of the human subject, ranging from 1m to 20m away. Using a static setup, where the drone does not move, allows for recognizing the persons only up to 4 meters. On the other hand, the rule-based based approach, which allows the drone to move closer to the subject at hand, enables confident recognition up to 6 meters. This demonstrates that active perception, even when implemented using simple rules, can indeed lead to improved perception accuracy. The proposed method outperforms all the other evaluated methods since it allows for confidently recognizing the persons even up to 15m, while it can work correctly even from larger distances (up to 20m).

Similar conclusions can be also drawn for the evaluation results, reported in Table 3, using the second setup. Again, the proposed method can significantly improve the view-invariance of face recognition, allowing not only for recognizing the persons at different distances, but also in a wide range of different angles, for some of which most face recognition pipelines typically fail. It is worth noting that at a distance of 7m only the proposed method manages to work correctly, while the provided face recognition accuracy is virtually the same with a robot that was initially placed in close distance in front of a human subject. Also, note that the proposed method does not need a face detector to actively perceive the surroundings, which can lead to significant performance improvements. Indeed, the proposed method runs on 180 FPS on

Table 3: Evaluation for controlling two axes (Setup 2). Face recognition confidence is reported. A value of zero is used when a person is not correctly recognized.

Angle	**Static**	**Rule-based**	**Proposed**
		3m	
0°	0.48	**0.77**	0.76
60°	0.24	0.32	**0.78**
120°	0	0	**0.78**
180°	0	0	**0.76**
240°	0	0	**0.79**
300°	0	0.14	**0.78**
		5m	
0°	0.18	0.53	**0.79**
60°	0	0.32	**0.79**
120°	0	0	**0.79**
180°	0	0	**0.77**
240°	0	0	**0.78**
300°	0	0.13	**0.79**
		7m	
0°	0	0	**0.78**
60°	0	0	**0.78**
120°	0	0	**0.77**
180°	0	0	**0.76**
240°	0	0	**0.77**
300°	0	0	**0.77**

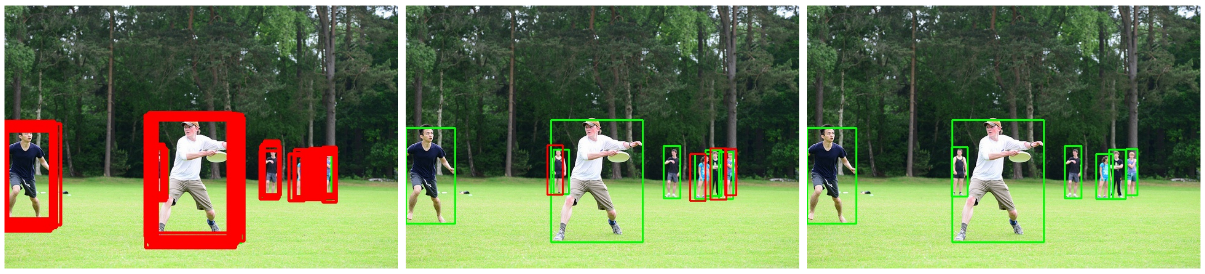
In this setup the drone is initialized at three different distances, while for each distance we also evaluated the performance of the agents at 6 different angles around the human model. We report the average face recognition confidence reached for 4 different human models at each distance.

average, while the rule-based approach runs on 62 FPS. A GPU-enabled workstation (8 GB VRAM, 9 TFLOPS) was used for measuring the performance of the evaluated agents.

2.3 Neural attention-driven Non-Maximum Suppression for person detection

2.3.1 Introduction and objectives

Non-Maximum Suppression (NMS) is a final refinement step incorporated in almost every visual object detection framework, performing the function of merging/filtering any spatially overlapping detected Regions-of-Interest (ROIs), i.e., bounding boxes, which correspond to the same visible object on an image. The problem it attempts to solve arises from the tendency of many detectors to output multiple, neighbouring candidate object ROIs for a single visible object, due to their implicit sliding-window nature. Thus, an NMS algorithm processes the raw



(a) Raw ROIs/detections. (b) ROIs/detections after applying GreedyNMS at 0.5 IoU. (c) ROIs/detections after applying the proposed method.

Figure 2: (a) Candidate ROIs/detections from Faster-RCNN in an image from the COCO [90] dataset. (b)(c) ROIs/detections after the application of GreedyNMS and the proposed method. Detections matched successfully to humans are colored green, while "incorrect" detections are colored red.

object detector outputs identified on an input image and attempts to filter out the duplicate ROIs.

Person/Pedestrian detection still remains a challenging task for most NMS methods, facing difficulties in identifying individuals especially within a crowd, due to various levels of occlusions. The majority of existing NMS methods are oriented towards fast execution, but person detection in human crowds requires a high degree of accuracy; this is critical for ensuring human safety in domains such as autonomous systems [16] [143] [72] [110].

The proposed NMS approach offers the following contributions:

- a novel reformulation of the NMS task for object detection as a sequence-to-sequence problem.
- a novel deep neural architecture for NMS, called *Seq2Seq-NMS*, relying on the Scaled Dot-Product Attention mechanism.
- a new, fast, efficient and GPU-based neural implementation of the low-level Frame Moments Descriptor (FMoD) [97], which is employed for feeding the proposed DNN with appearance-based representations of detected candidate ROIs.

An example, where our proposed method outperforms Greedy NMS on an image from the COCO dataset, is depicted in Fig. 2.

A summary of this work is provided hereafter. The corresponding preprint is listed below, and can be found in Appendix 8.2:

- [145] C. Symeonidis, I. Mademlis, I. Pitas and N. Nikolaidis, "*Neural attention-driven Non-Maximum Suppression for person detection*", TechRxiv preprint, 10.36227/techrxiv.16940275, 2021.

Despite being studied so far for the person detection task, the proposed NMS approach can obviously be also applied in other object detection problems.

2.3.2 Summary of state of the art

The de facto standard in NMS for object detection is GreedyNMS [36]. It selects high-scoring detections and deletes less confident neighbours, since they most likely cover the same object. An Intersection-over-Union (IOU) threshold determines which less-confident neighboring detections are suppressed. It is a simple, well-known, but limited method, leading to several attempts for replacing it with much improved alternatives.

In Soft-NMS [10], a rescoring function decreases the score of neighboring less-confident detections, instead of completely eliminating them, achieving better precision and recall rates compared to GreedyNMS. The authors experiment with Gaussian and linear weighting functions, which both require a hyper-parameter tuning similar to GreedyNMS. GossipNet [55] is a DNN designed to perform NMS, by processing the coordinates and scores of the detections. Overall, it jointly analyzes all detections in the image, so as not to directly prune them, but to rescore them. In [58], an attention module is applied with the task to exploit relations between the input detections, in order to classify them as duplicate or not. The authors in [91] proposed Adaptive-NMS, a dynamic thresholding version of GreedyNMS. A relatively shallow neural network predicts a density map and sets adaptive IoU thresholds in NMS for different detections according to the predicted density. An accelerated NMS method has been proposed in [11], allowing shorter inference times in exchange for a small performance drop, due to the large number of boxes that are likely to be over-suppressed.

GossipNet was modified in [144], for the specific case of person detection from aerial views, so as to jointly process visual appearance and geometric properties of candidate ROIs. The method exploited handcrafted descriptors encoding statistical ROI appearance characteristics, which were computed on the spatial distribution of edges or interest-points detected within each ROI. These distributions acted as a discriminant factor for identifying complete vs partial object silhouettes, since the silhouette of any person seen from an aerial view is rather similar in shape.

2.3.3 Description of work performed so far

In this work, NMS for object detection is first reformulated as a sequence-to-sequence task. This approach is highly related to the evaluation criteria established in object detection [90] [33], where the candidate ROIs identified on an input image are assumed to indirectly form a sequence, based on the scalar confidence score assigned to each of them by the detector (in descending order). Traditionally, evaluating a detector's accuracy on a known dataset involves an analysis of this sequence. At each step, a candidate ROI is processed and matched to a ground-truth object, if and only if: (a) their IoU is higher than a predefined threshold, and (b) that ground-truth object has not been previously matched to a higher-scoring candidate detection. In the case where both (a) and (b) are fulfilled, the candidate ROI is marked as "correct", otherwise it is marked as "false". In the special case where only (a) is fulfilled, the candidate detection is marked as "false", due to it being a "duplicate" detection. Thus, the position of a candidate ROI in the sequence can be a significant factor when taking the decision to classify it as a "duplicate" or not.

This emphasis in the ordering is shared with problems traditionally viewed as sequence-to-sequence ones. For instance, in machine translation, a sequence of words from one language must be transformed into a sequence of words in another language. The order of each word (*token*) in the sentence is crucial and can modify its meaning (*context*). Similarly, in object detection evaluation, although a candidate ROI (token) can be successfully matched to a ground-truth object, it can be classified as "duplicate" and therefore as "false", instead of being classified as "correct", due to the fact that a higher-scoring candidate detection, which has been positioned earlier in the sequence, has already been matched with the same ground-truth object.

Motivated by these notions, this work explicitly formulates the NMS task as a mapping from an input sequence of candidate ROIs to a corresponding output sequence with identical length. Let $\mathbf{R}^{in}$ be the input sequence of candidate ROIs, in descending order based on their scalar confidence scores assigned by the detector:

$$\mathbf{R}^{in} = [\mathbf{r}_1^{in}, ..., \mathbf{r}_N^{in} | r_i^{score_{det}} \geq r_{i+1}^{score_{det}}] \tag{6}$$

where $\mathbf{r}_i^{in} = [r_i^{x_{min}}, r_i^{y_{min}}, r_i^{x_{max}}, r_i^{y_{max}}, r_i^{score_{det}}]$ is an input candidate ROI expressed through its spatial 2D image coordinates, along with its corresponding score assigned by the detector, and N is the number of candidate detections. Let $\mathbf{R}^{out}$ be the output sequence of candidate ROIs, in descending order based on the scores assigned by the NMS method:

$$\mathbf{R}^{out} = [\mathbf{r}_1^{out}, ..., \mathbf{r}_N^{out} | r_i^{score_{NMS}} \geq r_{i+1}^{score_{NMS}}] \tag{7}$$

where $\mathbf{r}_i^{out} = [r_i^{x_{min}}, r_i^{y_{min}}, r_i^{x_{max}}, r_i^{y_{max}}, r_i^{score_{NMS}}]$ is an NMS-rescored candidate ROI. The proposed formulation of the NMS task can be expressed as:

$$\mathbf{R}^{out} = NMS(\mathbf{R}^{in}) \tag{8}$$

Building upon this novel view of the NMS task, the method proposed in this work, which we call *Seq2Seq-NMS*, receives as input a sequence of candidate ROIs, generated by an object detector, and extracts rich representations regarding their appearance and geometry. The appearance-based ROI representations are extracted based on a new, fast, efficient and GPU-based neural implementation of FMoD extraction, which has already proven its worth in NMS for person detection from aerial viewpoints in [144]. An example of the appearance-based ROI representation extraction is illustrated in Fig. 3. Since, a set of purely geometric attributes has previously proven effective as an input descriptor, in the context of the GossipNet neural architecture [55], a slightly similar, but enriched set of attributes has been devised, serving as an additional, geometric representation for each candidate ROI. Additional details can be found in the preprint 8.2 appended in this Deliverable.

The appearance-based and geometry-based representations are fed to a DNN which processes them in parallel, while mainly paying attention to spatially neighboring, higher-scoring candidates when analyzing each ROI. The Scaled Dot-Product Attention mechanism [156], originally proposed for machine translation tasks, is employed to this end, since it has been proven effective in various applications, such as image classification [30] or image generation [114].

Finally, the proposed DNN outputs a sequence of scalar scores, each one defining the context of a candidate detection. This is essentially the information that determines the final decision of whether the respective ROI should be classified as "correct" or as "potentially suppressed", after the NMS task has been completed. In the proposed formulation, the context of the i^{th} candidate detection is expressed through the corresponding output score, which is a classification probability $p_i : \{p_i \in \mathbb{R} | 0 \leq p_i \leq 1\}$ (1/0 means "correct"/"potentially suppressed", respectively). After the inference stage, simple thresholding can be applied on these output probabilities/scores, in order to decide which candidate detections should be retained.

2.3.4 Performance evaluation

The performance of Seq2Seq-NMS was evaluated on three separate datasets for the person detection task. In order to assess its accuracy regardless of the selected object detector, candidate ROIs from three different detectors were employed. These differ significantly in the way they

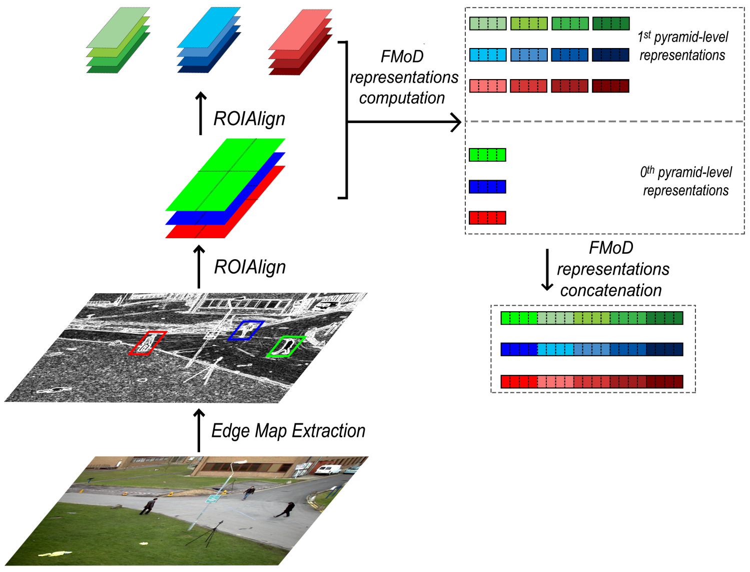

FMoD representations computation
ROIAlign
ROIAlign
Edge Map Extraction
1st pyramid-level representations
0th pyramid-level representations
FMoD representations concatenation

Figure 3: Computation of the visual appearance-based candidate ROI representations, by applying the fast FMoD implementation to an image with 3 ROIs and using 2 pyramid levels.

approach the object detection task in general. More details about the setting of the experimental evaluation is shown on Table 4.

Table 4: The setting of the experimental evaluation.

Dataset	**Detector**
PETS [37]	Joint Person Detector [148]
COCO [90]	Faster-RCNN [127]
CrowdHuman [135]	Yolov4 [9]

As it is shown in Tables 5, 6 and 7, the proposed Seq2Seq-NMS DNN achieves top accuracy on the $AP_{0.5}$ metric in all three datasets, being on par with GossipNet in CrowdHuman and outperforming all competing methods in the remaining two datasets. The results show that Seq2Seq-NMS can successfully capture interrelations between candidate detections for the person detection task, based both on their visual appearance and their geometry. The three datasets used for evaluation contain images with a great variety of visible persons density, ranging from images of individual people to photographs of large crowds, indicating that Seq2Seq-NMS is suitable for generic person detection.

Table 5: Comparison of different NMS methods on the PETS dataset, using detections from [148]. Bottom line reports on the proposed method.

Method	**Device**	**Max dets. = 400**			**Max dets.= 800**			**Max dets. = 1200**			**Max dets. = All**		
		$AP_{0.5}$	**$AP_{0.5}^{0.95}$**	**Average Inference Time (ms)**	**$AP_{0.5}$**	**$AP_{0.5}^{0.95}$**	**Average Inference Time (ms)**	**$AP_{0.5}$**	**$AP_{0.5}^{0.95}$**	**Average Inference Time (ms)**	**$AP_{0.5}$**	**$AP_{0.5}^{0.95}$**	**Average Inference Time (ms)**
Original NMS IoU>0.4	GPU	75.4%	31.7%	3.8	76.5%	31.3%	4.5	76.3%	31.2%	5.0	76.3%	31.2%	6.3
Original NMS IoU>0.5	GPU	73.1%	31.0%	4.1	73.6%	30.8%	5.7	73.6%	30.8%	7.5	73.6%	30.8%	9.8
Original NMS IoU>0.6	GPU	65.7%	28.6%	6.8	65.7%	28.6%	10.0	65.7%	28.6%	12.8	65.7%	28.6%	16.0
Original NMS IoU>0.4 (TorchVision)	GPU	75.6%	31.7%	0.4	76.4%	31.3%	0.5	76.4%	31.3%	0.8	76.4%	31.3%	0.8
Original NMS IoU>0.5 (TorchVision)	GPU	72.7%	30.8%	0.6	73.2%	30.8%	0.5	73.2%	30.8%	0.7	73.2%	30.8%	0.8
Original NMS IoU>0.6 (TorchVision)	GPU	65.1%	28.5%	0.5	65.1%	28.5%	0.5	65.1%	28.5%	0.7	65.1%	28.5%	0.8
Soft-NMS$_L$	CPU	76.6%	31.8%	27.1	76.5%	31.3%	64.0	76.3%	31.2%	98.5	76.3%	31.2%	143.3
Soft-NMS$_G$	CPU	77.4%	32.7%	23.8	76.7%	32.0%	54.4	76.0%	31.8%	89.5	75.8%	31.7%	154.7
Fast-NMS	GPU	75.3%	31.5%	3.2	75.0%	31.0%	1.5	74.5%	30.8%	2.8	74.4%	30.7%	3.5
Cluster-NMS	GPU	75.9%	31.4%	2.9	76.4%	31.3%	3.7	76.4%	31.3%	5.5	76.4%	31.3%	8.0
Cluster-NMS$_S$	GPU	75.4%	31.6%	4.8	74.0%	30.8%	3.8	73.6%	30.7%	5.0	73.0%	30.6%	7.3
Cluster-NMS$_D$	GPU	76.2%	31.6%	5.4	76.5%	31.2%	5.1	76.5%	31.1%	7.7	76.5%	31.1%	10.0
Cluster-NMS$_{S+D}$	GPU	76.4%	31.9%	3.7	75.9%	31.3%	5.4	75.2%	31.0%	8.4	74.9%	30.9%	12.1
Cluster-NMS$_{S+D+W}$	GPU	76.4%	31.9%	28.3	75.9%	31.3%	88.2	75.2%	31.0%	171.9	74.9%	30.9%	292.3
GossipNet	GPU	79.6%	34.4%	19.1	82.8%	35.6%	48.0	83.6%	35.9%	73.9	83.8%	36.1%	107.2
Seq2Seq-NMS	GPU	80.9%	36.5%	11.1	83.9%	**37.3%**	11.9	**84.7%**	**37.3%**	14.7	**-**	**-**	-

Regarding the $AP_{0.5}^{0.95}$ metric, Seq2Seq-NMS outperforms most competing methods but achieves top accuracy only on the PETS dataset. Most likely, this behaviour can be explained by the fact that the labels of the candidate detections for Seq2Seq-NMS training were created based on [90] using an IoU threshold of 0.5. Moving on to required inference running times, the proposed method is relatively slower than non-neural, mostly less accurate, GPU-executed algorithms. However, when compared against DNN architectures for NMS, such as GossipNet, the inference runtime of Seq2Seq-NMS seems less affected by the input sequence length (number of candidate detections), thus achieving faster inference when processing longer sequences, as shown in Tables 5 and 6.

Table 6: Comparison of different NMS methods on the COCO dataset using detections from Faster-RCNN [127]. Bottom line reports on the proposed method.

Method	**Device**	**Minitest set**		**Average Inference Time (ms)**
		$AP_{0.5}$	**$AP_{0.5}^{0.95}$**	
Original NMS IoU>0.4	GPU	65.4%	35.5%	4.3
Original NMS IoU>0.5	GPU	65.3%	35.8%	4.5
Original NMS IoU>0.6	GPU	63.2%	35.5%	4.8
Original NMS IoU>0.4 (TorchVision)	GPU	65.4%	35.6%	0.59
Original NMS IoU>0.5 (TorchVision)	GPU	65.2%	35.8%	0.60
Original NMS IoU>0.6 (TorchVision)	GPU	63.1%	35.5%	0.60
Soft-NMS$_L$	CPU	66.6%	36.9%	5.8
Soft-NMS$_G$	CPU	66.2%	36.7%	6.6
Fast-NMS	GPU	64.1%	35.3%	2.5
Cluster-NMS	GPU	65.4%	35.5%	4.9
Cluster-NMS$_S$	GPU	65.3%	36.0%	4.8
Cluster-NMS$_D$	GPU	65.5%	35.6%	7.6
Cluster-NMS$_{S+D}$	GPU	65.8%	36.6%	7.6
Cluster-NMS$_{S+D+W}$	GPU	65.8%	**37.6%**	9.7
GossipNet	GPU	66.9%	36.0%	6.1
Seq2Seq-NMS	GPU	**67.5%**	36.9%	10.1

Table 7: Comparison of different NMS methods on the CrowdHuman dataset using detections from Yolov4 [9]. Bottom line reports on the proposed method.

Method	**Device**	**Validation set**		**Average Inference Time (ms)**
		$AP_{0.5}$	**$AP_{0.5}^{0.95}$**	
Original NMS IoU>0.4	GPU	76.9%	44.8%	2.2
Original NMS IoU>0.5	GPU	80.6%	47.0%	2.9
Original NMS IoU>0.6	GPU	82.0%	48.2%	4.6
Original NMS IoU>0.4 (TorchVision)	GPU	77.1%	44.9%	0.4
Original NMS IoU>0.5 (TorchVision)	GPU	80.7%	47.0%	0.4
Original NMS IoU>0.6 (TorchVision)	GPU	82.0%	48.3%	0.4
Soft-NMS$_L$	CPU	82.5%	**49.3%**	30.8
Soft-NMS$_G$	CPU	81.7%	48.6%	39.8
Fast-NMS	GPU	81.0%	48.0%	1.6
Cluster-NMS	GPU	82.0%	48.2%	4.7
Cluster-NMS$_S$	GPU	80.3%	47.3%	3.4
Cluster-NMS$_D$	GPU	82.2%	48.6%	5.1
Cluster-NMS$_{S+D}$	GPU	81.4%	48.2%	5.5
Cluster-NMS$_{S+D+W}$	GPU	81.4%	48.2%	47.9
GossipNet	GPU	**83.7%**	49.0%	16.1
Seq2Seq-NMS$_{800}$	GPU	**83.7%**	49.1%	11.1

2.3.5 Future Work

Future extensions may focus on a training strategy suitable for various IoU thresholds, as well as on assessing method performance when using a different modality for describing the visual appearance of the cropped input detections (instead of edge maps). This may enable an extension of the proposed method to multiclass object detection tasks.

2.4 Active Face Recognition Using Synthesized Facial Views

2.4.1 Introduction and objectives

In recent years, the robotics community has started focusing in the field of active vision and exploration. More specifically, considering the problems of reconstructing [100, 22], [66] and recognizing an object, active vision methods are concerned with obtaining more information from the environment by actively choosing where and how to observe it using a camera [155]. This could be facilitated by camera motion, which plays a fundamental role in the reconstruction/recognition of the obtained scene [38], or by moving a mobile robot in different positions that offer different views of the candidate object. More specifically, active vision methods allow robots to conduct maneuvers, possibly ensuring limited energy consumption and motion complexity, in order to take novel views of a reconstructed/tracked scene or object.

An important application of vision in robotics is obviously face /person recognition. Deep Learning has been recently dominating the interest of researchers in this due to the superior performance achieved. Face recognition can be combined with methods that adopt an active approach in controlling the robot movement. Indeed, the face recognizer on a robot can obtain more robust results when perceiving the respective face in an active fashion. The robot that performs the recognition can move to capture the face from more informative views, at the expense of energy consumption and additional time needed. Synthesized novel views of faces whose images were acquired through appropriate acquisition systems could be used for robot guidance in an active face recognition setup. A synthesized facial view, could be used instead of an image captured by the robot's visual sensor. In other words, instead of having the robot move itself or its camera for capturing a novel view in a physical way, it can use a synthesized view, at least as an aid in its attempt to improve recognition.

In this subsection, we propose an active face recognition approach that utilizes facial views produced by photorealistic facial image rendering. Essentially, the robot that takes the snapshots and performs recognition selects the best among a number of candidate movements around the face of interest by simulating their results through view synthesis. In other words, once the robot, that is at a certain location with respect to the subject, acquires an image from the current location, it feeds the face recognizer with this image as well as with synthesized views that differ by $\pm$ degrees from the current view. Subsequently, it either stays in the same position or moves to the position that corresponds to one of the two synthesized views. The respective decision is based on the confidence of the three recognitions (on the real and the two synthesized views). In case of a "move" decision, it proceeds in acquiring a "real" image from its new location. The procedure repeats once again in the same manner, for this location. This line of work has started in the previous period and continued in this one.

2.4.2 Summary of state of the art

Multi-view Facial Image Synthesis for improving Face Recognition

A significant number of techniques for synthesizing facial images in novel views appeared in the recent years, as such images can have a number of applications, e.g., in improving face recognition accuracy. For example, since profile faces provide inferior recognition results compared to frontal faces, generative adversarial network (GAN) based methods for the frontalization of profile facial images [32] or generation of other facial views [64] have been proposed for enhancing the face recognition results. Furthermore, a method for the generation of frontal views from any view by a novel generative adversarial architecture called the Attention Selective Network (ASN) is described in [89]. Towards improving single-sample face recognition by both generating additional samples and eliminating the influence of external factors (illumination, pose), [152] presents an end-to-end network for the estimation of intrinsic properties of a facial image with recovery of the albedo UV map and the 3D facial shape. In [99] a facial image rendering technique is used both in the training and the testing stages of a face recognition approach. Finally, a method that produces photorealistic facial image views is described in [179]. The basic idea of this approach is that rotating faces in the 3D space and re-rendering them to the 2D plane can serve as a strong self-supervision. A 3D head model (obtained by utilizing the 3D-fitting network 3DDFA), accompanied by the projected facial texture of a single view, is being rotated and multi-view images of the face are rendered using the Neural 3D Differential Renderer [74] accompanied by 2D-to-3D style transfer and image-to-image translation with GANs to fill in invisible parts. This last state-of-the-art method was selected due to its robustness and photorealistic quality to be used in our method.

Active Face Recognition

Active-vision-based face recognition is an understudied field in current literature. A recent active method is described in [106] and comprises of 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. Authors in [119] propose a DL-based active perception method for embedding-based face recognition and examines its behavior on a real multi-view face image dataset. In [68] another active vision-based object recognition approach is presented, among other contributions. More a CNN-based approach is described that allows object recognition over arbitrary camera trajectories, (which generate multi-view image sequences) without requiring explicit training over the potentially infinite number of camera paths and lengths. This is done by decomposing an image sequence into a set of image pairs, classifying each pair independently, and then learning an object classifier by weighting the contribution of each pair. The method is then extended to the next-best-view problem in an active recognition framework. This is accomplished by training a second CNN to map from an observed image to next viewpoint and incorporated into a trajectory optimisation task.

It shall be noted that active methods mentioned above, as well as the proposed method, involve a face recognizer that is trained to recognize frontal or nearly frontal faces, while having to deal with input facial images obtained from an arbitrary view point, a fact that makes recognition challenging.

2.4.3 Description of work performed so far

Let us denote as database split G a set of facial images for the person identities that can be recognized. The simple Deep Face Recognition method [41] is used by our approach. This method first performs face detection using HOG features and template matching. Subsequently, it uses landmark detection to identify 68 facial landmarks and then warps the image so that the eyes and mouth are centered. Subsequently FaceNet [131] is used to evaluate a 128-dimensional feature vector from each centered image. Finally, the face recognizer performs recognition by evaluating the database G image that is the nearest neighbor to the input image, using euclidean distance on the corresponding feature vectors.

Simply put, the proposed active vision algorithm uses the face recognition confidence metric and facial images synthesized for view angles around the current view, in order to select the next robot action. Starting from an initial position, the robot can take one of the following three decisions: stay at the current position, move by x° to the right or move by x° to the left, in order to acquire a new image. Depending on the achieved recognition confidence, an additional movement, towards the same direction as the first one, might be decided.

More specifically, given a facial query image I_r, captured by the robot camera at the robot starting position, the face synthesis algorithm [179] is utilized to render/generate facial views in 2 different viewing angles i.e. -15° and $+15^\circ$ in pan with respect to the pan of I_r (and the same tilt as I_r). These two images are denoted by I_s^- and I_s^+ respectively. Then, the face recognizer is fed with these three images (one real, two synthetic ones). Depending on the image that generated the biggest face recognition confidence (FRC), the robot stays in its current position (if FRC was maximum in I_r) or moves -15° ($+15^\circ$) (if FRC was maximum in I_s^- (I_s^+)) and acquires through its camera a new image I_r^- (I_r^+). If a "stay" decision was taken, the algorithm outputs the ID of the person it recognized in I_r and terminates. If the robot moved, face recognition is performed again in I_r^- (I_r^+) and the obtained FRC is compared to a threshold. In case a high enough confidence was observed, the algorithm outputs the ID of the person it recognized in I_r^- (I_r^+) and terminates. If not, it tries yet another 15° step (movement) in pan, in the same direction as the first step. In more detail, it generates/synthesizes a facial view -15° ($+15^\circ$) in pan from the current pan value (and the same tilt), denoted as I_s^{--} (I_s^{++}), and evaluates (by calling the face recogniser) FRC on this synthetic image. If $FRC(I_r^-) > FRC(I_s^{--})$ ($FRC(I_r^+) > FRC(I_s^{++})$) the algorithm outputs the ID of the person it recognized in I_r^- (I_r^+) and terminates. Otherwise, the robot moves -15° ($+15^\circ$), acquires through its camera a new image I_r^{--} (I_r^{++}) and the algorithm outputs the ID of the person it recognized in this image. The procedure is summarized in Algorithm 1. It should be noted that the output ID is always obtained by a real image, i.e., an image captured by the robot camera. The synthesized views are only used to aid the robot in deciding whether to move in a new position (and acquire a new image there) or stay in the current position. The rationale is that in case the initial robot position is far from a frontal or nearly frontal one, the algorithm will direct it to move towards a positions which is closer to a frontal one. Obviously, the procedure can be generalized to include additional steps (movements), i.e., more than the two movements it currently has. It can also work, in the same way, for tilt.

2.4.4 Performance evaluation

For the evaluation of the proposed active face recognition approach, the following experiment is being conducted in the **HPID** dataset [43] which is a head pose image database of 2790 face images of 15 persons in variations of pan and tilt angles from -90° to $+90^\circ$ degrees. Two series of images (each having a distinct pose) were captured for each person, having 93 images in each series. We use the setup "Set 1" in [119] to allow comparison with the active recognition method presented therein. In this setup a test subset of 25% of the persons (4 persons) is selected. Note that, unlike the method in [119] our proposed algorithm requires no training. Images of the selected subjects were divided into a database split and a query (test) split. Obviously, the database split contains different facial images from those used

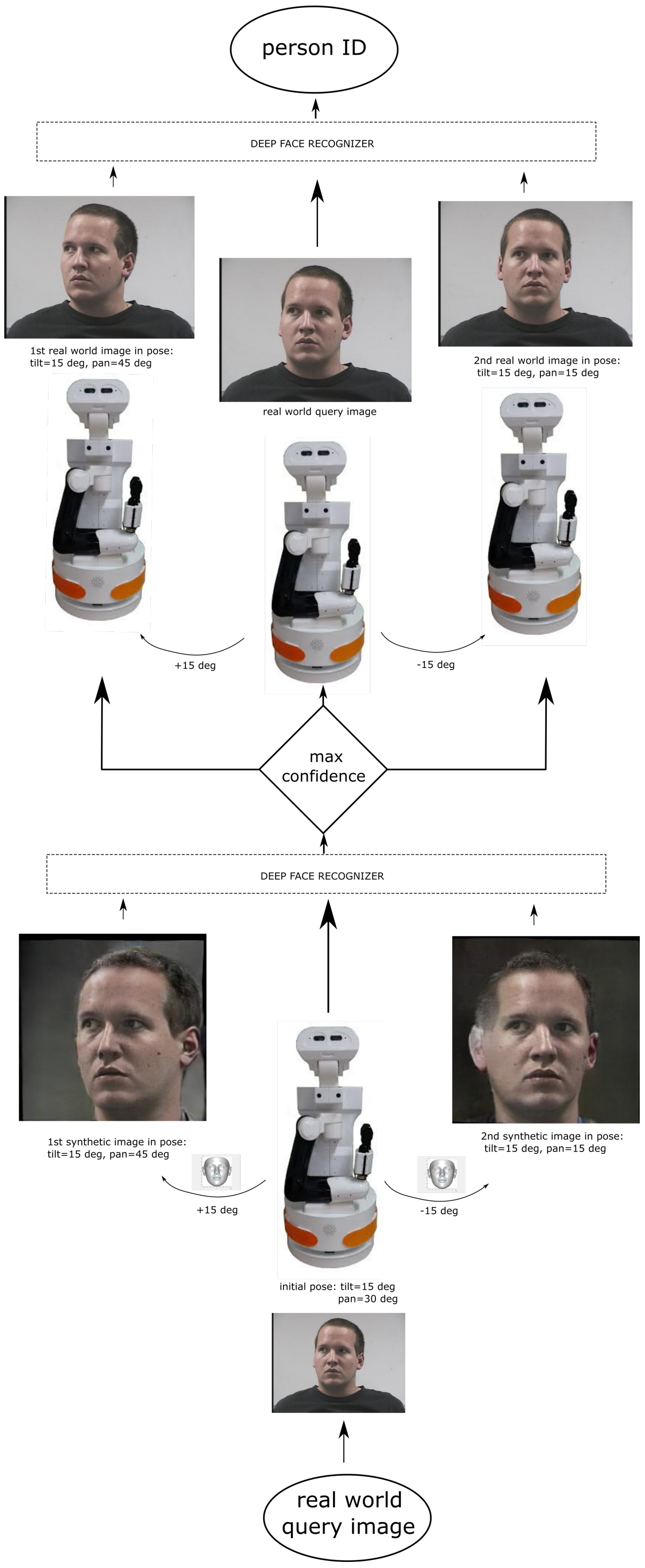

person ID
DEEP FACE RECOGNIZER
1st real world image in pose: tilt=15 deg, pan=45 deg
real world query image
2nd real world image in pose: tilt=15 deg, pan=15 deg
+15 deg
-15 deg
max confidence
DEEP FACE RECOGNIZER
1st synthetic image in pose: tilt=15 deg, pan=45 deg
2nd synthetic image in pose: tilt=15 deg, pan=15 deg
+15 deg
-15 deg
initial pose: tilt=15 deg pan=30 deg
real world query image

Figure 4: Active Vision Pipeline illustrated for an input query example.

Algorithm 1: Active Vision Algorithm

```
1  **Input**: Query real world facial image $I_{rw}$ ;
   **Result:** $Person_{ID}$
2  $I_{rw}^{encoding} \leftarrow$ extract deep feature vector
3  $FRC \leftarrow$ do face recognition
4  for $I_i \in S$ do
5      Synthetic Camera Position in $Pose_i(pan = \pm 15°, tilt)$;
6      $I_i \leftarrow$ render the image
7      $I_i^{encoding} \leftarrow$ extract deep feature vector
8      $FRC_i \leftarrow$ do face recognition
9  if $FRC \geq FRC_i$ then
10     $Person_{ID} \leftarrow$ do face recognition
11 else if $FRC \leq FRC_i$ then
12     $I_{irw}^{encoding} \leftarrow$ extract deep feature vector
13     $Person_{ID} \leftarrow$ do face recognition
14
```

for query. This was done by choosing a specific pan range which entered the database (i.e. the set of images G that the face recognizer uses to decide upon the ID of the query image through a nearest neighbor classifier) and we did queries with different pans. These query images are meant to be the images captured by the robot camera in its initial position. More specifically, the query subset contains face images with tilts $[-30°, -15°, 0°, 15°, 30°]$ and pans $[-90°, -75°, -60°, -45°, -30°, 30°, 45°, 60°, 75°, 90°]$. The database split contains facial images with tilt in angles $[-30°, -15°, 0°, 15°, 30°]$ and pans $[-15°, 0°, 15°]$, i.e. nearly frontal images. With the used setup we are simulating active vision where the robot is moving only in the pan direction. The results (in terms of recognition accuracy) are illustrated in Table 8. The second line in this table presents the result of the static equivalent of our approach, in which only the initial query facial image is used by the same recogniser involved in the active approach, i.e. the one in [41]. As can be seen in Table 8, our proposed active method outperforms both the active face recognition technique [119] and the static variant.

Table 8: Active Face Recognition Accuracy

Algorithm	**Accuracy**
[119] Active	62.2%
[41] Static	58.33%
Proposed Active	91.09%

2.4.5 Future Work

As future work, we plan to evaluate the proposed algorithm in a larger dataset and also by creating a simulation with human models in Webots [163, 101].

2.5 Utilizing a Realistic Image Data Generation Framework for the Training and Evaluation of Human-centric Methods

AUTH has developed a method that generates mixed image datasets for training and evaluation of: (a) pose estimation, (b) person detection, (c) identity recognition methods. Our method, which utilizes real background images and DL-generated 3D human models, was first introduced and described thoroughly in Section 2.2 of D6.1. Employing our method, a large-scale mixed dataset was generated. Details about the dataset are provided in Section 2.1 of D6.2. In this subsection we will present the experimental evaluation which was conducted on our dataset for the three corresponding computer-vision tasks. We believe that the experimental evaluation proves that the generated data are suitable for training DL models in most cases.

2.5.1 Introduction and objectives

The scale, diversity and quality of data used for training supervised deep learning methods have a major impact on their performance. Algorithms that are intended to be deployed in real-life conditions are usually trained on multiple datasets in order to improve their generalization abilities and ensure their robustness.

COCO [90] and Cityscapes [20] are two of the examples of large and diverse datasets providing annotations for training and evaluation of deep learning algorithms for computer vision tasks, such as object detection, human pose estimation, semantic image segmentation, etc. Collecting and manually annotating such amounts of data is usually a challenging and exhausting task, requiring a lot of time and resources. In the case of visual human-centric analysis, the collection and distribution of such data may also face restrictions due to the legislation regarding privacy. An alternative approach for collecting training data in a more automated manner [39], [160] is to generate them through a simulator. On the downside, these datasets often suffer in terms of realism and detail and/or are expensive to generate, requiring artists to carefully design specific models and environments.

Furthermore, apart from using synthetic data to create new or enrich existing datasets, synthetic data are also often implicitly employed during the validation of complex systems, such as robots [101]. These systems are often designed and tested in simulation environments before being deployed in real life conditions. However, deep learning methods trained on datasets containing only real data often exhibit an unstable behaviour or fail to perform adequately, as they have not adapted to the visual differences between the simulated and the real world data, as we also experimentally demonstrate in this work.

To overcome these limitations, we proposed an effective and low-cost data generation method for human-centric tasks that generates realistic and diverse data for person detection, pose estimation and face recognition. The method is capable of augmenting existing image datasets, eliminating additional costs that often occur, as well as potential data collection restrictions regarding privacy legislation. The proposed method consists of two stages. First, 3D human models are generated and skeleton-related information is extracted for each of them. For this task, we employ natural images, avoiding the need for using hand-crafted 3D models. Then, the proposed pipeline proceeds with the second stage, which concerns data generation through careful blending of real background images and 3D human models. The appropriate constraints are enforced during this step to ensure realistic placement of objects. The procedure is illustrated in Fig. 5. Finally, the method provides automatically-generated and detailed annotations. By employing our method, we generated a dataset that contains 50,000 images.

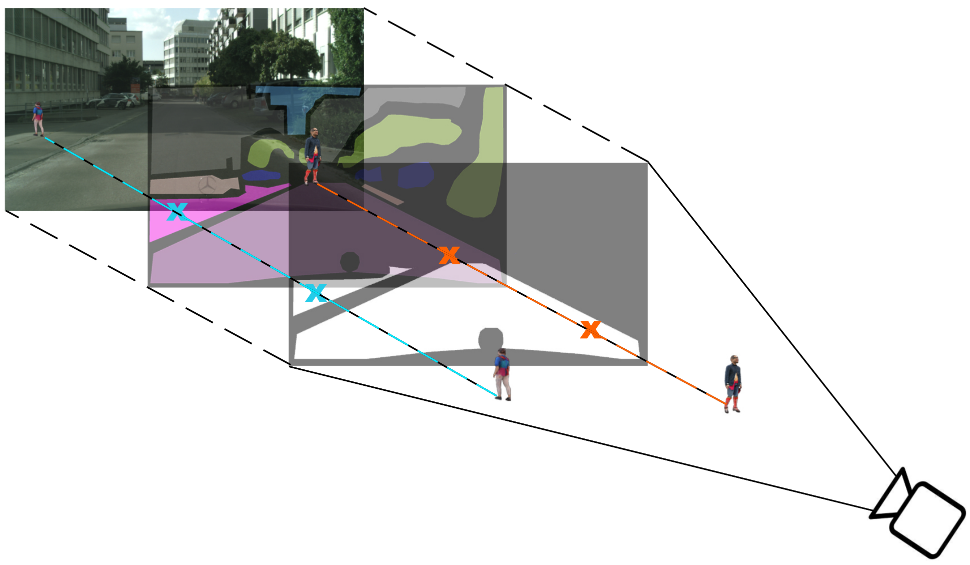


Figure 5: Visualization of the data generation process. The depicted background image is derived from the Cityscapes [20] dataset.

2.5.2 Description of work performed so far

Within WP3, an experimental evaluation was conducted using the dataset, in order to examine whether (a) its images are visually realistic enough (b) the performance of DNN methods on the three human centric perception tasks is improved when the data are used in their training. The procedure and its results are described in the following paragraphs whereas more details about the proposed pipeline can be found in the associated paper:

- [146] C. Symeonidis, P. Nousi, P. Tosidis, K. Tsampazis, N. Passalis, A. Tefas and N. Nikolaidis, "*Efficient Realistic Data Generation Framework Leveraging Deep Learning-Based Human Digitization*", in Proceedings of the 22nd Engineering Applications of Neural Networks Conference (EANN), 2021

The first conducted set of experiments on person detection aimed to evaluate the ability of models pretrained on the COCO dataset [90] to effectively detect persons that appear in the generated data, examining in this way whether the generated data are realistic enough. The results are reported in Table 9, for the person class of the COCO validation set as well as for the test set of the generated dataset. All detectors exhibited excellent precision on person detection tasks, validating the aforementioned hypothesis. It should be noted that, apart from CenterNet, the detectors generalize better on the proposed dataset compared to the validation set of COCO.

Table 9: Person Detection Evaluation (AP@0.5 is reported)

Model	**COCO**	**Synthetic**
CenterNet (RS-18)	38.2%	39.0%
SSD-512 (RS-50)	41.9%	86.8%
SSD-512 (VGG16)	41.5%	88.1%

"RS" refers to ResNet.

Similar results were obtained for the pose estimation using the lightweight OpenPose model [113]. The results are reported in Table 10. For these experiments, instead of evaluating the precision of different pretrained models, we evaluated the effect of using only real data for training ("R"), using only synthetic data ("S") and combining both of them for the training process ("R+S"). The training process ran for 140,000 iterations for all models following the setup proposed in [113].

First, note that the ability of pretrained pose estimators to generalize to the synthetic data is validated ("R" row). Then, training only with synthetic data leads to perfect generalization on the test set of the synthetic data, but fails to generalize on real data. On the other hand, when both synthetic and real data are combined, we observe a significant improvement on the precision on the synthetic test set, compared to using only real training data. A small decrease is observed on the COCO test, but this is probably due to using a lightweight model architecture with relatively low learning capacity.

Table 10: Human Pose Estimation Evaluation (AP@0.5 is reported)

Data	**COCO**	**Synthetic**
R	37.7%	52.7%
S	1.2%	93.3%
R+S	34.0%	88.7%

"R" refers to using real data for training (COCO dataset), "S" refers to using the proposed synthetic data and "R+S" to using both of them.

Finally, we evaluated the generated data for the task of face recognition, where we observed several interesting phenomena. The results are reported in Table 11. The used models were a) pretrained on the MSCeleb (trained for 120 epochs), b) trained both on the combined MSCeleb and the proposed dataset (trained for 120 epochs), c) pretrained on the MSCeleb data and fine-tuned on the combined dataset (trained for 5 epochs). An inverted residual model (IR-50) was used as feature extractor [8]. The evaluation datasets consisted of: a) 5,000 positive and 5,000 negative pairs of images taken from the MSCeleb dataset, b) 5,000 positive and 5,000 negative pairs of images of the test set of the generated synthetic data, c) the LFW dataset [62] and d) the CFP_FF dataset [133].

Table 11: Face Recognition Evaluation (face verification precision (%) is reported)

Training Setup	**MSCeleb**	**Synthetic Data**	**LFW**	**CFP_FF**
R	95.01%	90.63%	99.80%	99.67%
R+S	98.44%	99.75%	99.20%	99.05%
R+S (finetuning)	97.70%	99.47%	99.38%	99.27%

Several interesting conclusions can be drawn from the results reported in Table 11. First, note that the synthetic data are actually "harder" for a network trained on real data ("R" row). This can be explained if we consider the loss of facial detail that often occurs in the 3D models. Quite interesting, this is also the case for real face recognition systems that operate *in-the-wild* with low resolution cameras (e.g., footage from CCTV systems), as well as for robotics systems that are validated using simulators. Then, we observe that training using the combined real and synthetic set leads to significant improvements both for the real and synthetic data, without any significant impact on how the model generalizes on other real datasets (e.g., LFW and CFP_FF). Finally, we observed similar positive results even when we only fine-tuned the model trained for 5 epochs using the combined set (last row), demonstrating that we can obtain similar improvements with full training ("R+S") only spending a fraction of time for training (5 epochs instead of 120).

2.6 Knowledge Transfer for Human Detection in Fields

2.6.1 Introduction and objectives

Object Detection combines the tasks of classification and localization, i.e., it refers to finding *what* objects are pictured in an image, as well as *where* in the image they are located. In the case of multiple object, multiple class object detection, a generic object detector should be able to detect an unknown number of objects belonging to a number of different classes. Depending on the training dataset, these classes can include people, animals, inanimate objects, etc. Such datasets include the widely popular PASCAL VOC [33] and MS COCO [90] object detection benchmarks, containing objects from 20 and 80 classes respectively. Deep Learning brought significant improvements both in terms of effectiveness and efficiency and currently the top performing object detection methods on these challenging benchmarks are all based on Deep Convolutional Neural Networks (CNNs).

Despite improvements in these detectors on well-known object detection benchmarks, deploying them on new datasets highlights the domain adaptation problem. Domain adaptation refers to the process of alleviating the domain-shift between a source and target domain, i.e., how to effectively deploy a detector trained on a source domain onto a target domain, which differs from the source in some way. Recent works in domain adaptation for object detection propose a progressive shift towards the target domain [65, 57]. A somewhat more straightforward approach to domain adaptation is incremental learning [164]. In any case, knowledge transfer is of significant importance when training a detector on a new dataset.

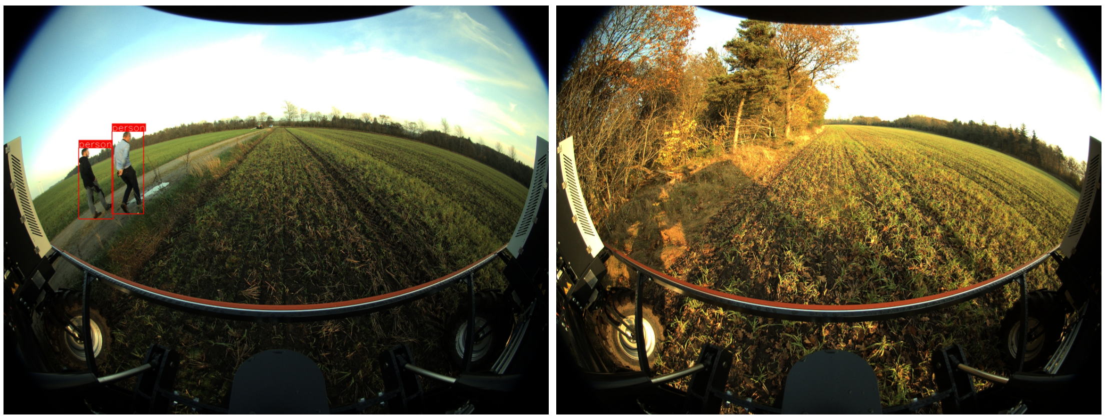
2.6.2 Summary of state of the art

Single-stage detectors have been shown to perform about as well as their two-stage, heavy-weight counterparts, while running at much faster speeds. The seminal methods of YOLO [125] and SSD [92] inspired many recent works which utilize the anchor-based, single-stage architectures proposed by them. Multiple variations of both methods have emerged, attempting to improve either the speed or accuracy, some of the most popular of which use different backbones to accommodate different tasks, like SSD Mobilenet [63].

Anchor-free object detectors aim to tackle issues arising from the use of predefined anchors, such as the need for thousands of such anchors in order to train dense object detectors, or the tedious hyperparameters they introduce, like the size, aspect ratio etc. CenterNet [31] is one such anchor-free object detector, taking into consideration the center of objects as well as the corners, to detect each object as a triplet. CenterNet also proposed three variants with different feature extraction backbones to improve detection speed.

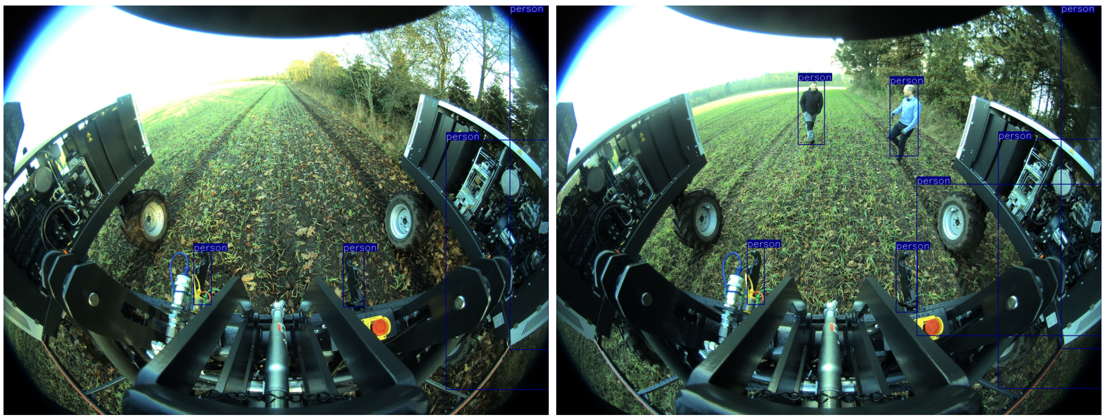
2.6.3 Description of work performed so far

A Robotti was deployed by AGI to collect images with a front and back camera, in a realistic scenario to mimic the images that the robot might encounter in the agricultural use case. Of the 818 collected images, 13 were discarded as they depicted unwilling participants to comply with GDPR. The remaining images were annotated with bounding boxes, where one bounding



(a) (b)

Figure 6: Examples of collected images: (a) Image depicting two humans, annotated with bounding boxes, (b) Image depicting no humans.



(a) (b)

Figure 7: Examples of false positive detections.

box corresponds to one depicted person. The LabelImg[1] tool was used for the annotation, which outputs annotations in PASCAL VOC .xml format. Figure 6a is an image from this dataset annotated with two bounding boxes for the two depicted humans. In total, 158 images contained people, and 647 images did not. The latter were annotated with an empty bounding box list, to be used as negative samples in object detection algorithms. Figure 6b is an example of an image from this dataset containing no humans.

An SSD network, pretrained on the WIDER Person dataset and available for download on the OpenDR server, was evaluated on this new dataset. The precision at 0.5 IOU dropped from 75.8 for WIDER Person to 24.8 for this dataset. The loss in performance can be attributed both to the detector's inability to detect small objects, as well as to the distribution shifts caused by the use of different optics in the camera lens compared to those used in the dataset used for the original training of the model. Figures 7a and 7b provide examples of detection on this dataset, where the existence of false positive is evident. This observation signified the need to incorporate images from this new dataset into the training process of detectors, to deal with the problem of domain transfer.

[1]`https://github.com/tzutalin/labelImg`

Method	Train Dataset	Human only	Human + negative	FPS
SSD	WIDER	43.7	26.2	23.6
SSD	VOC	53.5	42.2	23.6
SSD	COCO	78.3	68.0	23.6
SSD - MobileNet	VOC	40.8	18.8	35
SSD - MobileNet	COCO	60.7	42.3	35
CenterNet	VOC	43.4	28.6	16.1
CenterNet	COCO	63.1	54.8	16.1
YOLOv3	VOC	63.4	60.9	15.2
YOLOv3	COCO	78.9	74.7	15.2

Table 12: Evaluation of pretrained detectors on the collected dataset depicting humans in field.

2.6.4 Performance evaluation

An extensive evaluation of pretrained detectors of the SSD [92], YOLOv3 [126] and CenterNet [31] families, was conducted for this dataset. The results are summarized in Table 12 in terms of precision at 0.5 IoU and FPS on Jetson AGX. The detectors are trained on either the PASCAL VOC [33] and MS COCO [90] object detection benchmarks, containing objects of 20 and 80 classes respectively. The SSD detector is also trained on the WIDER Person dataset specifically for person detection. Finally, the MobileNet version of SSD [63] is also evaluated.

As expected, the addition of images without people highlights the false positive accumulation, due to the unseen backgrounds present in the dataset. Detectors trained on COCO seem to perform significantly better than those trained on VOC, which can be attributed to the wider range of appearance in people in the larger COCO dataset. The object scale in COCO is also more varied, containing people as small as 10 pixels in height. The YOLOv3 detector in general performs the best, but is the slowest of the evaluated detectors on the Jetson AGX. The SSD MobileNet variant, especially when trained on COCO, seems to give off the best speed/accuracy trade-off. Even so, the drop in precision is significant when considering negative-only samples.

2.6.5 Future Work

Based on the experimental study conducted for this dataset, further training of the detectors is necessary to avoid false positive detections as well as to increase the true positive ratio. Knowledge transfer from the COCO dataset seems to be the most promising direction for our future research.

2.7 Remote Multilinear Compressive Learning with Adaptive Compression

2.7.1 Introduction and Objectives

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 either 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 introduction of spatial-multiplexing cameras, over the past decade, Compressive Sensing (CS) [13] 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 [12, 29].

In applications that employ hybrid solutions, data transmission and informational content throughput, i.e., the amount of data content that can be analyzed under a given period, play an important role. Compressive Learning (CL) is a hybrid framework that combines Compressive Sensing and Machine Learning. With Compressive Learning, the signal is compressively acquired and the inference model is built on top of the compressed measurements. In Compressive Learning, smaller compressed measurements, i.e., higher compression rates, require less transmission bandwidth. However, the level of signal compression affects the inference performance of a Compressive Learning system, with higher compression rates often associated with higher losses of content, thus lower classification or detection accuracy. Since the transmission network's condition can vary over time, many modern communication protocols provide implementations for adaptive data transmission to maximize the throughput and minimize energy consumption. By enabling Compressive Learning systems to acquire signals and make inferences at an adaptive compression rate that is computed based on the adaptive transmission rate, we can maximize the informational content of the entire application.

A summary of this work is provided hereafter. The corresponding publication is listed below, and can be found in Appendix 8.3:

1. D. T. Tran, M. Gabbouj and A. Iosifidis, "*Knowledge Distillation by Sparse Representation Matching*", IEEE Internet-of-Things 2021.

2.7.2 Description of the work performed so far

A Multilinear Compressive Learning (MCL) system [151] consists of three modules: the Multilinear Compressive Sensing (MCS) component, the Feature Synthesis (FS) component and the task-specific neural network. The first 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). The second component in MCL is the Feature Synthesis (FS) component, which takes the compressed measurements as inputs and produces a tensor features.

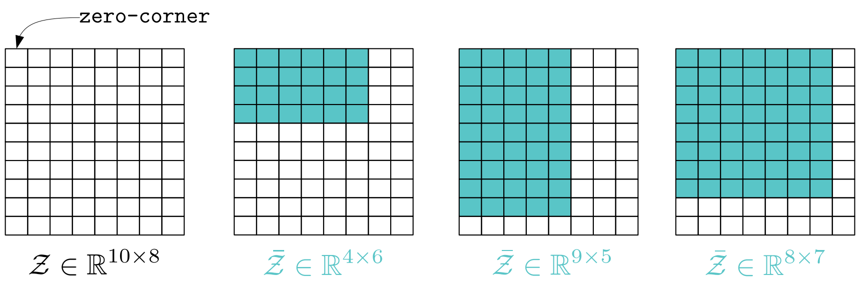

zero-corner
$\mathcal{Z} \in \mathbb{R}^{10\times 8}$
$\bar{\mathcal{Z}} \in \mathbb{R}^{4\times 6}$
$\bar{\mathcal{Z}} \in \mathbb{R}^{9\times 5}$
$\bar{\mathcal{Z}} \in \mathbb{R}^{8\times 7}$

Figure 8: Illustration of compressed measurements of high compression rates $\bar{\mathscr{Z}}$ constructed from the original compressed measurement $\mathscr{Z}$.

The last component in MCL is the task-specific neural network, which takes the output of the FS component and generates the predictions.

In compressive sensing, especially in remote compressive sensing applications, the ability to adaptively adjust the compression rate, thus the signal fidelity, can greatly improve the informational content throughput of the applications. This is because many existing communication protocols can support adaptive transmission rates to maximize the network's throughput. By combining the adaptive transmission rate feature and a compressive sensing device capturing signals at an adaptive compression rate, the data content throughput can be maximized. From the hardware point of view, it is infeasible to build an MCS sensor that utilizes multiple sets of sensing operators to operate at different compression rates. However, with a single set of sensing operators $\{\Phi_k \,|\, k = 1,\dots,K\}$ that produces a compressed measurement $\mathscr{Z} \in \mathbb{R}^{M_1\times\dots\times M_K}$, we can obtain a compressed measurement $\bar{\mathscr{Z}}$ of smaller dimensions ($\bar{\mathscr{Z}} \in \mathbb{R}^{m_1\times\dots\times m_K}$ with $m_k \le M_k, \forall k$) that corresponds to a (higher) compression rate by forming $\bar{\mathscr{Z}}$ from the elements of $\mathscr{Z}$, i.e.:

$$\bar{\mathscr{Z}}[i_1,\dots,i_K] \in \mathscr{Z}, \quad \forall\, i_k \le m_k \,|\, k = 1,\dots,K, \tag{9}$$

where $\bar{\mathscr{Z}}[i_1,\dots,i_K]$ denotes the element of $\bar{\mathscr{Z}}$ at position $(i_1,\dots,i_K)$.

From a single compressed measurement $\mathscr{Z}$, to construct multiple instances of $\bar{\mathscr{Z}}$, each of which retains the amount of signal information approximately proportional to its size, we can optimize the set of sensing operators $\{\Phi_k \,|\, k = 1,\dots,K\}$ such that $\mathscr{Z}$ has the following semantic structure: elements in $\mathscr{Z}$ carrying the most relevant information for the learning task concentrate around the zero-corner of $\mathscr{Z}$, i.e., the corner at position $(1,\dots,1)$. In addition, the elements of $\mathscr{Z}$ are arranged according to their importance, with the ones closer to position $(1,\dots,1)$ being more relevant. With this semantic structure, a compressed measurement $\bar{\mathscr{Z}}$ that corresponds to a higher compression rate can be constructed from $\mathscr{Z}$ as follows:

$$\bar{\mathscr{Z}}[i_1,\dots,i_K] = \mathscr{Z}[i_1,\dots,i_K],\ \forall\, i_k \le m_k \,|\, k = 1,\dots,K. \tag{10}$$

The construction of $\bar{\mathscr{Z}}$ is illustrated in Figure 8. Since the proposed semantic structure allows the construction of $\bar{\mathscr{Z}}$ from contiguous elements of $\mathscr{Z}$, generating $\bar{\mathscr{Z}}$ is computationally efficient since this step only requires accessing contiguous memory locations, which is more hardware-friendly compared to accessing non-contiguous elements. It should be noted that $\bar{\mathscr{Z}}$ is constructed on the client side, before being transmitted to the computing server. The FS

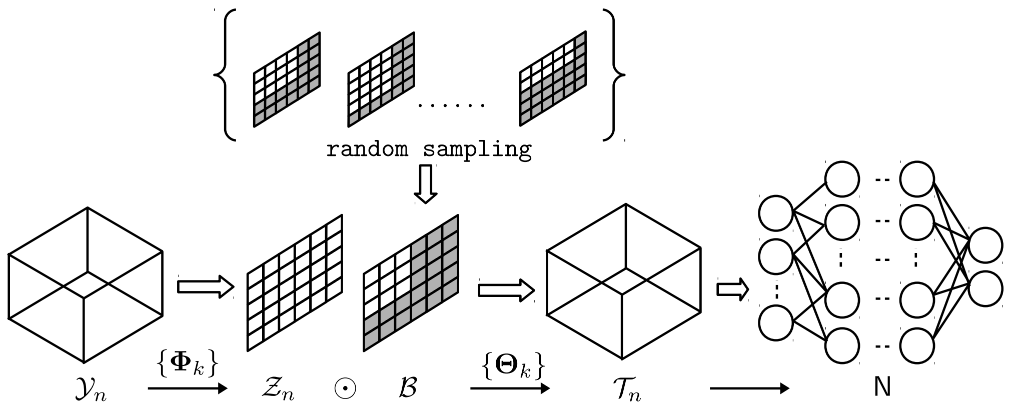


Figure 9: Illustration of the proposed training method with stochastic binary mask $\mathscr{B}$.

component and the task-specific neural network N are implemented on the server side to make predictions with incoming variable-sized compressed measurements. Thus, the FS component must be able to handle inputs with variable dimensions. A simple approach is to set the input dimensions of the FS component to the maximum dimensions of incoming compressed measurements, i.e., the size of $\mathscr{Z}$, and the incoming compressed measurements are appropriately zero-padded to have the same dimensions as $\mathscr{Z}$.

To this end, the proposed remote MCL model with an adaptive compression feature has the following optimization criteria: (i) the MCS device captures $\mathscr{Z}$ that has the aforementioned semantic structure, and (ii) the server side utilizes a single model instance to make predictions with variable-sized compressed measurements. To fulfill both criteria, the effect of variable compression rates is simulated during stochastic optimization by using a stochastic structural dropout strategy. More specifically, after all components of the MCL model are initialized, a stochastic gradient descent-based optimizer is used to train all parameters with the following objective:

$$\underset{\{\Phi_k\},\{\Theta_k\},\Omega}{\operatorname{argmin}} \quad \sum_{n=1}^{N} L\Big(f_{\mathsf{N}}(f_{\mathsf{FS}}(\mathscr{Z}_n \odot \mathscr{B})), \mathbf{c}_n\Big) \tag{11}$$

where $\odot$ denotes the element-wise multiplication operator. f_{FS} and f_{N} denote the mathematical functions induced by the FS component and task-specific neural network N, respectively. Ω denotes the parameters of the task-specific neural network. $\mathscr{B} \in \mathbb{R}^{M_1 \times \dots \times M_K}$ is a random binary tensor with the elements defined as follows:

$$\mathscr{B}[i_1,\dots,i_K] = \begin{cases} 1 & \text{if } i_k \leq m_k^{(r)} \\ 0 & \text{else} \end{cases} \quad \forall k = 1,\dots,K \tag{12}$$

where $m_k^{(r)}$ denotes an integer value randomly drawn from the set $\{M_k^{(\min)}, M_k^{(\min)}+1,\dots,M_k\}$ for all $k = 1,\dots,K$. $M_k^{(\min)}$ and M_k denote predefined minimum and maximum values for the k-th mode of the compressed measurement tensor.

The proposed training process with stochastic binary mask $\mathscr{B}$ is illustrated in Figure 9. The intuition behind the stochastic dropout mask is as follows: with the binary mask that stochastically changes its values during optimization, the optimizer is instructed to optimize all parameters of the model so that the classification error is minimized for *any compressed measurement dimension that lies within the minimum and maximum values.* In other words, the MCL model is implicitly trained to perform sensing and learning with a compressed measurement of size $m_1^{(r)} \times \ldots \times m_k^{(r)} \ldots \times m_K^{(r)}$, with $m_k^{(k)}$ being randomly defined for $k = 1, \ldots, K$ during stochastic optimization.

On a final note, with the proposed adaptive compression mechanism, the computing server is not strictly constrained to use a single instance of the FS component and task-specific neural network. However, using separate instances of the FS component and task-specific neural network for every supported compression rate could incur a high computational cost on the server side. In cases where the server has sufficient computational power, the overall performance can be enhanced by using a dedicated FS component and a dedicated task-specific neural network for each compression rate. That is, after solving Eq. (11), we can freeze parameters of the MCS component and finetune server-side components (FS and N) for each supported compression rate.

2.7.3 Performance Evaluation

Experiments were conducted using a face recognition dataset: CelebA [94]) to evaluate the efficiency of remote MCL model with the adaptive compression feature in comparisons with the original MCL model. All images were resized to the resolution of 32×32 pixels, i.e., $\mathscr{Y} \in \mathbb{R}^{32\times32\times3}$. In the following, the results produced by the original MCL training method described in [151] are denoted as `single-rate`, and results from the proposed one-shot training method are denoted as `adaptive-rate`.

For the `single-rate` experiments, multiple MCL models were trained with the following compressed measurement dimensions: $4 \times 6 \times 2$, $6 \times 4 \times 2$, $6 \times 8 \times 1$, $8 \times 6 \times 1$; $4 \times 7 \times 2$, $7 \times 4 \times 2$, $7 \times 8 \times 1$, $8 \times 7 \times 1$; $6 \times 6 \times 2$, $8 \times 9 \times 1$, $9 \times 8 \times 1$, $12 \times 6 \times 1$, $9 \times 4 \times 2$; $10 \times 10 \times 1$, $10 \times 5 \times 2$. For the proposed `adaptive-rate` training method, only one model instance was trained for each dataset with the maximum and minimum dimensions of the compressed measurements set to $15 \times 15 \times 2$ and $4 \times 4 \times 1$, respectively. After training, this single model instance was evaluated with different compressed measurement sizes that were used to train the `single-rate` models.

The results for the CelebA dataset are shown in Tables 13. In addition to the test accuracy for each compressed measurement size, the average accuracy of each method, and the total number of epochs used to train the model(s) are also provided. It is obvious that the proposed (`adaptive-rate`) models clearly outperform the `baseline` models, indicating that the original MCL training algorithm, without any modification, cannot be used to train a single model instance that can work for multiple compression rates. Large test accuracy variances imply that there is no semantic structure within the sub-tensors of the compressed measurement. On the other hand, a single model instance trained with the proposed `adaptive-rate` method produced more consistent performances between different runs.

Comparing `adaptive-rate` models and `single-rate` models, we can see that a single model trained using the proposed `adaptive-rate` approach achieved very competitive performances across different compressed measurement dimensions. Even though all-in-one models trained with the proposed `adaptive-rate` method are slightly inferior compared to dedicated `single-rate` models, these small performance losses are compensated with significant computational and memory gains. Using the `single-rate` approach requires a total of 3150 epochs

Table 13: Test Accuracy (%) on CelebA

	`single-rate`	`one-shot`	
		`baseline`	`adaptive-rate`
$4 \times 6 \times 2$	49.12±00.17	06.42±00.17	45.60±00.31
$6 \times 4 \times 2$	53.38±00.13	08.02±00.52	51.37±00.17
$6 \times 8 \times 1$	61.00±00.17	09.65±02.66	57.60±00.48
$8 \times 6 \times 1$	63.38±00.88	16.60±00.52	62.12±00.54
$4 \times 7 \times 2$	51.79±00.24	07.06±00.12	49.33±00.74
$7 \times 4 \times 2$	60.17±00.12	11.42±00.13	56.56±00.35
$7 \times 8 \times 1$	68.08±00.37	16.31±01.31	64.21±00.25
$8 \times 7 \times 1$	68.75±00.82	17.58±00.12	64.77±00.56
$6 \times 6 \times 2$	60.40±01.54	13.17±01.37	58.00±00.23
$8 \times 9 \times 1$	73.06±00.19	23.50±00.58	69.44±00.17
$9 \times 8 \times 1$	74.40±00.52	28.35±00.77	71.31±00.39
$12 \times 6 \times 1$	74.48±00.23	30.92±01.46	72.23±00.29
$9 \times 4 \times 2$	66.40±00.19	17.02±00.15	64.44±00.35
$10 \times 10 \times 1$	80.00±00.21	44.71±01.88	76.75±00.29
$10 \times 5 \times 2$	72.92±00.25	22.73±02.02	70.00±00.41
average	65.16	18.23	62.25
$\sum$epochs	3150	210	210

of gradient updates for each dataset when training 15 different models for 15 different compressed measurement sizes. On the other hand, with `adaptive-rate` training, only a single model instance was trained for all 15 different configurations of the compressed measurement, requiring only 210 epochs.

From Table 13, we can see that with less than 3% of accuracy drop, the `adaptive-rate` formulation allows us to efficiently train and deploy a single model that can be used for inference with an adaptive compression rate. In addition to the single model setup, we also evaluated the scenario when the computing server has sufficient computational power to run dedicated instances of the FS component and the task-specific classifier. That is, after optimizing a single model instance using the `adaptive-rate` method, parameters of the MCS component were frozen while parameters of the FS component and the task-specific classifier were finetuned for 30 epochs for each target compressed measurement size. The results, which are denoted as `adaptive-rate*`, are shown in Table 14. It is obvious that the `adaptive-rate*` setup leads to noticeable performance improvements, which are on-par or better compared to the `single-rate` training approach. Thus, higher performance can be achieved under different conditions in the case of varying transmission channel conditions, compared to the case of having a MCL model suitable for single-rate based classification. We should note here that for achieving similar performance in such cases one could opt for deploying multiple single-rate MCL models, however this would lead to more inefficient inference compared to using the proposed approach. Although the training complexity (total number of epochs) using the `adaptive-rate*` approach is higher than using a single model instance (`adaptive-rate`), it is still far below the training complexity of the `single-rate` training method.

Table 14: Accuracy after Finetuning under `adaptive-rate*` on CelebA

	`single -rate`	`adaptive -rate`	`adaptive -rate*`
$4 \times 6 \times 2$	49.12±00.17	45.60±00.31	49.92±00.42
$6 \times 4 \times 2$	53.38±00.13	51.37±00.17	53.85±00.15
$6 \times 8 \times 1$	61.00±00.17	57.60±00.48	61.15±00.19
$8 \times 6 \times 1$	63.38±00.88	62.12±00.54	65.04±01.19
$4 \times 7 \times 2$	51.79±00.24	49.33±00.74	52.67±00.49
$7 \times 4 \times 2$	60.17±00.12	56.56±00.35	61.50±00.47
$7 \times 8 \times 1$	68.08±00.37	64.21±00.25	68.17±00.68
$8 \times 7 \times 1$	68.75±00.82	64.77±00.56	68.63±00.33
$6 \times 6 \times 2$	60.40±01.54	58.00±00.23	62.73±01.31
$8 \times 9 \times 1$	73.06±00.19	69.44±00.17	73.04±00.37
$9 \times 8 \times 1$	74.40±00.52	71.31±00.39	74.15±00.19
$12 \times 6 \times 1$	74.48±00.23	72.23±00.29	74.48±00.15
$9 \times 4 \times 2$	66.40±00.19	64.44±00.35	67.67±01.04
$10 \times 10 \times 1$	80.00±00.21	76.75±00.29	79.44±00.96
$10 \times 5 \times 2$	72.92±00.25	70.00±00.41	71.12±00.38
average	65.16	62.25	65.57
$\sum$epochs	3150	210	660

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

3.1 Continual 3D Convolutional Neural Networks for Real-time Processing of Videos

3.1.1 Introduction and objectives

Video Recognition is a challenging and computationally costly task, which has been tackled by many different computational approaches and neural network architectures. While multiple works achieve good accuracy on Human Action Recognition benchmarks, such as the challenging Kinetics-400 [75], their computational cost per prediction is prohibitively high for deployment in both online scenarios and on embedded devices, and thereby for robotics applications as such.

In the work presented here, we aim to tackle this issue with the introduction of Continual 3D Convolutional Neural Networks (Co3D CNNs). Co3D CNNs are able to reuse the weights from existing efficient 3D CNNs, which operate on spatio-temporal input clips, and perform the same computations frame by frame, thereby increasing the prediction frequency at marginal cost. Co3D CNNs improve substantially on the state-of-the-art in efficient activity recognition and can deliver 67.3% / 71.0% accuracy at 40.1 / 23.7 predictions per second on the embedded device NVIDIA Xavier for continual input video streams. The accuracy/complexity trade-off for the Continual X3D and recent state-of-the-art methods on Kinetics-400 dataset is shown in Fig. 10. It is our hope that the improvements introduced here will make higher levels of spatio-temporal reasoning viable for future robotics applications.

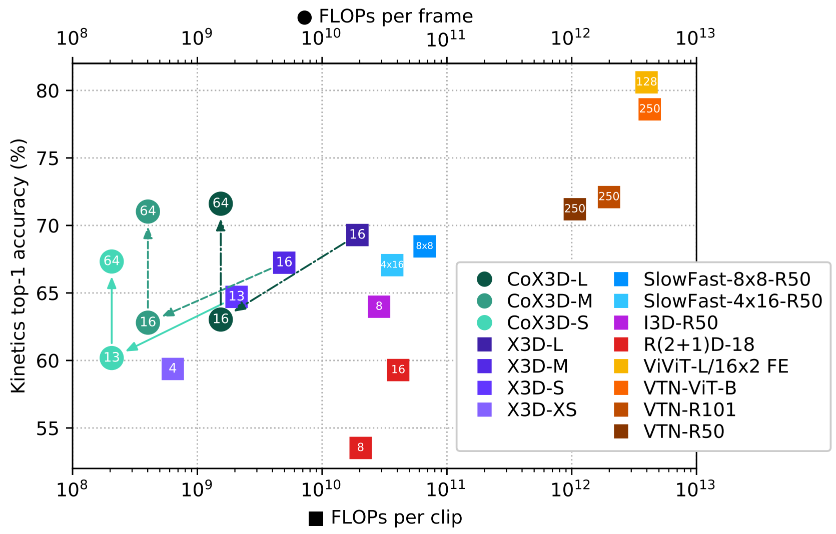


Figure 10: **Accuracy/complexity trade-off** for Continual X3D (*Co*X3D) and recent state-of-the-art methods on Kinetics-400 using 1-clip/frame testing. The ■ FLOPs per *clip* are noted for regular networks, while the ● FLOPs per *frame* are shown for the Continual 3D CNNs. Frames per clip / global average pool size for each model is noted in the representative points. Diagonal and vertical arrows indicate respectively a direct weight transfer from regular to Continual 3D CNN and an extension of receptive field.

3.1.2 Summary of state of the art

Video Recognition has been tackled with multiple architectural approaches such as 2D CNNs + RNNs [27, 67], 3D CNNs [15, 150, 35, 34], and Transformer-based methods [3, 107]. While some methods achieve high accuracy on common Human Activity Recognition benchmarks, their high accuracy is usually accompanied by equally high computational complexity. Recent research [85, 34, 180] has begun to tackle this computational burden by taking inspiration in building blocks from lightweight image recognition research [56, 173, 147]. The Extended 3D (X3D) [34] is one such architecture family, which has produced state-of-the-art results in the trade-off between accuracy and floating point operations (FLOPs). In essence, it can be considered as a 3D EfficientNet [147] where the input resolutions, channels sizes and depth were found by progressive search. Yet, X3D and all other 3D CNNs are limited in their online inference capabilities, as they operate on clips instead of frames. When frame-wise predictions are required, this could only be achieved with overlapping clips as seen in Fig. 11.

Some prior methods have proposed modifications to the spatio-temporal 3D convolution to combat this issue. The Recurrent Convolutional Unit (RCU) [139] is one such method, which replaces the 3D convolution with an aggregation of the spatial 2D convolution over the current input with a 1D convolution over the prior output. Dissected 3D CNNs [81] (D3D) define an architecture that caches the $1 \times n_H \times n_W$ frame-level features in the network residual connections and concatenates them with the corresponding features in the next frame. This produces intermediary spatio-temporal features of shape $2 \times n_H \times n_W$, which are used as input to a block of convolutional layers. With kernel sizes $k_T \times k_H \times k_W$ of $2 \times 3 \times 3$ and $1 \times 3 \times 3$, the block produces features of shape $1 \times n_H \times n_W$ to be cached once again. Here, n denotes the feature map size and k the kernel size and subscripts T, H, and W denote the time, height, and width dimensions. Both RCU and D3D are causal and operate frame-by-frame. Unlike our

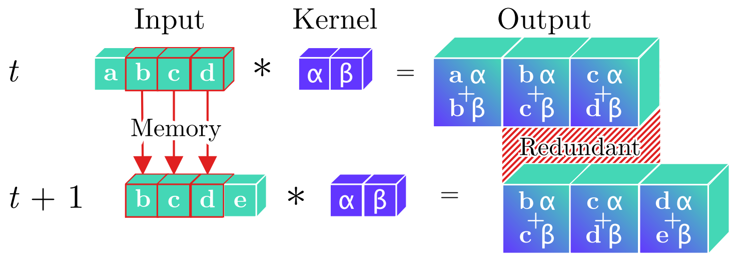


Figure 11: **Redundant computations** for a temporal convolutional layer during online processing of video clips, as illustrated by the repeated convolution of inputs (green $\mathbf{b},\mathbf{c},\mathbf{d}$) with a kernel (blue α,β) in the temporal dimension. Moreover, prior inputs ($\mathbf{b},\mathbf{c},\mathbf{d}$) must be stored between time-steps for online processing tasks.

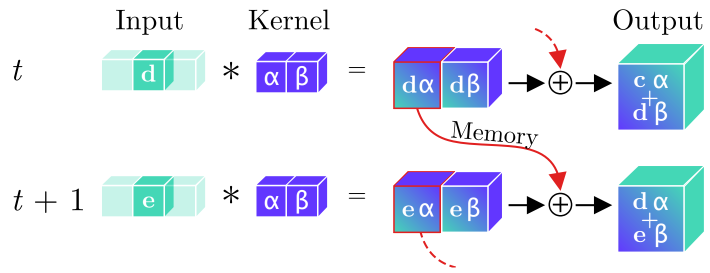


Figure 12: **Continual Convolution**. An input (green $\mathbf{d}$ or $\mathbf{e}$) is convolved with a kernel (blue α,β). The intermediary feature-maps corresponding to all but the last temporal position are stored, while the last feature map and prior memory are summed to produce the resulting output. For a continual stream of inputs, Continual Convolutions produce identical outputs to regular convolutions.

proposed Continual Convolutions, however, they are incompatible with pre-trained 3D CNNs, and must be trained from scratch.

3.1.3 Description of work performed so far

Continual 3D Convolutional Neural Networks mirror regular 3D CNNs, but replace convolutions, pooling operators and residual connections with corresponding weight-compatible continual versions.

As illustrated for the 1D-case in Fig. 12, Continual Convolutions (CoConvs) convolve each frame with the kernel separately. The intermediary results are then cached and summed up at the appropriate time-steps to produce the resulting output corresponding to a multi-frame convolution. Consider the scenario where a unique prediction is required for each frame. Compared with regular convolutions, CoConvs reduce the computational burden in proportion to the size of the input clip, that the prediction should cover.

In the context of a CNN, residuals are often added to representation following the downstream network layer. In the context of CoConv layers, however, the residual feature-map cannot be added naively in this manner, as this would results in a temporal misalignment. Considering a CoConv layer with temporal dilation d_T, kernel size k_T and padding p_T, the residual must delayed by $d_T \cdot (k_T - p_T - 1)$ to be aligned.

Since the per-prediction computational complexity of Co3D CNNs is virtually independent of the corresponding clip size, we can increase their receptive field at negligible cost to increase predictive performance. This can be easily done by increasing the temporal receptive field of the final pooling layer, which is commonly found in 3D CNNs. The principles, properties and limitations of Continual 3D CNNs are described in more detail in the associated paper, which can be found in Appendix 8.4:

- [49] L. Hedegaard, and A. Iosifidis, "*Continual 3D Convolutional Neural Networks for Real-time Processing of Videos*", arXiv preprint, arXiv:2106.00050, 2021.

3.1.4 Performance evaluation

To serve as our baseline, we performed a comprehensive benchmark of per-prediction accuracy on Kinetics-400 for prior 3D CNNs [15, 150, 35, 34], as well as their FLOPs and throughput on various computational devices. This is presented in Table 15. To test our method, we implemented continual versions of the X3D S, M, and L networks (*Co*X3D), which underwent the same benchmark. Compared to the original X3D models, we see remarkable reductions in per prediction FLOPs and throughput as summarized in Table 16. To further validate the method, we evaluate the Slow [35] and *Co*Slow networks on Charades [138]. As seen in Table 17, we again observe computational benefits in proportion to the clip-size of the original network.

	Model	**Acc. (%)**	**Params (M)**	**FLOPs (G)**	**Throughput (evaluations/s)**			
					CPU	TX2	Xavier	RTX 2080 Ti
Clip	I3D-R50	63.98	28.04	28.61	0.93 ± 0.04	2.54 ± 0.02	5.37 ± 0.01	77.15 ± 0.88
	R(2+1)D-18$_8$	53.52	31.51	20.35	1.75 ± 0.11	3.19 ± 0.04	6.82 ± 0.01	130.88 ± 0.29
	R(2+1)D-18$_{16}$	59.29	31.51	40.71	0.83 ± 0.06	1.82 ± 0.01	3.77 ± 0.01	75.81 ± 0.21
	SlowFast-8×8-R50	68.45	66.25	50.72	0.34 ± 0.01	0.87 ± 0.00	1.66 ± 0.03	30.72 ± 0.34
	SlowFast-4×16-R50	67.06	34.48	36.46	0.55 ± 0.02	1.33 ± 0.01	2.13 ± 0.05	41.28 ± 0.51
	X3D-L	69.29	6.15	19.17	0.25 ± 0.01	0.19 ± 0.00	0.88 ± 0.00	16.58 ± 0.13
	X3D-M	67.24	3.79	4.97	0.83 ± 0.04	1.47 ± 0.00	3.69 ± 0.02	55.27 ± 0.67
	X3D-S	64.71	3.79	2.06	2.23 ± 0.11	2.68 ± 0.01	8.07 ± 0.12	138.04 ± 1.69
	X3D-XS	59.37	3.79	0.64	8.26 ± 0.11	8.20 ± 0.09	26.37 ± 0.03	430.15 ± 9.29
Frame	*Co*X3D-L$_{16}$	63.03	6.15	1.54	2.30 ± 0.07	0.99 ± 0.00	6.30 ± 0.00	101.38 ± 3.36
	***Co*X3D-L$_{64}$**	**71.61**	**6.15**	**1.54**	$\mathbf{2.30 \pm 0.08}$	$\mathbf{0.99 \pm 0.00}$	$\mathbf{6.30 \pm 0.01}$	$\mathbf{111.53 \pm 4.55}$
	*Co*X3D-M$_{16}$	62.80	3.79	0.40	7.57 ± 0.14	7.26 ± 0.13	23.70 ± 0.09	335.42 ± 9.91
	***Co*X3D-M$_{64}$**	**71.03**	**3.79**	**0.40**	$\mathbf{7.51 \pm 0.17}$	$\mathbf{7.04 \pm 0.03}$	$\mathbf{23.70 \pm 0.09}$	$\mathbf{323.56 \pm 9.91}$
	*Co*X3D-S$_{13}$	60.18	3.79	0.21	13.16 ± 0.35	11.06 ± 0.03	40.09 ± 0.04	722.43 ± 56.95
	***Co*X3D-S$_{64}$**	**67.33**	**3.79**	**0.21**	$\mathbf{13.19 \pm 0.37}$	$\mathbf{11.13 \pm 0.04}$	$\mathbf{40.10 \pm 0.04}$	$\mathbf{726.81 \pm 67.38}$

Table 15: **Kinetics-400 benchmark**. The noted accuracy is the single clip or frame top-1 score using RGB as the only input-modality. The performance was evaluated using publicly available pre-trained models without any further fine-tuning. For speed comparison, evaluations per second denote frames per second for the *Co*X3D models and clips per second for the remaining models. Speed results are the mean ± std of 100 measurements. Pareto-optimal models are marked with bold.

3.1.5 Future work

Our research into continual formulations of common building blocks in deep neural networks is ongoing. Specifically, we are exploring various approaches for weight transfer and fine-tuning in the context of Graph Convolutional Neural Networks for Skeleton-based Action Recognition. Our current results are presented in Section 3.2. Another exciting direction of future research is

Model	**FLOPs**	**Throughput (evaluations/s)**			
		CPU	TX2	Xavier	RTX
*Co*X3D-L_{64}	12.44×	9.20×	5.21×	7.16×	6.73×
*Co*X3D-M_{64}	12.37×	9.05×	4.79×	6.42×	5.85×
*Co*X3D-S_{64}	10.01×	5.91×	4.15×	4.97×	5.27×

Table 16: **Relative improvements** in frame-by-frame prediction in *Co*X3D-$\{S,M,L\}_{64}$ networks compared to X3D. The improvements (× lower for FLOPs and × higher for throughput) correspond to the results in Table 15.

	Model	**FLOPs (G) × views**	**mAP (%)**
Clip	$\text{Slow}_{8\times8}$ [35]*	54.9×1	21.4
Clip	$\text{Slow}_{8\times8}$ (ours)	54.9×1	24.1
Fr.	*Co*Slow_{8}	6.9×1	21.5
Fr.	*Co*Slow_{64}	6.9×1	25.2

Table 17: **Charades benchmark.** Noted are the FLOPs × views and video-level mean average precision (mAP) on the validation set using pre-trained model weights. The result denoted with '*' was achieved using the publicly available SlowFast code [35].

the reformulation of the multi-head self-attention and Transformers for temporal sequences as continual.

3.2 Continual Skeletons for Efficient Online Activity Recognition

3.2.1 Introduction and objectives

Skeleton-based human activity recognition has attracted research interest in recent years and many deep-learning methods have been proposed in this area. In these methods, a human activity video is modeled as a spatio-temporal graph representing a sequence of human body poses in a time period, and each each body pose is modeled as a skeleton formed by an arrangement of 2D or 3D coordinates of human body joints and the natural physical connections between them. Accordingly, the human activity recognition is formulated as a spatio-temporal graph classification problem. Graph Convolutional Networks (GCNs) have been successful when applied to many classification tasks by generalizing the convolution operation from grid data into graph data structures to take advantage of the non-Euclidean structure of graphs.

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. However the high computational complexity of these methods make their applications infeasible for real-time scenarios with restricted computational capacity. Minimizing the number of floating point operations (FLOPs) can make a large step towards addressing the computational and memory efficiency in these methods. In this regard, we propose to capture the temporal features in a skeleton sequence by employing continual convolution operation, instead of standard 2D convolution, to perform the computation in a streaming manner while storing the intermediate results. In the following, we provide a brief description of the state-of-the-art, the proposed method, and the experimental results obtained so far.

3.2.2 Summary of state of the art

The ST-GCN method [168] is the first GCN-based method proposed for skeleton-based human activity recognition. It is composed of several feature extraction layers applying graph convolutions in both spatial and temporal domains to aggregate action-related information in both dimensions, and one classification layer to predict the activity labels. Recently, several methods have built on top of ST-GCN to improve its performance by enhancing the feature extraction or optimizing the model's structure. 2s-AGCN [137] is one of the state-of-the-art methods proposed on top of ST-GCN which adaptively updates the graph structure of the skeleton in each layer of the model by learning a spatial attention mask and a data-dependant graph in an end-to-end manner. Besides, it trains the model with two different data streams to utilize both joint and bone features. GCN-NAS [123] is a neural architecture search method which has improved the classification performance by exploring a search space to determine the best graph structure at each layer of the ST-GCN. More recently, Shift-GCN [18] and SGN [172] methods have been proposed to improve both the classification accuracy and computational efficiency. Shift-GCN employs lightweight shift operation as an alternative of 2D convolution to reduce the number of Floating Point Operations (FLOPs), and SGN utilises semantic information like joint type and frame index as side information to design a compact semantics-guided neural network (SGN) for capturing both spatial and temporal correlations in joint and frame level.

During the first year of the OpenDR project, we contributed in this area by proposing three new methods, TA-GCN [50], PST-GCN [53] and ST-BLN [51], for skeleton-based human activity recognition. TA-GCN tries to make the inference process of ST-GCN more efficient by selecting a subset of key skeletons, which hold the most important features for activity recognition, PST-GCN tries to find an optimized data-dependent ST-GCN architecture to increase the efficiency of the classification task, and in ST-BLN we proposed to formulate the spatial graph convolution as a bilinear mapping which provides more design flexibility for the user. The full description of our previously proposed methods was provided in deliverable D3.1, and the model implementations are integrated in OpenDR toolkit. In our new work, our focus is on improving the computational efficiency of the skeleton-based human activity recognition methods in the online inference scenario, while preserving the state-of-the-art performance.

3.2.3 Description of work performed so far

An undirected spatio-temporal graph on a sequence of skeletons is denoted as $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, where the set of nodes $\mathscr{V}$ indicates 2D or 3D coordinates of N body joints of a skeleton in a sequence of T time steps and $\mathscr{E}$ 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. Fig. 13 (right) shows the spatio-temporal graph on a sequence of skeletons. Considering the spatial partitioning process [168], each node has 3 subsets of neighbors, which is illustrated in Fig. 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. Accordingly, the adjacency matrix $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N = \sum_p \mathbf{A}_p$ is defined as the summation of 3 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 $\hat{\mathbf{A}}_p = \mathbf{D}_p^{-\frac{1}{2}} \mathbf{A}_p \mathbf{D}_p^{-\frac{1}{2}}$, where $\mathbf{D}_p$

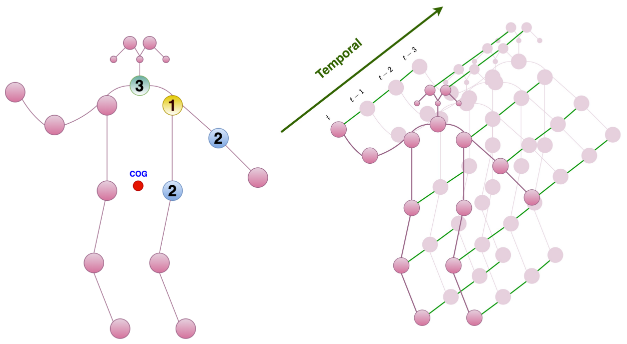


Figure 13: Illustration of an example human body poses encoded as a Spatio-temporal graph (right), and the neighboring subsets in spatial partitioning process in different colors (left).

denotes the degree matrix.

In the baseline method ST-GCN and its extensions like 2s-AGCN, the model receives the feature tensor $\mathbf{X} \in \mathbb{R}^{C^{in} \times T \times V}$ as input, where C^{in} denotes the number of input channels, T is the number of skeletons and V is the number of body joints in each skeleton. The feature vectors of the nodes are updated by applying successive layers of spatio-temporal graph convolutions. The spatial graph convolution in layer l is defined as follows:

$$\mathbf{H}_s^{(l)} = ReLU\left(\sum_p (\hat{\mathbf{A}}_p + \mathbf{M}_p)\mathbf{H}^{(l-1)}\mathbf{W}_p^{(l)}\right), \tag{13}$$

where $\mathbf{M}_p \in \mathbb{R}^{N \times N}$ is a learnable attention map which highlights the elements of each adjacency matrix, and $\mathbf{W}_p^{(l)}$ denotes the weight matrix which transforms the node features of each partition. Practically, the feature transformation is performed by a standard 2D convolution operation which applies $C^{(l)}$ filters of size $C^{(l-1)} \times 1 \times 1$ on the layer's input $\mathbf{H}_s^{(l)}$.

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, i.e. $\mathbf{H}_s^{(l)} \in \mathbb{R}^{C^{(l)} \times T \times V}$. In the baseline method ST-GCN and most of the state-of-the-art methods such as 2s-AGCN, DGNN, and GCN-NAS, the temporal convolution is basically a standard 2D convolutions applying $C^{(l)}$ filters of size $C^{(l)} \times k \times 1$ to aggregate the features through k consecutive skeletons in a sequence.

The primary contribution in this work is the application and exploration of Continual 2D Convolutions as a replacement for the regular 2D convolutions. Continual Convolutions were first proposed in [49] and are briefly described in Section 3.1. Since the ST-GCN method and its extensions employ temporal convolutions over long sequences ($\approx$ 300 frames) for many of the standard benchmarks, they can greatly benefit from the continual formulation, as this reduces the per-prediction complexity in proportion to the employed sequence length. *Co*ST-GCNs thus lead to new opportunities and challenges, that may contribute to uncover properties and best practices for the transfer of "regular" spatio-temporal neural networks to continual ones.

3.2.4 Performance evaluation

Thus far, our experiments comprise the evaluation of multiple transfer approaches using the (one-stream) ST-GCN and AGCN architectures on the human activity recognition dataset NTU-RGBD-60 [134]. The ST-GCN-based architecture include convolutional layers with stride > 1.

Model	**Frames per pred**	**Accuracy (%)** X-Sub	X-View	**Params (M)**	**FLOPs (G)**	**Throughput (preds/s)** CPU
ST-GCN	300	86.0	93.4	3.14	16.73	2.3
ST-GCN*	300	87.2 ($+1.2$)	93.6 ($+0.2$)	3.14	36.90 ($+2.2\times$)	1.1 (-2.1)
*Co*ST-GCN	4	80.6 (-5.4)	89.9 (-3.5)	3.14	0.27 ($-63.2\times$)	24.1 ($-10.5\times$)
*Co*ST-GCN*	1	86.5 ($+0.5$)	93.2 (-0.2)	3.14	0.16 ($-107.7\times$)	45.8 ($-19.9\times$)
AGCN	300	86.4	94.3	3.47	18.69	2.1
AGCN*	300	86.6 ($+0.2$)	94.3 ($=$)	3.47	40.87 ($+2.2\times$)	1.0 (-2.1)
*Co*AGCN	4	80.3 (-6.1)	85.3 (-9.0)	3.47	0.30 ($-63.2\times$)	14.6 ($-7.0\times$)
*Co*AGCN*	1	-	-	3.47	0.17 ($-108.8\times$)	34.6 ($-16.5\times$)

Table 18: **NTU-RGBD-60 Transfer performance benchmarks**. Noted is the top-1 validation accuracy using joints as the only input-modality. The FLOPs and throughput on CPU (2.6 GHz 6-Core Intel Core i7) accounts for one new prediction. The colored figures in parentheses denote the relative changes for the specific model relative to is source model.

In the continual inference scenario, this results in a reduction of the prediction rate. For instance, two layers with stride 2 each would reduce predictions rate to a quarter. Therefore, we propose to modify and finetune the ST-GCN architectures by setting their temporal strides to 1 in all layers. The models marked with '*' in Table 18 enforce temporal stride 1, while the other models follow the original architecture. Table 18 shows the evaluation of accuracy, parameter counts, floating points operations (FLOPs) per prediction and the real-life throughput on a one core of a 2.6 GHz Intel i7 CPU. These preliminary results are very encouraging, and showcase a perfect use-case of continual convolution for online skeleton-based human action recognition.

3.2.5 Future work

The work on Continual Skeletons is still in progress, and it is our plan to extend the presented results with additional models and datasets, as well as further on-hardware benchmarks.

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. It has been shown that the FER performance can be improved by utilizing the localized facial landmark coordinates for face alignment [105]. 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

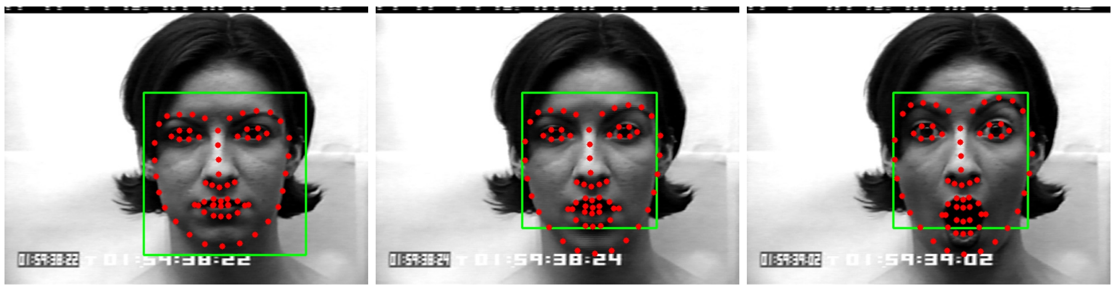


Figure 14: Illustration of facial images in 3 different time steps and their corresponding extracted landmark points which are denoted as graph nodes in a spatio-temporal graph. The images are from CK+ dataset [95].

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.

Similar to human body skeletons used for human activity recognition, 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. Therefore, the Spatio-Temporal Graph Convolutional Networks (ST-GCNs) [168, 137], the Progressive ST-GCN (PST-GCN) [53] which tries to find an optimized ST-GCN architecture, or the recently introduced Spatio-Temporal Bilinear Network (ST-BLN) [51] can be employed to extract informative features from a sequence of graphs, encoding facial landmarks through different time steps, for facial expression recognition.

Accordingly, we proposed the Progressive Spatio-Temporal Bilinear Network (PST-BLN) method for landmark-based facial expression recognition, by inheriting the advantage of ST-BLN [51] to learn graph structures at each layer of the network. The PST-BLN method automatically defines an optimized, compact and data-dependant network topology for ST-BLN and also captures the model's uncertainty by employing Monte Carlo Dropout [154] technique in training and inference process which gives the users of FER system the possibility to treat uncertain cases explicitly. A summary of this work is provided hereafter. The corresponding publication is listed below, and can be found in Appendix 8.5:

1. [52] N. Heidari, and A. Iosifidis, "*Progressive Spatio-Temporal Bilinear Network with Monte Carlo Dropout for Landmark-based Facial Expression Recognition with Uncertainty Estimation*", IEEE International Workshop on Multimedia Signal Processing (MMSP) 2021

4.1.2 Summary of state of the art

Recently, many real-time facial landmark detection methods have been developed which achieve good performance in addition to their high efficiency [28]. In some studies [70, 80, 47], multi-modal data fusion based on facial landmarks and images or videos are proposed to improve the FER method's performance. Fig. 14 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 intermediate point in time (middle)) and the aligned landmark points for each image, which are extracted using the method in [76].

To the best of our knowledge, there is only one GCN-based method for landmark-based FER which has been proposed in [108] and it adopts landmark extractor [28] to extract accurate 2D coordinates of 68 landmark points from each facial image in an image sequence. The extracted landmarks are modeled by a directed spatio-temporal graph which is formed using landmark points as nodes and triangle meshes among all landmarks, built by Delaunay method, as edges. Inspired by methods recently proposed for skeleton-based human action recognition, like the DGNN [136], the FER method [108] 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.

4.1.3 Description of work performed so far

In this work, we adopted the Dlib's facial landmark extractor [76] to extract accurate 2D coordinates of 68 landmark points from each facial image. A spatio-temporal graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ can be constructed where $\mathscr{V}$ is the node set of 2D coordinates of the facial landmarks and $\mathscr{E}$ is the set of graph edges encoding spatial and temporal connections between the landmarks through different time steps. The triangle meshes among all landmarks obtained by Delaunay method make the spatial graph edges and the temporal graph edges connect each landmark into its corresponding landmark in its previous and subsequent frames. We utilize the edge features of the graph which encode the motion of the facial muscles instead of the landmark coordinates.

Therefore, the proposed PST-BLN model receives as input a tensor $\mathbf{X} \in \mathbb{R}^{F \times T \times E}$ encoding a sequence of T spatial graphs expressing the connections of the graph edges each with F feature dimension, and finds an optimized compact architecture for ST-BLN model based on the received data. A ST-BLN model is composed of several Spatio-Temporal Bilinear Layers (ST-BLLs) for feature extraction and one fully connected layer for classification. Each ST-BLL is composed of a bilinear transformation and a temporal convolution. The bilinear transformation receives as input $\mathbf{H}^{(l-1)}$ which is the representations for the $E^{(l-1)}$ facial graph edges at layer $l-1$, and transforms them by using a learnable weight matrix $\mathbf{W}^{(l)}$ as follows:

$$\mathbf{H}_s^{(l)} = ReLU\left(\mathbf{U}^{(l)}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)}\right), \tag{14}$$

where $\mathbf{U}^{(l)} \in \mathbb{R}^{E^{(l)} \times E^{(l-1)}}$ is a randomly initialized learnable matrix indicating the spatial weighted connections between the facial graph edges. The spatially transformed feature tensor $\mathbf{H}_s^{(l)} \in \mathbb{R}^{F^{(l)} \times T^{(l)} \times E}$ is introduced to the temporal convolution, which captures the motion of the facial muscles by propagating the features through the time domain using a standard 2D convolution with a predefined kernel size $K \times 1$. The structure of the ST-BLL is shown in Fig. 15. The output of the l^{th} ST-BLL, is passed to a global average pooling layer to produce a feature vector of size $F^{(l)} \times 1$ which is then classified by a fully connected layer which maps features from $F^{(l)}$ to C dimensional subspace.

The PST-BLN method starts with an empty network and builds the ST-BLLs one by one and adds them to the model until reaching convergence. Let us assume that a ST-BLN with $l-1$ layers has been already built, and the method proceeds in building the l^{th} layer. When the method starts building the l^{th} layer, the number of output channels in all the 2D convolutions is set to a predefined fixed number $F^{(l)} = b$ and at each iteration, it is increased by $F^{(l)} = F^{(l)} + b$. While all the model's parameters in the previously built layers are initialized by the finetuned weights, the newly added neurons to the network are initialized randomly and all the model parameters are fine-tuned in an end-to-end manner using back-propagation. The layer's

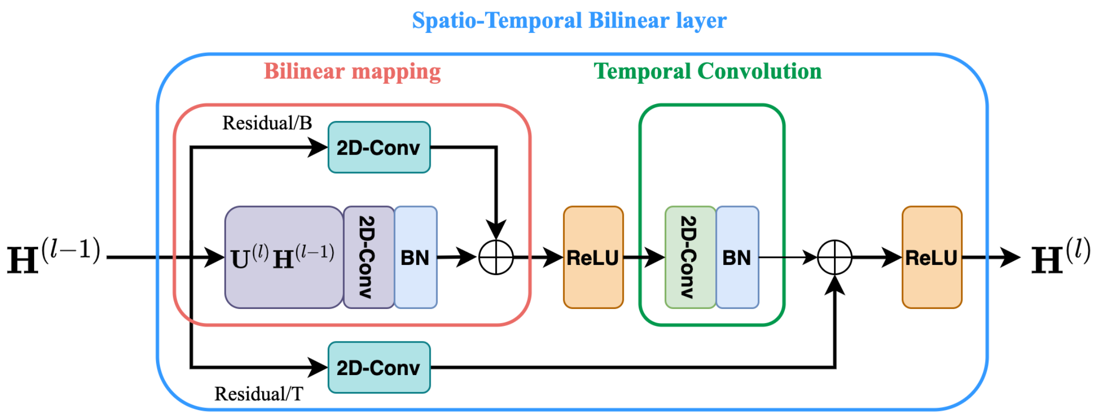


Figure 15: Illustration of spatio-temporal bilinear layer l. It receives $\mathbf{H}^{(l-1)}$ of size $F^{(l-1)} \times T^{(l-1)} \times E$ as input and applies bilinear projection and temporal convolution to produce the output representation $\mathbf{H}^{(l)}$ of size $F^{(l)} \times T^{(l)} \times E$. The bilinear mapping block and the temporal convolution block are both followed by batch-normalization (BN) and ReLU activation function.

width progression at iteration t is evaluated according to the model's performance in terms of categorical loss value on training data, i.e. $\alpha_w = (\mathscr{L}_{t-1}^{(l)} - \mathscr{L}_t^{(l)})/\mathscr{L}_{t-1}^{(l)}$. If $\alpha_w < \varepsilon_w$ with $\varepsilon_w > 0$, it shows that increasing the layer's width does not improve the model's performance anymore and the method stops progression in that layer. Otherwise, the newly added parameters are saved and the next iteration starts increasing the layer's width by adding b more output channels to the filters of all the 2D convolutions in that layer. After building each layer of the network, the method evaluates the model's depth progression using the rate of improvement in model's performance, i.e. $\alpha_d = (\mathscr{L}^{(l-1)} - \mathscr{L}^{(l)})/\mathscr{L}^{(l-1)}$, in terms of categorical loss value on training data. When $\alpha_d < \varepsilon_d$ with $\varepsilon_d > 0$, the method stops depth progression and the newly added layer is removed. Finally, all the model's parameters are fine-tuned together and the method returns the optimized topology for the ST-BLN model and its performance on training and validation data.

Since facial expression datasets are small in size, regularization of the network parameters is needed to prevent overfitting. To address this, we add a dropout layer after each ST-BLL built by the proposed method using a dropout rate p of 0.2. This choice also allows us to use Monte Carlo Dropout [40] to capture the uncertainty of the model during inference. The main idea of Monte Carlo Dropout is to activate dropout not only in the training phase, but also during the inference. By repeating the inference for an input facial spatio-temporal graph with an activated dropout, the outputs of the PST-BLN are combined as an ensemble of different PST-BLN models and the variance in the outputs are used to capture the classification uncertainty.

4.1.4 Performance evaluation

Our method's performance is evaluated on three widely used datasets, Extended Cohn–Kanade (CK+) [95, 73], Oulu-CASIA [174], and Acted Facial Expressions in the Wild (AFEW) [26, 25]. The experiments are conducted with a single GeForce RTX 2080 GPU. The performance of the proposed method is compared with both video-based and landmark-based state-of-the-art methods on AFEW, Oulu-CASIA and CK+ datasets in Tables 19, 20, and 21, respectively. The ST-GCN [168] and AGCN [137] methods are also included in the comparisons to evaluate their

Table 19: Comparison of video/image-based and landmark-based methods on the validation set of AFEW dataset

Method	Acc(%)	#Params	Data type
SSE-HoloNet [60]	46.48	-	Video
VGG-LSTM [157]	48.60	-	Video
C3D-LSTM [157]	43.20	-	Video
C3D-GRU [88]	49.87	-	Video
ST-GCN [168]	28.17	131.3k	Landmark
AGCN [137]	24.21	143.7k	Landmark
DGNN [136]	32.64	538k	Landmark
ST-BLN w/MCD	**36.11**	**132.3k**	Landmark
ST-BLN wo/MCD	34.13	132.3k	Landmark
PST-BLN w/MCD	**33.33**	**10.8k**	Landmark
PST-BLN wo/MCD	30.15	10.8k	Landmark

Table 20: Comparison of video-based and landmark-based methods on Oulu-CASIA dataset using 10-fold cross validation

Method	Acc(%)	#Params	Data type
DTAN [70]	74.38	-	Video
DTGN [70]	74.17	177.6k	Landmark
DTAGN [70]	81.46	-	Video + Landmark
PPDN [176]	84.59	6.8m	Video
PHRNN-MSCNN [171]	86.25	1.6m	Video + Landmark
DCPN [169]	86.23	-	Video
CDLM [83]	91.67	2.7m	Video
FDRL [128]	88.26	-	Video
ST-GCN [168]	77.08	131.3k	Landmark
AGCN [137]	75.62	143.7k	Landmark
DGNN [136]	81.46	535,7k	Landmark
ST-BLN w/MCD	**83.54**	**132.3k**	Landmark
ST-BLN wo/MCD	82.08	132.3k	Landmark
PST-BLN w/MCD	**79.79**	**7.59k**	Landmark
PST-BLN wo/MCD	78.74	7.59k	Landmark

Table 21: Comparison of the video-based and landmark-based methods on CK+ dataset using 10-fold cross validation

Method	Acc(%)	#Params	Data type
DTAN [70]	91.44	-	Video
DTGN [70]	92.35	177.6k	Landmark
DTAGN [70]	97.25	-	Video + Landmark
PPDN [176]	99.3	6.8m	Video
PHRNN-MSCNN [171]	98.5	1.6m	Video + Landmark
DCPN [169]	99.6	-	Video
CDLM [83]	98.47	2.7m	Video
FDRL [128]	99.54	-	Video
ST-GCN [168]	93.64	131.3k	Landmark
AGCN [137]	94.18	143.7k	Landmark
DGNN [136]	96.02	535,7k	Landmark
ST-BLN w/MCD	**95.47**	**132.3k**	Landmark
ST-BLN wo/MCD	93.19	132.3k	Landmark
PST-BLN w/MCD	**93.34**	**9.79k**	Landmark
PST-BLN wo/MCD	93.1	9.79k	Landmark

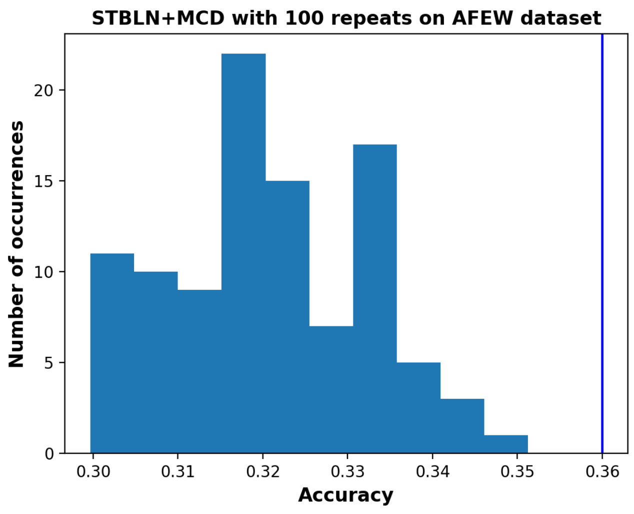


Figure 16: The distribution of 100 classification accuraceis obtained by the ST-BLN w/MCD method on AFEW dataset. The vertical line in the left side indicates the classification accuracy obtained by the ensembled predictions.

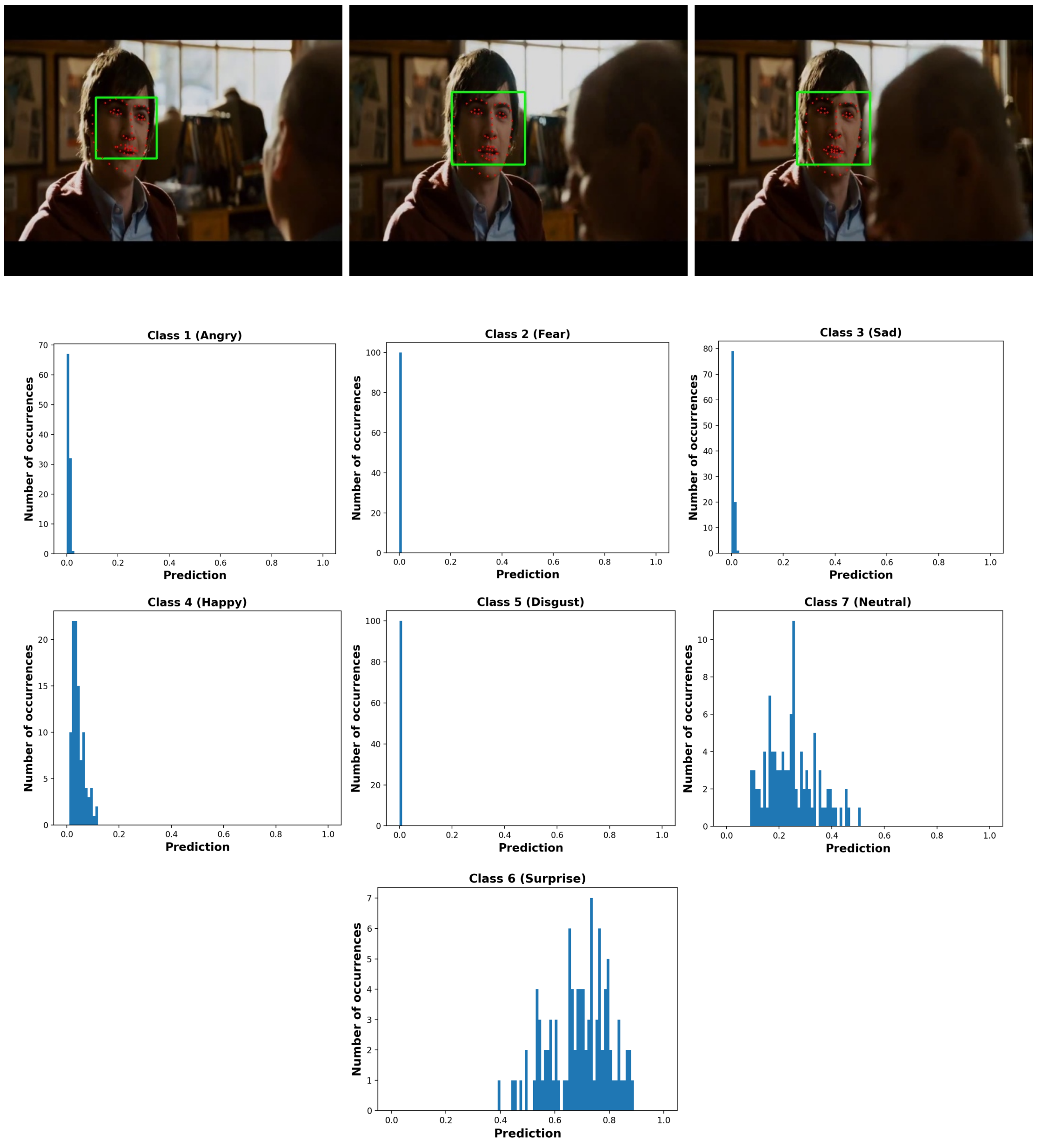


Figure 17: Illustration of 3 frames of a sample video in AFEW dataset expressing ‘Surprise‘, top row, and the distribution of 100 predictions for each class, obtained by our proposed ST-BLN model.

performance on the landmark-based facial expression recognition task. The experimental results show that the ST-BLN and PST-BLN outperform other all other state of the art methods in terms of computational complexity while having similar or better classification performance compared to other GCN-based methods. The best classification accuracy on all three dataset belongs to a video/image-based method; however, since these methods train a CNN/RNN architecture like ResNet-18, VGG16, LSTM, C3D, they have more computational complexity.

To capture the model's uncertainty, we evaluated both ST-BLN and PST-BLN models with activated dropout layers during the inference and we repeated the inference for 100 times on each sample to get 100 different prediction vectors. ST-BLN w/MCD and PST-BLN w/MCD denote the model's classification accuracy obtained as the mean of 100 different predictions and ST-BLN wo/MCD and PST-BLN wo/MCD report the classification accuracy obtained by performing the inference only once. The results show that the model achieves better performance when it ensembles the predictions of 100 models rather than performing the inference only once. Fig. 16, shows the distribution of 100 classification accuracy values of the ST-BLN w/MCD on AFEW dataset. The classification accuracy obtained by the ensembled predictions is shown by a vertical line in the left side of the figure which is around 4% better than the mean accuracy.

Fig. 17 illustrates 3 frames of a sample video and the prediction distribution for each expression class. The model classifies this sample correctly in the Surprise class with mean probability of 0.69 while it can also be classified in Neutral and Happy classes with mean probabilities of 0.24, 0.04, respectively. The variance of the model predictions on classes Surprise, Neutral and Happy is 0.1, 0.9, 0.02, respectively which is interpreted as the model's uncertainty on each class.

5 Deep speech and biosignals analysis and recognition

5.1 Deep Gaussian Filtering for Improving Biosignal Timeseries Classification

5.1.1 Introduction, objectives, and state-of-the-art

Deep Learning (DL)-based approaches have been employed for several time series analysis applications in different domains with many problems being successfully tackled, e.g., novel DL architectures for classifying electrocardiograms (EEGs) [124]. Although DL approaches had great success, such time series problems still pose significant challenges that are hard to tackle by conventional DL methods. The inherently noisy and non-stationary nature of many time-series data make the training process unstable, which usually results in their convergence to a sub-optimal solution. This has led to the development of a wide range of methods, such as deep normalization layers [122]. However, DL models are still vulnerable to such phenomena, as we highlight later in this section.

In this work we aim to integrate a low-pass filtering procedure as an end-to-end trainable component of DL models that can perform denoising that has been adapted to the task at hand. Low pass filters are widely used to smooth the noisy signal and provide a more appropriate form of the data for further processing/analysis. Yet, the cut-off frequency of such filters is fixed and usually predetermined, leading to concealing important information from the model that otherwise could have been exploited. Applying a trainable low-pass filter that can learn how to extract meaningful features can overcome these limitations. For example, learning the cut-off frequency of the filters could be part of the learning process of DL models and be adapted to the task at hand, significantly improving the performance of such models. To this end, AUTH worked towards developing a deep high-order Gaussian filtering method for time series analysis. We construct a discrete approximation of a Gaussian filter, which is trainable and can be learned in an end-to-end fashion through backpropagation. To further improve the performance of the proposed method, we propose extracting multiple features, using high-order derivatives of the Gaussian function by convolving the derivatives of the Gaussian filters with the input signal, leading to a smooth high-order representation of the input. The proposed series of layers are lightweight, since they consist only of a few trainable parameters, and can significantly improve time series classification, as demonstrated in the experimental evaluation.

5.1.2 Description of work performed so far

Let $\mathbf{x}$ be an input timeseries and $f_{\mathbf{W}}(\mathbf{x})$ be a DL model trained for classification tasks, where $\mathbf{W}$ are the trainable parameters of the model. This model can be trained for classification tasks using gradient descent and an appropriate loss function, such as the cross entropy loss. Instead of directly feeding the input $\mathbf{x}$ to the DL model $f_{\mathbf{W}}(\cdot)$, the proposed method uses low pass filters to extract the filtered signal, along with its filtered high order derivatives, which can be more appropriate for the task at hand. We define three filters, one for the Gaussian function and its first two derivatives, as shown below:

$$g(x,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}, \tag{15}$$

$$\frac{\partial g(x,\sigma)}{\partial x} = \frac{-x}{\sigma^3\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}, \tag{16}$$

and

$$\frac{\partial^2 g(x,\sigma)}{\partial x^2} = \frac{x^2-\sigma^2}{\sigma^5\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}, \tag{17}$$

respectively. Note that instead of calculating the signal's derivative and performing convolution with a Gaussian filter, we can directly perform convolution with the filter's derivative, exploiting the properties of convolution. Using higher order derivatives allows for extracting more information from the input signal that could improve the performance of the subsequent DL model. Therefore, the kernels, which are discrete approximations of the above functions are convolved with the input signal and their outputs are concatenated and fed to the rest of the network. The kernels have the size of k, which is an odd number. To keep the same dimensionality between the signal and the layer's outputs, padding was applied to the input by expanding its first and last value. Note that the denominator of the first term in each function can be skipped as it only provides a scaling factor, which is redundant in the case in which trainable scaling factors are used.

The proposed method aims to directly learn the optimal σ, which controls the width of the Gaussian function, i.e., the amount of denoising that is applied to the signal. Note that a different parameter is used for each kernel. Therefore, the proposed method does not lead to a significant increase in models parameters, apart from a slight increase in feed-forward time due to the additional filtering operation. However, as it is demonstrated in the next section, this process can lead to significantly higher gains, often allowing to lower the complexity of the subsequent model. Therefore, the proposed method can lead to an overall decrease in computational complexity. Furthermore, every operation that has been applied is differentiable and therefore the gradients can be directly used to optimize the parameter σ through back-propagation, while the existing highly-optimized GPU-based convolution algorithms can be easily employed to further improve the performance of the proposed method.

5.1.3 Performance evaluation

Electrocardiograms (ECG) are produced by placing different electrodes on the skin and entail information about the heart. They are commonly used by doctors to identify different heart diseases. Past research has used the labeled data collected to build DL models for automated disease classification [71]. For all the conducted experiments, the proposed method is evaluated in binary disease classification tasks (healthy time series vs non-healthy time series) using the following datasets: a) PTB Diagnostic ECG (PTBDB) dataset [2] and b) MIT-BIH Arrhythmia (MITBIH) dataset.

Table 22: Deep Gaussian Filtering: Baseline evaluation with four different models and 2 datasets. The average of each metric for 5 different runs is reported.

Model	**Accuracy**	**F1**	**Loss**	**Precision**	**Recall**
Dataset PTBDB					
MLP	0.9285	0.9096	0.1887	0.9118	0.9117
Proposed + MLP	**0.9367**	**0.9190**	**0.1873**	**0.9266**	**0.9154**
CNN	0.9412	0.9269	0.1563	0.9239	0.9332
Proposed + CNN	**0.9542**	**0.9425**	**0.1362**	**0.9424**	**0.9452**
LSTM	0.9553	0.9429	0.1280	0.9505	0.9378
Proposed + LSTM	**0.9749**	**0.9683**	**0.0769**	**0.9694**	**0.9683**
ResNet	0.9241	0.9047	0.1996	0.9051	0.9084
Proposed + ResNet	**0.9446**	**0.9299**	**0.1757**	**0.9319**	**0.9320**
Dataset MITBIH					
MLP	0.9469	0.7178	0.2071	0.6214	0.8496
Proposed + MLP	**0.9637**	**0.8202**	**0.1319**	**0.7639**	**0.8854**
CNN	0.9603	0.8096	0.1485	0.7378	**0.8969**
Proposed + CNN	**0.9671**	**0.8392**	**0.1246**	**0.7938**	0.8901
LSTM	0.9658	0.8221	0.1297	0.7518	0.9070
Proposed + LSTM	**0.9741**	**0.8603**	**0,0949**	**0.8123**	**0.9142**
ResNet	0.9541	0.7808	0.1753	0.7073	0.8717
Proposed + ResNet	**0.9661**	**0.8304**	**0,1251**	**0.7850**	**0.8814**

For all the models the learning rate was set to 10^{-3}, while the Adam optimizer [77] was used for updating the model. The total epochs were set to 20 and 10 for the PTBDB and MITBIH datasets, respectively. Each model was evaluated 5 different times and the average evaluated metrics are reported. First, we evaluated the proposed method using a few baseline models: 1) a multilayer perceptron (MLP) with 3 hidden layers having 16 neurons each. The activation function for the hidden layers was PReLU. 2) A ResNet [48] having a similar structure with the MLP. 3) A convolutional network (CNN) comprising of one layer convolutional layer with 16 filters of size 3 whose output was fed to the MLP as previously defined. 4) A Long Short Term Memory (LSTM) network with 1 layer, the output of which was also fed to an MLP as previously defined.

In Table 22, we report the experimental evaluation, where we compare different baseline architectures with and without the proposed trainable Gaussian approach on two datasets. In virtually all the evaluated cases, the proposed method improves the accuracy over the baseline. The ability of the proposed method to be combined with models proposed in the literature was also evaluated in Table 23. Again the experimental evaluation highlights the ability of the proposed method to improve over state-of-the-art architectures proposed in the literature, demonstrating the potential of the proposed method for time series analysis tasks. Finally, in Table 24 we demonstrate that the proposed method can achieve competitive performance even when combined with more lightweight architectures, providing a way to further reduce the complexity of the deployed models. The backbone network architecture presented in [42] was employed for the conducted experiments.

Table 23: Deep Gaussian Filtering: Evaluating the ability of the proposed method to be combined with state-of-the-art models proposed in the literature. The average of each metric for 5 different runs is reported.

Model	**Accuracy**	**F1**	**Loss**	**Precision**	**Recall**
Dataset PTBDB					
[84]	0.9855	0.9815	**0.0503**	0.9844	0.9795
Proposed + [84]	**0.9873**	**0.9840**	0.0602	**0.9861**	**0.9826**
[167]	0.9759	0.9696	0.0812	0.9699	0.9705
Proposed + [167]	**0.9920**	**0.9901**	**0.0465**	**0.9921**	**0.9886**
[159](feed)	0.9940	0.9926	0.0243	0.9927	0.9929
Proposed + [159](feed)	**0.9958**	**0.9948**	**0.0179**	**0.9947**	**0.9950**
[159](concat)	0.9918	0.9898	0.0261	0.9901	0.9900
Proposed + [159](concat)	**0.9943**	**0.9927**	**0.0255**	**0.9933**	**0.9925**
[71]	0.9901	0.9874	0.0323	0.9876	9878
Proposed + [71]	**0.9934**	**0.9918**	**0.0286**	**0.9932**	**0.9907**
Dataset MITBIH					
[84]	0.9787	0.8834	0.0834	0.8496	0.9199
Proposed + [84]	**0.9799**	**0.8945**	**0.0802**	**0.8679**	**0.9228**
Model [167]	0.9751	0.8640	0.0947	0.8235	0.9086
Proposed + Model [167]	**0.9826**	**0.9084**	**0.0695**	**0.8768**	**0.9424**
Model [159] (feed setup)	0.9830	0.9076	0.0648	0.8815	**0.9354**
Proposed + Model [159] (feed setup)	**0.9847**	**0.9143**	**0.0589**	**0.8964**	0.9330
Model [159] (concat setup)	0.9836	0.9102	0.0634	0.8834	0.9388
Proposed + Model [159] (concat setup)	**0.9840**	**0.9139**	**0.0597**	**0.8891**	**0.9401**
Model [71]	0.9820	0.9034	0.0655	0.8745	0.9342
Proposed + Model [71]	**0.9828**	**0.9068**	**0.0652**	**0.8871**	**0.9273**

Table 24: Deep Gaussian Filtering: Evaluating the performance of the proposed method when combined with [42] and appropriately tuned to reduce the number of parameters for the baseline model. The average of each metric of 5 different runs is reported. The total number of parameters is also reported.

Model	**Accuracy**	**F1**	**Loss**	**Precision**	**Recall**	**Parameters**
Dataset PTBDB						
Lightweight	0.9886	0.9856	0.0305	0.9876	0.9844	3.1M
Lightweight + Proposed	**0.9948**	**0.9933**	**0.0197**	**0.9955**	**0.9915**	3.1M
Baseline	0.9937	0.9932	0.0199	0.9930	0.9914	5.5M
Dataset MITBIH						
Lightweight	0.9818	0.9040	0.0643	0.8715	0.9390	3.1M
Lightweight + Proposed	**0.9846**	**0.9124**	**0.0581**	**0.8734**	**0.9551**	3.1M
Baseline	0.9818	0.9052	0.0661	0.8707	0.9429	5.5M

5.2 Hybrid Speech Command Recognition Models with Self-organized Operational Layers

5.2.1 Introduction, objectives, and state-of-the-art

Spoken commands recognition (SCR), also referred to in literature as keyword spotting (KWS), 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. SCR is also of particular importance to OpenDR project, as it offers possibilities of basic voice-driven communication and control in a computationally limited environment (such as on-board devices of robots).

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 beyond what was ever achievable with classical methods. These deep models can 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 factors 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 full exploitation of the benefits of highly accurate, but complex neural models. As robotics applications mainly target computationally constrained environments, fast and lightweight implementations are required for practical operation.

To tackle the problem of computational complexity on embedded devices, two general approaches are commonly used. One is of network compression [19, 102], whereas the state of the art deep models are compressed by a variety of methods to reduce the computational costs without severely degrading the task performance (e.g. recognition 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 have previously proposed and validated a concept of self organized operational neural networks (SelfONNs) [79, 141]. We now build upon previously reported work and propose to hybridise existing SCR architectures with SelfONN formulation, allowing us to keep the advantages of the models' original designs while expanding on their learning capacity and flexibility of representation. We are currently considering MatchBoxNet [98] as a base architecture, due to its highly computationally efficient design and simultaneously high recognition accuracy, but the core approach is applicable to a wide array of other models. We describe the specific modifications and their outcomes in the following subsections.

5.2.2 Description of the work performed so far

Operational Neural Networks (ONNs) [78] 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 require an expensive greedy search to find the optimal operators that achieve the best possible accuracy. The performance is also highly dependent on the selection of the initial operator set dictionary. In order to resolve these issues, SelfONNs were proposed in [79], defining 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 a greedy search within that set to find the optimal operators. SelfONNs were subsequently applied together with quadratic form kernels for the speech command recognition tasks [141] as part of OpenDR project, the development and integration concluding in the previous reporting period.

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. A standard convolutional layer computes the convolution of the input with the layer weights as follows:

$$Y = X \circledast W + b, \tag{18}$$

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

ONN formulation generalizes the above to the following form:

$$Y_{\text{ONN}} = \Psi(X, W) + b, \tag{19}$$

where Ψ is an arbitrary nodal operator and may be a combination of different functions. If we set Ψ function equal to the regular convolution operator ($\circledast$), the ONN layer yields an equivalent of the standard convolution.

In SelfONNs, instead of using a specific function Ψ as the nodal operator (and therefore having to select it), an operation is instead approximated with a truncated Taylor series expansion. So, the SelfONN layer is defined as:

$$\begin{aligned} Y_{\text{SelfONN}} &= W_{T_0} + (X - A) \circledast W_{T_1} + (X - A)^2 \circledast W_{T_2} + \\ &\quad \ldots + (X - A)^q \circledast W_{T_q} + b_S \\ &= \sum_{i=0}^{q} (X - A)^i \circledast W_{T_i} + b_S, \end{aligned} \tag{20}$$

where A is the point around which Ψ is expanded. $W_{T_i} = \frac{1}{i!}\frac{\partial \Psi}{\partial X^i}$ is the i'th order partial derivative of Ψ with respect to the input, and q controls the order of approximating polynomial. b_S is the bias term associated with the SelfONN layer. It is argued in [79] 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 Ψ, 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).

MatchboxNet [98] is a computationally efficient model for speech command recongition, which has been previously integrated into OpenDR toolkit as a baseline. It takes advantage of 1D time-channel separable convolutions to reduce the overall number of parameters. As is common for SCR models, MatchboxNet takes as input the mel-frequency cepstral coefficients (MFCC) of the audio signal. Dimensionality of such features may vary, but we follow the original model and use 64 MFCC values, calculated from 25 ms windows with 10 ms overlap. Structurally MatchBoxNet is composed of B residual blocks (where B is an adjustable parameter for model scaling), preceded by a prologue layer and followed by three epilogue layers. Prologue and epilogue both use standard convolutions, while each residual block contains R sub-blocks, where 1D time-channel separable convolution is followed by 1x1 pointwise convolution. Batch normalization is used throughout the model.

The proposed hybridization involves replacing convolutional layers of the model with Self-ONN layers. Specifically, the following configurations have been considered so far:

- **A**: original MatchboxNet
- **B**: prologue layer replaced with SelfONN, $q = 7$
- **C**: final epilogue layer replaced with SelfONN, $q = 7$
- **D**: both prologue and epilogue replaced with SelfONN, $q = 5$
- **E**: both prologue and epilogue replaced with SelfONN, $q = 7$
- **F**: pointwise convolutions inside residual blocks replaced with SelfONN, $q = 7$

The network structure (specifically configuration **E**) is depicted on Figure 18. The only additional constraint imposed by the substitution of layers is the assumption for the SelfONN input to be zero centered. While for the prologue layer this can be addressed by normalizing the data, intermediate inputs inside the network also violate this assumption. For this reason we additionally insert hyperbolic tangent functions into the model just before the SelfONN layers, where appropriate. This ensures the values to be between -1 and 1, preserving the Taylor series approximation.

5.2.3 Performance evaluation

Preliminary evaluation results for the hybrid models are reported in the Table 25. Google Speech Commands dataset [162] was used for experiments as a de-facto standard for speech command recognition tasks. Every reported value is an average of 3 independent runs. Other training parameters follow the standard MatchboxNet setup as closely as possible for compatibility and reproducibility. Specifically, Novograd optimizer is used with $\beta_1 = 0.95$ and $\beta_2 = 0.5$. The learning rate is determined via Warmup-Hold-Decay scheduler, with ratios of different scheduling stages being 5%, 45% and 50% respectively and the learning rate being capped in the interval between 0.001 and 0.05. The training is performed for 500 epochs.

While configurations B through D demonstrate results comparable to the baseline, configuration E proves capable of surpassing the original MatchboxNet. A wider range of model configurations and settings are being investigated to establish the potential benefits of using the hybrid approach.

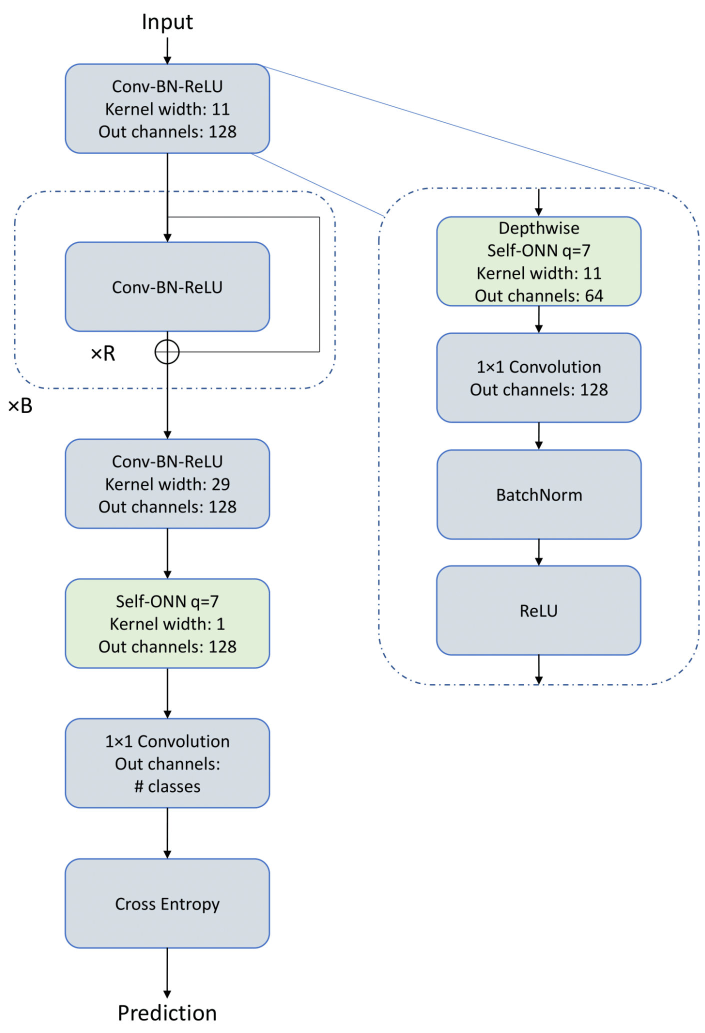


Figure 18: Example of the hybrid MatchboxNet model with SelfONN layers.

Table 25: Classification accuracy of different model configurations on Google Speech Commands dataset

Configuration	**Accuracy, %**
A (baseline)	97.1789
B	97.1763
C	96.3570
D	96.6057
E	**97.7105**
F	93.8259

5.3 Biosignal Classification with Codeword-Temporal Self-Attention Neural Bag of Features

5.3.1 Introduction, objectives, and state-of-the-art

Timely diagnostics of potential heart abnormalities, such as atrial fibrillation or other cardiovascular diseases is an important problem in the modern world, with a multitude of solutions proposed to address it. Most of these methods operate on multi-dimensional time-series data, where the signal is represented by sequence of features, where multiple features are present at each timestamp. One successful approach for such problems has been developed during the first year of OpenDR, where a 2D attention for matrix data was developed as an extension to the Neural Bag-of-Features (NBoF) model, including three different formulations with different interpretations, as a way to improve the robustness of the model. In our work, we aim to extend this approach further, by proposing a codeword-temporal attention formulation as well as reformulating independent codeword and temporal attention formulations.

Neural Bag of Features is a neural extension of the Bag of Features feature extraction algorithm. NBoF receives as input a variable-size representation and quantizes it into a fixed-size histogram representation. Quantization is performed using a learned dictionary that can be optimized jointly with the whole architecture in an end-to-end manner. The extracted histogram representations, known as codewords, are aggregated by averaging. Several feature quantization approaches have been proposed to date, including those based on Radial Basis Function (RBF) and hyperbolic kernel [121].

During the first year of OpenDR, an attention module for sequence data was developed and utilized together with Neural Bag of Features formulation for heart anomaly classification problems. The developed methodology defines 2D-attention, referred to as 2DA, that can be utilized as a plug-in extension in a variety of sequence data problems and architectures. Specifically, codeword attention, temporal attention, an input attention variants were proposed to highlight most relevant features of intermediate representations, temporal timestamps, or input data features, respectively. During the second year, work towards extending this approach has been performed, as described in Section 6.2.2.

The goal of codeword attention is to highlight most relevant codewords obtained at quantization step of the NBoF model while suppressing the non-relevant ones, under the assumption that the output of each quantization neuron contributes differently to the final prediction. Given the output of the quantization step $\Phi \in \mathbb{R}^{K \times N}$, an attention mask $\mathbf{A} \in \mathbb{R}^{K \times N}$ of attention weights is applied to the features Φ in order to highlight or suppress its rows, i.e., codewords, by aplying the 2DA to Φ^T. Similarly, 2DA can be applied directly on the input of NBoF rather than its quantized output in order to improve the robustness of the model towards noise. Since we would like to highlight individual series in the input data, the process is similar to codeword attention, and 2DA is applied to $\mathbb{X}^T$. This type of attention is referred to as input attention. Temporal attention aims to highlight relevant timestamps in the sequence during the aggregation step of he NBoF model. Formally, it is achieved by applying 2DA on columns of Φ.

5.3.2 Description of the work performed so far

Formally, 2DA is defined as follows. Given a feature representation Φ, 2DA learns an attention matrix $\mathbf{A}$:

$$A = softmax(\Phi \mathbf{W}), \tag{21}$$

where softmax() function is applied row-wise to encourage competition between columns of Φ, and $\mathbf{W}$ is a learnable weight matrix with diagonal elements fixed at $\frac{1}{N}$. The learnt attention matrix i subsequetly applied as:

$$\mathscr{F}_{2DA}(\Phi) = \tilde{\Phi} = \alpha(\Phi \odot \mathbf{A}) + (1-\alpha)\Phi, \tag{22}$$

where α is a learnt parameter controlling the strength of attention matrix.

Although the proposed 2DA approach addresses certain limitations of the NBoF model in terms of highlighting most relevant attributes in the quantized feature representation, we hypothesise that further improvement can be achieved by reformulating the attention learning methodology.

One limitation of previously-proposed 2DA attention mechanism is that attention is applied separately to either codebook or temporal dimensions. Even if both attention masks are applied simultaneously, being learned independently, this approach does not take into account potential relationships of learned codewords with the temporal representations. At the same time, they are not necessarily independent in real scenario, as certain codewords can have different importance at different timestamps. We therefore hypothesize that learning of joint codeword-temporal attention map can be beneficial for learning better feature representations and therefore assist in classification task.

Formally, we define the codeword-temporal attention as follows, building on top of the well studied self-attention module. Considering a NBoF-learned feature representation $\Phi \in \mathbb{R}^{K \times N}$, where K denotes the number of codewords and N denotes the temporal length, we obtain the attention matrix by quantifying the relations of codeword and temporal features in a joint learnt space. Formally, we define two learnable projection matrices $\mathbf{W}_q^n \in d \times N$, $\mathbf{W}_k^n \in d \times K$ and project the representation Φ temporally and codeword-wise into a joint d-dimensional space.

$$\begin{aligned} \mathbf{q}_n &= \Phi \mathbf{W}_q^{nT} \in K \times d \\ \mathbf{k}_n &= \Phi^T \mathbf{W}_k^{nT} \in N \times d \end{aligned} \tag{23}$$

Further, we calculate the scaled dot-product similarity between representations learned from temporal dimension and the ones learned from the codebook and apply an activation function σ, such as softmax, to scale the values and promote competition between different elements. The learnt attention matrix is applied element-wise to the input feature representation. Following the widely-used definition of multi-head self-attention [156], n attention matrices can be calculated independently, with the outputs of all heads subsequently concatenated.

$$A_n = \sigma(\frac{\mathbf{q}_n \mathbf{k}_n^T}{\sqrt{d}}) \in K \times N \tag{24}$$

$$\begin{aligned} \tilde{\Phi}_n &= \alpha_n \Phi + (1-\alpha_n) A_n \odot \Phi \\ \tilde{\Phi} &= [\tilde{\Phi}_1, ..., \tilde{\Phi}_n] \end{aligned} \tag{25}$$

A similar idea can be further developed into enhancing the independent codebook and temporal attetions in 2DA. In the current definition, the projection matrix $mathbfW$ outlined in equation 21 is fully learnt from scratch, with a role of highlighting relevant codewords or temporal features in Φ. Instead of learning this weight matrix directly from scratch, we propose to follow a different self-attention based approach and learn the attention matrix by quantifying the relative importance of each codeword or temporal dimension in a learnt latent space.

That is, considering the codeword attention, we define two projection matrices $\mathbf{W}_q^n \in d \times N$ and $\mathbf{W}_k^n \in d \times N$ from which latent representations of Φ are learnt as:

$$\begin{aligned}\mathbf{q}_n &= \Phi \mathbf{W}_q^{nT} \in K \times d \\ \mathbf{k}_n &= \Phi \mathbf{W}_k^{nT} \in K \times d\end{aligned} \tag{26}$$

Following this, we can calculate the codeword attention as a $K \times K$ matrix following equation 24. The learnt attention matrix is subsequently multiplied with a feature representation Φ to highlight the most relevant codewords and multi-head approach can be followed here as well, as follows:

$$\begin{aligned}\tilde{\Phi}_n &= \alpha_n \Phi + (1 - \alpha_n) A_n \Phi \\ \tilde{\Phi} &= [\tilde{\Phi}_1, ..., \tilde{\Phi}_n].\end{aligned} \tag{27}$$

Similarly, temporal attention can be calculated by transposing the feature representation Φ, leading to $N \times N$ attention matrix encoding relative importance of each temporal dimension.

5.3.3 Performance evaluation

To evaluate the performance of the developed methodology, we perform experiments on time series classification of biosignals. Specifically, we utilize Phonocardiogram (PCG) and Electrocardiogram (ECG) signals to predict cardiovascular diseases. PCG signal is used to evaluate the hemodynamic status of the patient and predict presence of any cardiovascular abnormalities, and we employ Physionet challenge dataset for this purpose. Additionally, on this dataset we predict the quality of the given phonogram (high vs low). For evaluation we perform experiments on 5 second segments of the recordings and extract 24-band Mel-spectrogram as features. AF dataset poses the problem of detection of atrial fibrillation, with the task of classifying the ECG signals into normal sinus rhythm, atrial fibrillation, alternative rhythm, and noise. Here, we utilize 30-second segments and train the model on the raw signals.

In both cases, we preprocess the data with several 1d-convolutional layers prior to passing the feature to the NBoF model. In addition to previously-described NBoF formulation, we utilize recently-propozed TNBoF formulation as well. The average F1-score over 3 or 5 folds for PCG and AF datasets, respectively, are reported in Table 26. Here, CA and TA correspond to standard codebook and temporal attention, and CsA, TsA, and CTsA correspond to codebook self-attention, temporal self-attention, and joint codebook-temporal self-attention. We employ architecture with 3 heads and dropout applied on the attention matrix of CsA and TsA, and report results with latent spaces of two different dimensionalities of 512 and 1024.

As can be seen, in most cases codebook-temporal attention provides competitive performance outperforming the vanilla NBoF, and both codebook and temporal attentions. Additionally, it can be seen that in most cases conventional 2DA in both codebook and temporal formulations is outperformed by the proposed self-attention based counterparts. Further experiments evaluating the proposed approach remain for future work.

Table 26: Average F1 score of proposed self-attention based approaches (CsA, TsA, CTsA) and competing approaches (CA, TA)

	PCG-AD	PCG-QE	AF
NBoF	87.88 + 0.39	72.61 + 1.35	76.59 + 1.24
NBoF-CA	88.05 + 0.65	73.7 + 0.58	77.02 + 2.11
NBoF-TA	87.25 + 0.44	73.51 + 1.54	77.24 + 2.08
NBoF-CsA	**88.55 + 0.18**	74.12 + 0.8	76.77 + 1.71
NBoF-CsA	87.97 + 0.59	74.07 + 0.79	77.0 + 1.84
NBoF-TsA	87.79 + 0.27	74.45 + 0.76	76.32 + 2.3
NBoF-TsA	87.88 + 0.51	**74.7 + 0.8**	76.04 + 1.83
NBoF-CTsA	87.27 + 0.32	73.85 + 0.6	**77.75 + 1.31**
NBoF-CTsA	87.53 + 0.66	74.16 + 0.68	76.62 + 1.77
TNBoF	87.94 + 0.82	73.05 + 1.26	76.77 + 1.16
TNBoF-CA	87.31 + 0.79	73.32 + 0.46	77.06 + 1.74
TNBoF-TA	87.38 + 0.58	74.0 + 1.0	77.33 + 2.19
TNBoF-CsA	87.48 + 0.87	**74.08 + 1.07**	77.95 + 1.78
TNBoF-CsA	87.95 + 0.87	73.9 + 1.13	77.25 + 1.64
TNBoF-TsA	87.68 + 0.61	73.45 + 0.45	76.2 + 0.75
TNBoF-TsA	87.43 + 0.93	73.49 + 1.59	77.07 + 1.16
TNBoF-CTsA	88.02 + 0.17	74.02 + 0.98	77.01 + 1.15
TNBoF-CTsA	**88.03 + 0.87**	73.61 + 0.57	**78.0 + 1.74**

6 Multi-modal human centric perception and cognition

6.1 Learning to ignore in Convolutional Neural Networks

6.1.1 Introduction and objectives

Attention mechanisms in Convolutional Neural Networks (CNNs) are developing rapidly due to their improved performance in a wide range of computer vision tasks. Majority of these approaches aim to explicitly identify and highlight relevant regions of the image and pass the attended representation to further layers of the network. On the other hand, it can be argued that explicit learning of the parts of the image relevant to the given task is generally more challenging than learning which parts of the image are less relevant and, thus, should be ignored. Especially in computer vision problems, oftentimes many easy-to-identify patterns of irrelevant features can be present in a data sample. For example, image regions located closer to the center of the task are generally more likely to contain useful information compared to the borderline pixels. Following this intuition, we propose to reformulate the attention mechanism in CNNs to learn to ignore instead of learning to attend. More specifically, we propose a learning mechanism aiming to explicitly learn irrelevant information in the scene and suppress it in the produced representation, keeping only relevant attributes. The resulting implicit attention can be incorporated into existing attention mechanism and we specifically validate our method using two of the widely-used attention mechanisms, namely, Squeeze and Excitation (SE) block [59] and Convolutional Block Attention Module (CBAM) [165]. Furthermore, as utilization of these attention approaches have been successfully utilized for multimodal fusion, we extend our approach to this domain and present an approach for multimodal fusion in vision tasks via ignoring.

A summary of this work is provided hereafter. The corresponding publication is listed below, and can be found in Appendix 8.6.

- [86] Laakom, F., Chumachenko, K., Raitoharju, J., Iosifidis, A., & Gabbouj, M, "*Learning to ignore: rethinking attention in CNNs*", British Machine Vision Conference 2021.

6.1.2 Summary of state of the art

Attention in computer vision is inspired from the human visual system, leading to development of a wide range of attention approaches aimed at improving the performance of models in a variety of tasks. A set of these methods is directed at Convolutional Neural Networks, and aim to learn to select and emphasize relevant features or regions in images. This is conventionally achieved by learning an attention mask over an intermediate feature representation, whether spatial or channel-wise, where the mask encodes the importance of each given attribute. The attention mask is subsequently applied to the feature representation, hence emphasizing the relevant features and making the learning problem easier for the subsequent layers.

Notable works in this field include Squeeze-and-Excitation block (SE) [59] and Convolutional Block Attention Module (CBAM) [165], where SE learns attention weights channel-wise, and CBAM extends the approach of SE to include spatial attributes. Specifically, in SE, an attention mask is learned based on global average-pooled features of intermediate representations and applied at multiple layers of the ResNet architecture. CBAM enriches the SE mechanism by additional max-pooled input, hence leading to learning of spatial attention. The learned attention weight masks are then applied channel-wise or pixel-wise to corresponding feature maps. Futhermore, utilization of channel-wise attention has been shown to be beneficial for multimodal fusion in multi-stream architectures on a wide range of tasks [69, 142]. In these methods, attention is used for knowledge transfer between two streams of the multi-modal architecture, where a joint feature representation is learnt between the streams similarly to "squeeze" stage of SE-block, and the joint representation is subsequently "excited" to each of the streams, resulting in two attention matrices applied to each of the two modalities.

Concurrently, some works have focused on the ignoring paradigm rather than explicit attention in a variety of applications. This idea has been shown useful in saliency estimation [4] and color constancy [87], where auxiliary variables are used to encode prior knowledge on regions to be ignored, such as dark regions, as it is assumed that they are less-likely to contain salient object. Other applications include machine translation [44] and domain adaptation [175].

6.1.3 Description of work performed so far

We present a methodology that reformulates conventional attention into learning-by-ignoring paradigm. Formally, given a feature map $\mathbf{F}$, attention mask in CNN can be defined as follows:

$$\mathbf{F}' = \mathbf{F} \otimes f_\theta(\mathbf{F}), \tag{28}$$

where $\mathbf{F}'$ is the attended feature map output, $\otimes$ is the element-wise multiplication and $f_\theta(\cdot)$ is an attention function with learnable parameters θ, which takes as input a feature map $\mathbf{F}$ and returns an attention mask $f_\theta(\mathbf{F}) \in [0,1]$. This mask is then element-wise multiplied with the original map $\mathbf{F}$ in order to produce the output map $\mathbf{F}'$. The mask $f_\theta(\mathbf{F})$ is expected to identify relevant spatial or channel information and output the 'importance score' for each attribute, producing high response for most relevant regions and smaller values for regions of lesser interest. This can be seen as an explicit attention mechanism, where the model $f_\theta(\cdot)$ learns to directly identify and highlight relevant information.

In our ignoring approach, we train the model to predict irrelevance of features, rather than their importance, and by this we expect to simplify the training objective, hence leading to improved performance. Our approach consists of a function which learns to identify irrelevant or confusing parts of the feature map in order to suppress them, followed by inversion of predicted irrelevance scores. Formally, this can be formulated as follows:

$$\mathbf{F}' = \mathbf{F} \otimes T(g_\theta(\mathbf{F})), \tag{29}$$

where $g_\theta(\cdot)$ is a function with learned parameters θ that is expected to learn to highlight information in the feature map that is irrelevant or confusing for the prediction. This can be seen as an *ignoring mask* that outputs high values for attributes and regions that should be suppressed in the feature map. The function $T(\cdot)$ is a function with an output $T(x)$ inversely proportional to x, hence flipping the learned ignoring mask and transforming it into an attention mask. Similarly to Eq. (28), the final feature map $\mathbf{F}'$ is obtained by element-wise multiplication of the input map $\mathbf{F}$ and the flipped ignoring mask $T(g_\theta(\mathbf{F}))$.

Given an ignoring mask $g_\theta(\mathbf{F})$, the function $T(\cdot)$ can be any function satisfying the condition of being inversely proportional to its input and bounded between $[0,1]$. We propose three variants:

$$T_1(x) = 1 - \alpha x, \tag{30}$$

$$T_2(x) = sigmoid(\frac{1}{x}), \tag{31}$$

$$T_3(x) = sigmoid(-x). \tag{32}$$

The first variant $T_1(\cdot)$ linearly converts the ignoring mask to an attention one, and α is a hyperparameter controlling this linear scaling. The extreme case $\alpha = 0$ corresponds to the extreme case $\mathbf{F}' = \mathbf{F}$, i.e., none of the features are emphasized or suppressed. For the second and third variants T_2 and T_3, a sigmoid function is applied to ensure that the output is bounded between $[0,1]$.

The main difference between implicit and explicit attention formulations is the presence of a flipping function $T(\cdot)$. It can be seen from Eq. (28) and Eq. (29) that $f_\theta(\cdot)$ can be directly replaced by $T(g_\theta(\cdot))$. This makes it straightforward to reformulate any existing explicit attention method to learn to ignore instead of learning to attend by applying an inversion function $T(\cdot)$ on top of the learned mask. This way, the model $g_\theta(\cdot)$ can be learned as the model $f_\theta(\cdot)$ in conventional attention methods, while its parameters will be optimized to detect irrelevant or confusing regions instead of relevant ones. To utilize our approach for multimodal fusion, we rely on Squeeze-and-Excitation (SE) block, defined as follows:

$$f_\theta(\mathbf{F}) = \sigma(\mathbf{W}_2\delta(\mathbf{W}_1 GAP(\mathbf{F}))), \tag{33}$$

where $GAP(\cdot)$ denotes Global Average Pooling, δ is a ReLU activation, σ is the sigmoid function, $\mathbf{W}_1 \in \mathbb{R}^{c\times\frac{c}{r}}$ and $\mathbf{W}_2 \in \mathbb{R}^{\frac{c}{r}\times c}$ are linear layers, c is the number of channels in $\mathbf{F}$, and r is the reduction rate in the bottleneck block. Given the output $f_\theta(\mathbf{F})$, the attended feature map is obtained by applying the learned mask element-wise between corresponding channels. To incorporate our ignoring paradigm into SE, we apply $T(\cdot)$ to the output $f_\theta(\mathbf{F})$, hence transforming its objective into learning features that should be ignored. Specifically, we define the three variants as: $f^1_\theta(\mathbf{F}) = 1 - \alpha\sigma(\mathbf{W}_2\delta(\mathbf{W}_1 GAP(\mathbf{F})))$; $f^2_\theta(\mathbf{F}) = \sigma(\frac{1}{\sigma(\mathbf{W}_2\delta(\mathbf{W}_1 GAP(\mathbf{F})))})$; $f^3_\theta(\mathbf{F}) = \sigma(-\mathbf{W}_2\delta(\mathbf{W}_1 GAP(\mathbf{F})))$ using the definitions of T_1, T_2, and T_3, respectively. As can be noticed, in the first two variants $T(\cdot)$ is applied directly on $f_\theta(\mathbf{F})$, while in the third case it is applied on pre-sigmoid output to ensure sufficiently wide range for attention scores.

For multimodal fusion, this ignoring-based SE-block is placed between the streams of a multi-modal architecture, aggregating the information from both modalities to a joint squeezed representation and subsequently expanding it into each modality following our ignoring paradigm. Specifically, considering a two-stream scenario, intermediate feature representations from two network branches corresponding to two modalities are first spatially squeezed into channel descriptors by applying global average pooling in each branch. The squeezed representations are subsequently concatenated and projected into a joint lower-dimensional space. The resulting features are transformed with two projection matrices corresponding to each of the two modalities to the spaces of original dimensionalities, and the proposed ignoring approach is then applied to obtain attention masks. The masks are further multiplied element-wise with original feature representations in each branch.

6.1.4 Performance evaluation

Here, we report the results of both multimodal fusion in human action recognition task from RGB and skeleton data based on NTU-RGBD dataset. We initialize the model from ImageNet+Kinectics pretrained weights, finetune for 10 epochs with batch size 8, and report the test set performance of the model that performed best on validation set. To further validate the effectiveness of our learning to ignore framework, we perform additional experiments on a more generic image classification task on Cifar10 and Cifar100 datasets using ResNet50 and DenseNet. Optimization is done with SGD with momentum of 0.9. Each experiment is repeated three times and the average performance is reported. 40k images are used for training and 10k for validation. Standard data augmentation is used. Results of SE and SE via ignoring framework on Cifar10 and Cifar100 can be seen from Table 27. As can be noticed, SE-Ign approaches outperform both vanilla architectures and standard SE in vast majority of cases, with the differences being especially high in CIFAR100 dataset. For example, SE-Ign$_3$ outperforms SE and vanilla ResNet50 by 1.9 and 3.1 percent, respectively. Table 28 shows similar results, where most variants following the ignoring framework outperform the standard approach on NTU-RGBD dataset.

6.2 RGBD Hand-gesture Recognition via Early Fusion

6.2.1 Introduction, objectives and state-of-the-arts

With the advent of new technologies and intelligent systems, industrial automation and robotics have undergone major revolution. There is an increasing demand for automated systems in industrial environments in order to increase production output and reduce production cost. As the need to automate complex tasks increases, there is also increasing need to improve the efficiency in those tasks that inherently require human interaction. In order to enable fluid interaction between human and robots, there must be an efficient way for the human controller to communicate with the robots. There are several communication approaches that have been proposed for human-machine interaction. In the most basic and traditional form of communication, commands are delivered by the human via keyboard inputs. Despite the simplicity of the setup,

		CIFAR 10	CIFAR 100	
		Top-1 Error%	Top-1 Error%	Top-5 Error%
ResNet50	Standard	08.27 ± 0.54	34.06 ± 1.02	10.97 ± 0.54
	SE	07.63 ± 0.37	32.80 ± 0.11	09.97 ± 0.50
	SE-$\text{Ign}_{1(\alpha=1)}$	07.42 ± 0.29	32.50 ± 0.26	09.92 ± 0.37
	SE-$\text{Ign}_{1(\alpha=0.5)}$	07.61 ± 0.46	31.40 ± 0.68	**09.39 ± 0.19**
	SE-$\text{Ign}_{1(\alpha=0.8)}$	07.76 ± 0.73	32.71 ± 1.15	10.07 ± 0.64
	SE-Ign_2	07.66 ± 0.13	32.78 ± 0.77	10.11 ± 0.56
	SE-Ign_3	**07.28 ± 0.17**	**30.95 ± 0.08**	09.49 ± 0.36
DenseNet	Standard	07.07 ± 0.33	29.25 ± 0.10	08.26 ± 0.12
	SE	06.96 ± 0.05	29.43 ± 0.44	08.36 ± 0.33
	SE-$\text{Ign}_{1(\alpha=1)}$	06.94 ± 0.07	29.17 ± 0.07	08.22 ± 0.13
	SE-$\text{Ign}_{1(\alpha=0.5)}$	06.69 ± 0.04	**27.64 ± 0.30**	**07.30 ± 0.10**
	SE-$\text{Ign}_{1(\alpha=0.8)}$	06.95 ± 0.14	27.73 ± 0.41	07.39 ± 0.07
	SE-Ign_2	06.80 ± 0.09	28.08 ± 0.35	07.39 ± 0.23
	SE-Ign_3	**06.41 ± 0.08**	27.77 ± 0.54	07.65 ± 0.20

Table 27: Results of SE variants on CIFAR10 and CIFAR100 datasets.

	MMTM	$\text{Ign}_{1(\alpha=1)}$	$\text{Ign}_{1(\alpha=0.5)}$	$\text{Ign}_{1(\alpha=0.8)}$	Ign_2	Ign_3
NTU-RGBD	89.98	89.99	**90.52**	88.70	90.21	90.36

Table 28: Accuracy on NTU-RGBD dataset

this approach is very rigid and has many limitations in practice. A more flexible communication approach is through voice interaction, which allows faster and more natural communication from the human controller. One drawback in interaction via voice commands is that in many industrial environments, the level of ambient noise is often very high, making it challenging for the speech recognition task.

To program a robot, either Industrial or Collaborative, different programming methods can be adopted. Conventional approaches are subdivided into online and offline programming methods: in the former case, the robot cell is actively used during the programming process, while in the latter the robot cell is not involved in the task definition, thus non-productive robot time is kept at a minimum. Online teaching methods involve walk-through programming, where the operator manually moves the robot along the trajectory to be reproduced, and lead-through programming, where the operator defines the robot program using only the teach pendant, a method that does not require any contact with the robot. Both methods require the operator to have proper knowledge of the robot to be used and a suitable programming experience. Offline programming methods require a higher knowledge of both programming languages and robotics fundamentals and are usually adopted in a simulated environment to build complex robotic cell programs. These methods are extremely time consuming and often stress the operator. Collaborative Robots often adopt a hybrid programming method combining both typologies: the logic of the program can be developed offline in a simulated environment, while the motion instructions and the positioning can be programmed online.

Recently, in the search for a flexible and robust communication method between human and robots, hand gestures have been proposed as a communication medium. This communication medium allows the robot to operate behind a safety cage while communicating with the human controller intuitively and naturally. In the present work, this idea has been applied as well by means of a human-robot communication language based on a Gestures Dictionary, where simple gestures can be combined to form powerful, customized operations. Body language, for example, hand gestures, allows a natural and robust interaction. Hand gesture recognition is a topic that has been vastly explored through the years [166, 112, 111] and a plethora of sensors may be used according to the type of communication methods, for example by using wearable sensors such as haptic gloves or EMG or IMU devices. In later years, vision has become the preferred method to perform a plethora of different tasks, including hand-gesture recognition, robot teleoperation, scene monitoring, and scene reconstruction. This is due to the fact that these sensors are contactless, reliable, and cost-effective, considering that a single camera can be adopted in combination with a suitable Computer Vision or Deep Learning algorithm to achieve satisfactory results depending on the task. Adopting vision sensors to monitor the surroundings of the robot also allows the definition of safety strategies based on human-robot relative positions in parallel with the human-robot communication system.

In this work, we adopted Gesture Dictionary proposed by [112] to construct a vision based hand gesture recognition model, which utilizes both the RGB and depth information to perform the recognition task. The two modalities are combined in a early fusion manner, enabling efficient processing and the ease to use pretrained vision models, i.e., transfer learning.

6.2.2 Description of the work performed so far

Our hand gesture recognition method adopts and modifies the approach proposed in [112]. In [112], the authors proposed a Gesture Dictionary of 16 static hand gestures, which are depicted in Figure 19. Each static gesture can be mapped to a specific command, or they can be combined in sequence to generate more complex commands. Here we should note that although [112] proposes a specific set of machine commands that correspond to these gestures, a user of the OpenDR toolkit can simply utilize the recognition of these hand gestures to re-define the set of commands/meanings they correspond to.

In order to recognize the hand gestures, a color and a depth image of the person depicting the gesture must be captured. The color and depth information are combined into a four-channel image, which is then introduced to the recognition model. This approach is also known as early fusion when processing data from multiple modalities. This is different from the late fusion approach in which different modalities are often processed separately before their deep features are combined in the later stage of the pipeline. One advantage of the early fusion is that transfer learning can be easily utilized. More specifically, since the RGB and the depth map are combined into a single image frame having four channels, any state-of-the-art image recognition architecture can be used for finetuning for this task. This exempts us from the task of designing and validating a suitable architecture for the given task. In addition, there are many state-of-the-art architectures with pretrained models publicly available, thus enabling efficient transfer learning. Since all pretrained models are available only for RGB images, the weight of the first convolution layer is not used.

In [112], the authors proposed to utilize an object detector network, more specifically, the R-FCN object detector [21], in order to recognize the hand gestures. To achieve a computationally efficient solution, we combine transfer learning and simplify the hand gesture recognition model by utilizing pretrained convolutional neural network classifiers that have been trained on the ImageNet dataset. Since the main objective is to recognize the hand gestures, rather than the localization of the hand gestures, the implementation in the toolkit turns to a classification

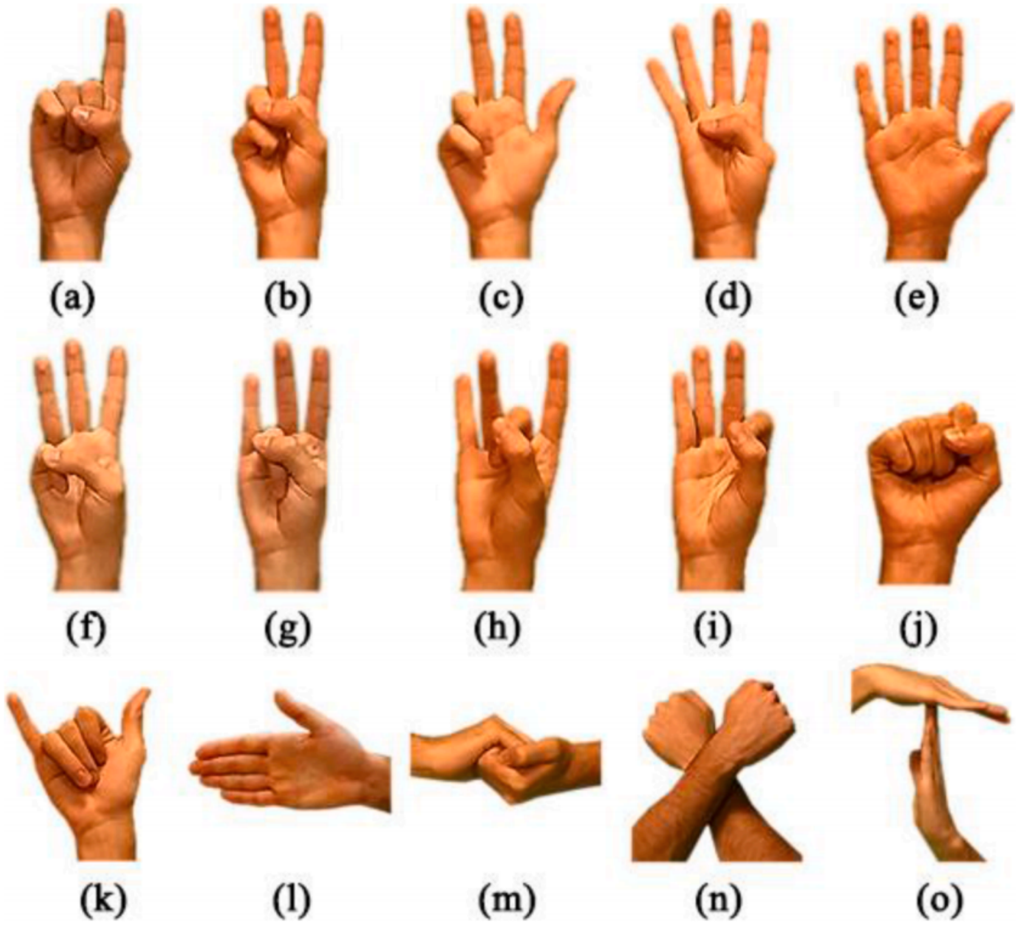


Figure 19: The set of hand gestures proposed in [112]. Gestures from (a) to (i) depict numbers from one to nine. Gesture (j) depicts *punch*, which corresponds to the *confirm* command. Gesture (k) depicts *span*, which corresponds to the *delete* command. Gesture (l) depicts the *left direction* (alternatively, there is also a similar gesture for *right direction*). Gesture (m) is the *collab* gesture, which corresponds to the *open file* command. Gesture (n) depicts an *x sign*, which corresponds to the *exit* command. Gesture (o) is the *time out* gesture, which corresponds to the *pause* command.

approach, which is much more computationally efficient compared to a detection pipeline.

In the toolkit, several convolutional neural network architectures are implemented, such as VGG architectures, ResNet architectures and their variants, DenseNet architectures. In addition, light-weight architectures for mobile and embedded devices such as the MobileNet architecture or the MNasNet architecture are also supported. All of these models are modified to take a four-channel image as input and outputs a class label.

6.2.3 Performance Evaluation

In order to evaluate the performance of the proposed approach, we used the HANDS dataset curated by [111]. The dataset contains both RGB and 2D depth map that were acquired using Kinect V2 sensor. The frames have been calibrated and aligned so that the RGB and the depth map match. The resolution of the images provided in by [111] are 960×540. However, to conduct the experiments, we resized the images to a lower resolution: 224×224. The images were collected from five subjects performing the task, with each gesture captured in 150 frames.

Since one of the objectives of the OpenDR toolkit is to provide computationally efficient models for a given task, we evaluated the pretrained *mobilenet_v2* model, which achieves an accuracy of 85% when evaluating on the HANDS dataset, on different platforms, namely the NVIDIA RTX 2080 Ti (GPU device), AMD Ryzen 3970X (desktop CPU device) and NVIDIA Xavier AGX (embedded device). The time taken (in seconds) to infer a single RGBD sample on different platforms, and the corresponding Frames per Second (FPS), are shown in Table 29.

Table 29: Time taken for the *mobilenet_v2* architecture to make inference on a single RGBD image and the corresponding FPS on different platforms. The results are averaged over 1000 runs.

Platform	**Inference Time (in seconds)**	**FPS**
RTX 2080 Ti	0.0076	131.5
Ryzen 3970X	0.1208	8.2
Xavier AGX	0.0263	38.0

7 Conclusions

AUTH worked towards objective O1a by developing an adaptive inference approach for deep representation learning tasks, such as face recognition (Section 2.1), which allows for meeting the strict speed and latency requirements for many robotics applications. In addition, AUTH developed a DNN-based NMS method (Section 2.3) capable to improve the performance of person detection methods, especially in the case where humans appear in crowded areas. Furthermore, AUTH worked towards evaluating the impact of domain shifts for object detection models trained on well known datasets (Section 2.6), identifying critical limitations and training approaches to mitigate approaches that could increase the robotic autonomy in the field. AUTH also evaluated the impact of using mixed image data for training DNN methods for human centric perception tasks (Section 2.5), demonstrating that they can improve the performance of such methods on real-world settings as well on simulation environments. Also, AUTH worked towards objective O2b by developing an active vision approach for human-centric perception (Section 2.2), exploiting the ability of robots to interact with their environment in order to better sense their surroundings. To this end, a DRL-based control approach was proposed for training agents that are able to identify and focus on task-relevant objects, i.e., humans, as well as issue the appropriate control commands accordingly to acquire better results. Moreover, AUTH in Section 2.4 proposed a novel approach for active face recognition using synthesized facial views, which allows for exploiting photorealistic facial view rendering to discover the best facial view to use for face recognition. Also, AUTH worked towards O1a by developing an deep high-order Gaussian filtering method for time series analysis (Section 5.1), such as biosignals, that can allow for achieving higher analysis accuracy, while using smaller and faster architectures.

TAU worked towards objective O1 by developing an attention-based methodology for Convolutional Neural Networks that can be used both as a standalone module for a variety of computer vision tasks, as well as a multimodal fusion method in multi-stream CNN architectures (Section 6.1). Specifically, the method was evaluated for image classification and human action recognition from RGB+skeleton tasks. In addition, TAU has developed and integrated into the toolkit a hand gesture recognition method that operates on RGB and Depth images (Section 6.2). The developed model exploits early fusion resulting in lightweight architecture with fast inference. TAU has also worked towards development of new attention methods for Neural Bag of Features models for time-series analysis (Section 5.3). Specifically, self-attention based extensions to previously-developed codebook and temporal attention of NBoF were formulated, along with a codebook-temporal self-attention block that allows to jointly learn codebook-temporal representations. The developed methodology has been shown beneficial for the task of biosignal classification. Moreover, TAU worked towards Objective O1 by proposing a new optimization process for training Multilinear Compressive Learning models (Section 2.7), which leads to increased robustness with respect to the size of the compressed signal, and enables practical implementation of adaptive compressive signal acquisition and inference systems. Finally, TAU has improved upon the previously integrated speech command recognition models (Section 5.2) by replacing the standard convolutional layers with more general and adaptive SelfONN layers, achieving better recognition accuracy in some scenarios.

AU worked towards objective O1 by developing two methodologies for efficient continual human activity recognition based on video data (Section 3.1) and skeletal data (Section 3.2), which reduce the per-prediction floating point operations by an order of magnitude. Moreover, AU proposed an efficient methodology for landmark-based facial expression recognition (Section 4.1) which has achieved comparable performance to video-based state-of-the-art methods while capturing the model's uncertainty and having much less computational complexity.

Altogether the approaches and methods developed during the second year of the project are well aligned with the stated project objectives, build upon the work done in the first project year, and include new promising directions. Most of the described methodologies will be implemented and integrated to the OpenDR toolkit, expanding its capabilities. The progress so far gives us confidence to achieve the overall project goals within the targeted time frame and to deliver a capable and practical set of tools for human centric perception and cognition.

References

[1] Face.evoLVe.PyTorch.

[2] Ptb diagnostic ecg database. `https://www.physionet.org/content/ptbdb/1.0.0/`.

[3] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid. Vivit: A video vision transformer. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6836–6846, 2021.

[4] C. Aytekin, A. Iosifidis, and M. Gabbouj. Probabilistic saliency estimation. *Pattern Recognition*, 74:359–372, 2018.

[5] 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 Conf.*, pages 1472–1476, 2018.

[6] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos. Revisiting active perception, 2016.

[7] M. Bastioni, S. Re, and S. Misra. Ideas and methods for modeling 3d human figures: the principal algorithms used by makehuman and their implementation in a new approach to parametric modeling. In *Proceedings of the Bangalore Annual Compute Conference*, pages 1–6, 2008.

[8] J. Behrmann, W. Grathwohl, R. T. Q. Chen, D. Duvenaud, and J.-H. Jacobsen. Invertible residual networks, 2019.

[9] A. Bochkovskiy, C.-Y. Wang, and H. Liao. YOLOv4: Optimal speed and accuracy of object detection. *ArXiv*, abs/2004.10934, 2020.

[10] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis. Soft-NMS: Improving object detection with one line of code. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[11] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee. YOLACT: Real-time instance segmentation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.

[12] 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.

[13] E. J. Candès and M. B. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 25(2):21–30, 2008.

[14] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. VGGFace2: A dataset for recognising faces across pose and age. In *Proc. IEEE Intl. Conf. on Automatic Face & Gesture Recognition*, pages 67–74, 2018.

[15] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733, 2017.

[16] L. Chen, S. Lin, X. Lu, D. Cao, H. Wu, C. Guo, C. Liu, and F.-Y. Wang. Deep neural network-based vehicle and pedestrian detection for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[17] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *Proc. Intl. Conf. on Machine Learning*, pages 1597–1607, 2020.

[18] K. Cheng, Y. Zhang, X. He, W. Chen, J. Cheng, and H. Lu. Skeleton-based action recognition with shift graph convolutional network. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[19] Y. Cheng, D. Wang, P. Zhou, and T. Zhang. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Process. Mag.*, 35(1):126–136, 2018.

[20] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. Conf. on Computer Vision and Pattern Recognition*, 2016.

[21] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. *arXiv preprint arXiv:1605.06409*, 2016.

[22] 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.

[23] J. Deng, J. Guo, N. Xue, and S. Zafeiriou. Arcface: Additive angular margin loss for deep face recognition, 2019.

[24] J. Deng, J. Guo, Y. Zhou, J. Yu, I. Kotsia, and S. Zafeiriou. Retinaface: Single-stage dense face localisation in the wild, 2019.

[25] A. Dhall, R. Goecke, J. Joshi, K. Sikka, and T. Gedeon. Emotion recognition in the wild challenge 2014: Baseline, data and protocol. In *International conference on multimodal interaction*, 2014.

[26] A. Dhall, R. Goecke, S. Lucey, and T. Gedeon. Collecting large, richly annotated facial-expression databases from movies. *IEEE Annals of the History of Computing*, 19(03):34–41, 2012.

[27] 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.

[28] X. Dong, Y. Yan, W. Ouyang, and Y. Yang. Style aggregated network for facial landmark detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[29] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[30] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[31] 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.

[32] Q. Duan and L. Zhang. Look more into occlusion: Realistic face frontalization and recognition with boostgan. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[33] M. Everingham, L. Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88:303–338, 2009.

[34] C. Feichtenhofer. X3D: Expanding architectures for efficient video recognition. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. `https://github.com/facebookresearch/SlowFast`. Apache 2.0 Licence.

[35] C. Feichtenhofer, H. Fan, J. Malik, and K. He. Slowfast networks for video recognition. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. `https://github.com/facebookresearch/SlowFast`. Apache 2.0 Licence.

[36] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[37] J. M. Ferryman and A. Ellis. PETS2010: Dataset and challenge. In *Proceedings of the IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, 2010.

[38] C. Forster, M. Pizzoli, and D. Scaramuzza. Appearance-based active, monocular, dense reconstruction for micro aerial vehicles. 2014.

[39] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proc. Conf. on Computer Vision and Pattern Recognition*, 2016.

[40] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International conference on machine learning*, 2016.

[41] A. Geitgey. Machine learning is fun! part 4: Modern face recognition with deep learning, 2016. [Online; accessed 16-October-2020].

[42] S. Goodfellow, A. Goodwin, D. Eytan, R. Greer, M. Mazwi, and P. Laussen. Towards understanding ecg rhythm classification using convolutional neural networks and attention mappings. 08 2018.

[43] N. Gourier, D. Hall, and J. L. Crowley. Estimating face orientation from robust detection of salient facial structures. In *FG Net workshop on visual observation of deictic gestures*, volume 6, page 7. FGnet (IST–2000–26434) Cambridge, UK, 2004.

[44] J. Gu, Y. Wang, K. Cho, and V. O. Li. Improved zero-shot neural machine translation via ignoring spurious correlations. *arXiv preprint arXiv:1906.01181*, 2019.

[45] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao. MS-CELEB-1M: A dataset and benchmark for large-scale face recognition. In *Proceedings og the European Conf. on Computer Vision*, pages 87–102, 2016.

[46] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *Proc. Intl. Conf. on Learning Representations*, 2016.

[47] 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.

[48] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[49] L. Hedegaard and A. Iosifidis. Continual 3d convolutional neural networks for real-time processing of videos. *preprint, arXiv:2106.00050*, 2021.

[50] 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.

[51] N. Heidari and A. Iosifidis. On the spatial attention in spatio-temporal graph convolutional networks for skeleton-based human action recognition. In *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021.

[52] N. Heidari and A. Iosifidis. Progressive spatio-temporal bilinear network with monte carlo dropout for landmark-based facial expression recognition with uncertainty estimation. *arXiv preprint arXiv:2106.04332*, 2021.

[53] N. Heidari and A. Iosifidis. Progressive spatio-temporal graph convolutional network for skeleton-based human action recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3220–3224, 2021.

[54] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *Proc. NIPS Deep Learning and Representation Learning Workshop*, 2015.

[55] J. Hosang, R. Benenson, and B. Schiele. Learning Non-Maximum Suppression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[56] 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.

[57] H.-K. Hsu, C.-H. Yao, Y.-H. Tsai, W.-C. Hung, H.-Y. Tseng, M. Singh, and M.-H. Yang. Progressive domain adaptation for object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 749–757, 2020.

[58] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei. Relation networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[59] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[60] P. Hu, D. Cai, S. Wang, A. Yao, and Y. Chen. Learning supervised scoring ensemble for emotion recognition in the wild. In *ACM International Conference on Multimodal Interaction*, 2017.

[61] G. B. Huang, V. Jain, and E. Learned-Miller. Unsupervised joint alignment of complex images. In *Proc. Intl. Conf. on Computer Vision*, 2007.

[62] 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.

[63] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7310–7311, 2017.

[64] R. Huang, S. Zhang, T. Li, and R. He. Beyond face rotation: Global and local perception gan for photorealistic and identity preserving frontal view synthesis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2439–2448, 2017.

[65] N. Inoue, R. Furuta, T. Yamasaki, and K. Aizawa. Cross-domain weakly-supervised object detection through progressive domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5001–5009, 2018.

[66] 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.

[67] 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.

[68] E. Johns, S. Leutenegger, and A. J. Davison. Pairwise decomposition of image sequences for active multi-view recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3813–3822, 2016.

[69] H. R. V. Joze, A. Shaban, M. L. Iuzzolino, and K. Koishida. Mmtm: Multimodal transfer module for cnn fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13289–13299, 2020.

[70] 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.

[71] M. Kachuee, S. Fazeli, and M. Sarrafzadeh. Ecg heartbeat classification: A deep transferable representation. *2018 IEEE International Conference on Healthcare Informatics (ICHI)*, Jun 2018.

[72] E. Kakaletsis, E. Symeonidis, M. Tzelepi, I. Mademlis, T. A., N. Nikolaidis, and I. Pitas. Computer vision for autonomous UAV flight safety: An overview and a vision-based safe landing pipeline example. *ACM Computing Surveys*, 2021. accepted.

[73] T. Kanade, J. F. Cohn, and Y. Tian. Comprehensive database for facial expression analysis. In *IEEE International Conference on Automatic Face and Gesture Recognition*, 2000.

[74] H. Kato, Y. Ushiku, and T. Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2018.

[75] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman. The kinetics human action video dataset. *preprint, arXiv:1705.06950*, 2017.

[76] V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[77] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. Intl. for Learning Representations*, 2015.

[78] S. Kiranyaz, T. Ince, A. Iosifidis, and M. Gabbouj. Operational neural networks. *Neural Computing and Applications*, 32(11):6645–6668, 2020.

[79] S. Kiranyaz, J. Malik, H. B. Abdallah, T. Ince, A. Iosifidis, and M. Gabbouj. Self-organized operational neural networks with generative neurons. *Neural Networks*, 140:294–308, 2021.

[80] 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.

[81] O. Köpüklü, S. Hörmann, F. Herzog, H. Cevikalp, and G. Rigoll. Dissected 3d cnns: Temporal skip connections for efficient online video processing. *arXiv preprint arXiv:2009.14639*, 2020.

[82] A. Krizhevsky and G. E. Hinton. Using very deep autoencoders for content-based image retrieval. In *Proc. European Symposium on Artificial Neural Networks*, volume 1, page 2, 2011.

[83] C.-M. Kuo, S.-H. Lai, and M. Sarkis. A compact deep learning model for robust facial expression recognition. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018.

[84] V. V. Kuznetsov, V. A. Moskalenko, and N. Y. Zolotykh. Electrocardiogram generation and feature extraction using a variational autoencoder, 2020.

[85] 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.

[86] F. Laakom, K. Chumachenko, J. Raitoharju, A. Iosifidis, and M. Gabbouj. Learning to ignore: rethinking attention in cnns. *British Machine Vision Conference 2021*, 2021.

[87] F. Laakom, J. Raitoharju, A. Iosifidis, U. Tuna, J. Nikkanen, and M. Gabbouj. Probabilistic color constancy. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 978–982. IEEE, 2020.

[88] M. K. Lee, D. Y. Choi, D. H. Kim, and B. C. Song. Visual scene-aware hybrid neural network architecture for video-based facial expression recognition. In *IEEE International Conference on Automatic Face & Gesture Recognition*, 2019.

[89] J. Liao, A. Kot, T. Guha, and V. Sanchez. Attention selective network for face synthesis and pose-invariant face recognition. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 748–752. IEEE, 2020.

[90] T.-Y. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2014.

[91] S. Liu, D. Huang, and Y. Wang. Adaptive NMS: Refining pedestrian detection in a crowd. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[92] 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.

[93] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. Sphereface: Deep hypersphere embedding for face recognition, 2018.

[94] 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)*, December 2015.

[95] 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.

[96] J.-H. Luo, J. Wu, and W. Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proc. IEEE Intl. Conf. on Computer Vision*, pages 5058–5066, 2017.

[97] I. Mademlis, N. Nikolaidis, and I. Pitas. Stereoscopic video description for key-frame extraction in movie summarization. In *Proceedings of the EURASIP European Signal Processing Conference (EUSIPCO)*. IEEE, 2015.

[98] S. Majumdar and B. Ginsburg. Matchboxnet: 1d time-channel separable convolutional neural network architecture for speech commands recognition. *arXiv preprint arXiv:2004.08531*, 2020.

[99] I. Masi, T. Hassner, A. T. Tran, and G. Medioni. Rapid synthesis of massive face sets for improved face recognition. In *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, pages 604–611. IEEE, 2017.

[100] 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.

[101] O. Michel. Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems*, 1(1):39–42, 2004.

[102] R. Mishra, H. P. Gupta, and T. Dutta. A survey on deep neural network compression: Challenges, overview, and solutions. *arXiv preprint arXiv:2010.03954*, 2020.

[103] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning, 2013.

[104] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[105] 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.

[106] 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.

[107] D. Neimark, O. Bar, M. Zohar, and D. Asselmann. Video transformer network. *arXiv preprint arXiv:2102.00719*, 2021.

[108] Q. T. Ngoc, S. Lee, and B. C. Song. Facial landmark-based emotion recognition via directed graph neural network. *Electronics*, 9(5):764, 2020.

[109] H. V. Nguyen and L. Bai. Cosine similarity metric learning for face verification. In *Proc. Asian Conf. on Computer Vision*, pages 709–720, 2010.

[110] P. Nousi, I. Mademlis, I. Karakostas, A. Tefas, and I. Pitas. Embedded UAV real-time visual object detection and tracking. In *Proceedings of the IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 2019.

[111] C. Nuzzi, S. Pasinetti, R. Pagani, G. Coffetti, and G. Sansoni. Hands: an rgb-d dataset of static hand-gestures for human-robot interaction. *Data in Brief*, 35:106791, 2021.

[112] C. Nuzzi, S. Pasinetti, R. Pagani, S. Ghidini, M. Beschi, G. Coffetti, and G. Sansoni. Meguru: a gesture-based robot program builder for meta-collaborative workstations. *Robotics and Computer-Integrated Manufacturing*, 68:102085, 2021.

[113] D. Osokin. Real-time 2d multi-person pose estimation on cpu: Lightweight openpose. In *arXiv preprint arXiv:1811.12004*, 2018.

[114] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. S. A. Ku, and D. Tran. Image transformer. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[115] N. Passalis, J. Raitoharju, A. Tefas, and M. Gabbouj. Adaptive inference using hierarchical convolutional bag-of-features for low-power embedded platforms. In *Proc. IEEE Intl. Conf. on Image Processing*, pages 3048–3052, 2019.

[116] N. Passalis, J. Raitoharju, A. Tefas, and M. Gabbouj. Efficient adaptive inference for deep convolutional neural networks using hierarchical early exits. *Pattern Recognition*, 105:107346, 2020.

[117] N. Passalis and A. Tefas. Learning bag-of-features pooling for deep convolutional neural networks. In *Proc. IEEE Intl. Conf. on Computer Vision*, pages 5766–5774, 2017.

[118] N. Passalis and A. Tefas. Learning deep representations with probabilistic knowledge transfer. In *Proc. European Conf. on Computer Vision*, pages 268–284, 2018.

[119] N. Passalis and A. Tefas. Leveraging active perception for improving embedding-based deep face recognition. In *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE, 2020.

[120] N. Passalis and A. Tefas. Adaptive inference for face recognition leveraging deep metric learning-enabled early exits. In *Proc. of the European Signal Processing Conference*, 2021.

[121] N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis. Temporal logistic neural bag-of-features for financial time series forecasting leveraging limit order book data. *arXiv preprint 1901.08280 (2019).*

[122] N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis. Deep adaptive input normalization for time series forecasting. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9):3760–3765, 2019.

[123] 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.

[124] B. Pyakillya, N. Kazachenko, and N. Mikhailovsky. Deep learning for ecg classification. In *Journal of Physics: Conference Series*, volume 913, page 012004, 2017.

[125] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[126] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[127] 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, 2017.

[128] D. Ruan, Y. Yan, S. Lai, Z. Chai, C. Shen, and H. Wang. Feature decomposition and reconstruction learning for effective facial expression recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[129] A. A. Rusu, M. Večerík, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell. Sim-to-real robot learning from pixels with progressive nets. In *Proceedings of the Conference on Robot Learning*, pages 262–270, 2017.

[130] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[131] 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.

[132] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017.

[133] S. Sengupta, J. Cheng, C. Castillo, V. Patel, R. Chellappa, and D. Jacobs. Frontal to profile face verification in the wild. In *IEEE Conf. on Applications of Computer Vision*, 2016.

[134] 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*, June 2016.

[135] S. Shao, Z. Zhao, B. Li, T. Xiao, G. Yu, X. Zhang, and J. Sun. Crowdhuman: A benchmark for detecting human in a crowd, 2018.

[136] 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.

[137] 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.

[138] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[139] G. Singh and F. Cuzzolin. Recurrent convolutions for causal 3d cnns. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 1456–1465, 2019.

[140] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Proc. Intl. Conf. on Neural Information Processing Systems*, pages 1857–1865, 2016.

[141] 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. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2021.

[142] L. Su, C. Hu, G. Li, and D. Cao. Msaf: Multimodal split attention fusion. *arXiv preprint arXiv:2012.07175*, 2020.

[143] C. Symeonidis, E. Kakaletsis, I. Mademlis, N. Nikolaidis, A. Tefas, and I. Pitas. Vision-based UAV safe landing exploiting lightweight deep neural networks. In *Proceedings of the International Conference on Image and Graphics Processing (ICIGP)*, 2021.

[144] C. Symeonidis, I. Mademlis, N. Nikolaidis, and I. Pitas. Improving neural Non-Maximum Suppression for object detection by exploiting interest-point detectors. In *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2019.

[145] C. Symeonidis, I. Mademlis, I. Pitas, and N. Nikolaidis. Neural Attention-driven Non-Maximum Suppression for Person Detection. 11 2021.

[146] C. Symeonidis, P. Nousi, P. Tosidis, K. Tsampazis, N. Passalis, A. Tefas, and N. Nikolaidis. Efficient realistic data generation framework leveraging deep learning-based human digitization. In *Proceedings of the 22nd Engineering Applications of Neural Networks Conference*, pages 271–283. Springer International Publishing, 2021.

[147] M. Tan and Q. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of Machine Learning Research*, volume 97, pages 6105–6114, 2019.

[148] S. Tang, M. Andriluka, A. Milan, K. Schindler, S. Roth, and B. Schiele. Learning people detectors for tracking in crowded scenes. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1049–1056, 2013.

[149] S. Teerapittayanon, B. McDanel, and H.-T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *Proc. Intl. Conf. on Pattern Recognition*, pages 2464–2469, 2016.

[150] 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.

[151] 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.

[152] H. Tu, G. Duoji, Q. Zhao, and S. Wu. Improved single sample per person face recognition via enriching intra-variation and invariant features. *Applied Sciences*, 10(2):601, 2020.

[153] A. Tzimas, N. Passalis, and A. Tefas. Leveraging deep reinforcement learning for active shooting under open-world setting. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 1–6, 2020.

[154] J. van Amersfoort, L. Smith, A. Jesson, O. Key, and Y. Gal. Improving deterministic uncertainty estimation in deep learning for classification and regression. abs/2102.11409, 2021.

[155] J. I. Vasquez-Gomez, D. Troncoso, I. Becerra, E. Sucar, and R. Murrieta-Cid. Next-best-view regression using a 3d convolutional neural network. *Machine Vision and Applications*, 32(2):1–14, 2021.

[156] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, 2017.

[157] V. Vielzeuf, S. Pateux, and F. Jurie. Temporal multimodal fusion for video emotion classification in the wild. In *ACM International Conference on Multimodal Interaction*, 2017.

[158] 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.

[159] J. Wang and W. Li. Atrial fibrillation detection and ecg classification based on cnn-bilstm, 2020.

[160] Q. Wang, J. Gao, W. Lin, and Y. Yuan. Learning from synthetic data for crowd counting in the wild. In *Proc. Conf. on Computer Vision and Pattern Recognition*, 2019.

[161] Y. E. Wang, G.-Y. Wei, and D. Brooks. Benchmarking tpu, gpu, and cpu platforms for deep learning. *arXiv preprint arXiv:1907.10701*, 2019.

[162] P. Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

[163] Webots. http://www.cyberbotics.com. Open-source Mobile Robot Simulation Software.

[164] X. Wei, S. Liu, Y. Xiang, Z. Duan, C. Zhao, and Y. Lu. Incremental learning based multi-domain adaptation for object detection. *Knowledge-Based Systems*, 210:106420, 2020.

[165] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[166] Z. Xia, Q. Lei, Y. Yang, H. Zhang, Y. He, W. Wang, and M. Huang. Vision-based hand gesture recognition for human-robot collaboration: A survey. In *2019 5th International Conference on Control, Automation and Robotics (ICCAR)*, pages 198–205, 2019.

[167] X. Xu and H. Liu. Ecg heartbeat classification using convolutional neural networks. *IEEE Access*, 8:8614–8619, 2020.

[168] S. Yan, Y. Xiong, and D. Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI Conference on Artificial Intelligence*, 2018.

[169] Z. Yu, Q. Liu, and G. Liu. Deeper cascaded peak-piloted network for weak expression recognition. *The Visual Computer*, 34(12):1691–1699, 2018.

[170] J. Zhang, Y. Yan, and M. Lades. Face recognition: eigenface, elastic matching, and neural nets. *Proceedings of the IEEE*, 85(9):1423–1435, 1997.

[171] K. Zhang, Y. Huang, Y. Du, and L. Wang. Facial expression recognition based on deep evolutional spatial-temporal networks. *IEEE Transactions on Image Processing*, 26(9):4193–4203, 2017.

[172] P. Zhang, C. Lan, W. Zeng, J. Xing, J. Xue, and N. Zheng. Semantics-guided neural networks for efficient skeleton-based human action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[173] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018.

[174] G. Zhao, X. Huang, M. Taini, S. Z. Li, and M. PietikäInen. Facial expression recognition from near-infrared videos. *Image and Vision Computing*, 29(9):607–619, 2011.

[175] X. Zhao, X. He, and P. Xie. Learning by ignoring, with application to domain adaptation. *arXiv preprint arXiv:2012.14288*, 2020.

[176] X. Zhao, X. Liang, L. Liu, T. Li, Y. Han, N. Vasconcelos, and S. Yan. Peak-piloted deep network for facial expression recognition. In *European Conference on Computer Vision*, 2016.

[177] T. Zheng and W. Deng. Cross-pose lfw: A database for studying cross-pose face recognition in unconstrained environments. *Beijing University of Posts and Telecommunications, Tech. Rep*, 5, 2018.

[178] T. Zheng, W. Deng, and J. Hu. Cross-age lfw: A database for studying cross-age face recognition in unconstrained environments. *arXiv preprint arXiv:1708.08197*, 2017.

[179] 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.

[180] L. Zhu, L. Sevilla-Lara, Y. Yang, M. Feiszli, and H. Wang. Faster recurrent networks for efficient video classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:13098–13105, 2020.

8 Appendix

8.1 Adaptive Inference for Face Recognition leveraging Deep Metric Learning-enabled Early Exits

The appended paper follows.

Adaptive Inference for Face Recognition leveraging Deep Metric Learning-enabled Early Exits

Nikolaos Passalis
Dept. of Informatics
Aristotle University of Thessaloniki, Thessaloniki, Greece
passalis@csd.auth.gr

Anastasios Tefas
Dept. of Informatics
Aristotle University of Thessaloniki, Thessaloniki, Greece
tefas@csd.auth.gr

Abstract—**Deep Learning (DL) models that support adaptive computational graphs allow for easily adapting the computations to the available resources by selecting the most appropriate computational path. However, such models are typically used in classification settings, e.g., using early exits, despite that DL models often aim at extracting representations, e.g., for face recognition. In this work, we provide a metric-learning oriented early exit methodology for DL models. As we demonstrate, employing early exits in metric learning scenarios pose unique challenges compared to existing methodologies for classification-oriented early exits. To this end, we employ the Bag-of-Features model to efficiently extract compact representations from any layer of a DL model that is then combined with an efficient linear regressor to match the final representation of the model (without having to feedforward the whole computational graph). The proposed method is agile and can be directly used with any pre-trained DL model, while it is end-to-end differentiable, allowing for further fine-tuning the models towards having multiple early exits. The effectiveness of the proposed method is demonstrated using five face verification/recognition datasets.**

Index Terms—**Adaptive Inference, Early Exits, Deep Metric Learning, Lightweight Deep Learning**

I. Introduction

A number of impressive applications, ranging from accurate robotics perception [1] to precise disease prognosis and diagnosis [2], are enabled by powerful Deep Learning (DL) models [3]. Indeed, DL models are becoming increasingly powerful, following the continuous improvements in dedicated hardware accelerators, such as Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs) [4], which allowed for training and deploying deeper and more complicated models. However, in many applications, such as robotics [1] and Internet-of-Things (IoT) [5], we are often still limited to using less powerful hardware, due to a number of limitations, ranging from energy and power constraints to constrained physical form factors. As a result, numerous methods have been proposed to allow for developing more *lightweight* DL models that will be deployed in such devices, while meeting critical application-specific requirements, such as low latency and real-time operation.

These methods include quantization [6], for reducing the number of bits spent for each of the parameters of the model, pruning methods [7], that discard parts of the model that are not critical for its operation, models that are lightweight by design [8], [9], as well as knowledge distillation approaches [10], [11], which aim to transfer the knolwedge from a larger and more complex neural network into a smaller and faster one. These approaches led to more lightweight models that could operate faster in many embedded and mobile devices. However, most of these methods are not capable of adapting to varying computational loads. In other words, the inference time is constant regardless the environmental conditions, e.g., the difficulty of each sample, the load of the system, etc.

This is a critical limiting factor in a number of embedded applications, where the load dynamically varies according to the environmental conditions. For example, for a face recognition application the time needed for face recognition depends on the number of faces that appear in a given frame. As a result, even through a model might operate in real time for a specific number of faces, e.g., 2-3 faces, this might not be the case when a larger number of people appears in a given frame. Therefore, in such cases, we need models that can effectively adapt to the current conditions, providing faster (and possibly less accurate) predictions when the load is higher in order to satisfy the processing time limitations of a given application. In this way, the models can provide accurate answers, exploiting all the available processing time, while still meeting the requirements of each application when the load is higher.

These limitations can be addressed by using models that support *adaptive computational graphs*, such as [12]–[14]. These approaches work by altering the number of computations in order to keep the load within certain limits. This is usually achieved by using multiple paths over the computation graph of the model. Among the most straightforward ways to achieve this is by using *early exits* [12]–[14]. By placing such early *classification* layers at various intermediate layers of the network we can early stop the computation whenever it is deemed appropriate (e.g., when the computational budget is spent or when the network is already confident enough regarding the provided prediction), obtaining an estimation for the representation that would be extracted from the final output layer of the network.

Even though early exits provided a very powerful tool to address the aforementioned limitations, its use is currently limited to classification settings [12]–[14]. However, in many cases, deep learning models aim at extracting representations (metric learning) [15], instead of directly predicting the class to which the input sample belongs to. Perhaps the most well known example of such metric learning task is face recognition [16] and content-based information retrieval [17]. To the best of our knowledge there has been no attempt to use early exits in such scenarios, such as metric learning-based face verification. Even though it could be argued that using early exits in such scenarios could be deemed redundant, since we can directly extract a representation from any layer of a DL model without any modification, we demonstrate that this naive approach has significant limitations and that we can achieve higher accuracy by employing appropriately designed and trained early exit layers.

The main contribution of this work is to provide a metric learning-oriented early exit methodology for DL models. As we experimentally demonstrate, employing early exits in metric learning scenarios pose unique challenges compared to existing methodologies for classification-oriented early exits. To this end, in this work we leverage the Bag-of-Features model to efficiently extract compact representations from any layer of a neural network. Then, an additional small linear regressor is used to regress the final output of the model at selected points of its computational graph. In this way, a representation that can be used in place of the final representation can be readily extracted from an early exit. This provides significant advantages over existing metric learning approaches, which would require keeping a separate database for the representations extracted from each exit layer, increasing the space required for using any additional layer as an early exit and reducing the accuracy of the resulting models, as we demonstrate in Section III. The proposed method is agile and can be directly used with any pre-trained metric learning DL model, while it is end-to-end differentiable, allowing for further fine-tuning the models towards having multiple early exits. The effectiveness of the proposed method is demonstrated using five face verification/recognition datasets, including DL models trained on the large-scale MS-Celeb-1M dataset [18] and evaluated using a wide range of datasets, as well as experiments conducted on two embedded platforms typically used in robotics applications.

The rest of the paper is structured as follows. The proposed method is introduced in Section II. Then, Section III provides the experimental setup and experimental evaluation, while Section IV concludes this paper.

II. Proposed Method

In this Section we present the proposed method. First, we introduce the used notation and then we present the proposed early exit strategy. Let $f_{\mathbf{W}}(\mathbf{x}, i)$ denote the response of the i-th layer of a neural network that is composed of a total of m layers. We used the notation $\mathbf{W}$ to refer to the trainable parameters of the network, while $\mathbf{x}$ denotes the input to the network. In this work we focus on convolutional neural networks that handle images as input, i.e., $\mathbf{x} \in \mathbb{R}^{W \times H \times C}$, where W is the width, H is the height, and C is the number of channels of the image. However, this is without loss of generality, since the proposed method can be directly employed for any other type of DL model. To simplify the used notation, we denote by $\mathbf{y}^{(i)} = f_{\mathbf{W}}(\mathbf{x}, i) \in \mathbb{R}^{W_i \times H_i \times C_i}$ the output of the i-th layer, where W_i, H_i and C_i refer to the width, height and number of channels of the extracted feature map. The notation $\mathbf{y} = f_{\mathbf{W}}(\mathbf{x}, m) \in \mathbb{R}^M$ is used to refer the final output of the network, where M is the dimensionality of the output layer. Finally, we use the notation $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ to refer to a training set of N images that will be used for training the early exits.

For the rest of this Section, we assume that the network has already been trained to perform a specific metric learning task [16], [19], and we will focus on training the early exits on top of the representations $\mathbf{y}^{(i)}$ extracted at specific points of its computational graph. Early exits typically employ an additional estimator, fitted on top of the representation extracted at various points of the computation graph of the model, to predict the final output of the model. Therefore, we can use an estimator $g^{(i)}_{\mathbf{W}_i}(\cdot)$ at the i-th layer of the network as:

$$g^{(i)}_{\mathbf{W}_i}\left(\mathbf{y}^{(i)}\right) = g^{(i)}_{\mathbf{W}_i}(f_{\mathbf{W}}(\mathbf{x}, i)) \in \mathbb{R}^M. \tag{1}$$

The notation $\mathbf{W}_i$ is used to refer to the parameters each early exit. Classification-based early exits are trained to directly solve the original classification task of the network [12]–[14]. However, for metric-learning oriented network this approach cannot be employed, since even if we use the original loss used for training the network, e.g., the contrastive loss [19], we will not learn representations in the same space as the one formed by the last layer of the network. As a result, the representations extracted by the early exits would not be useful for performing queries in a database that consists of representations extracted from the final layer of the network.

To overcome this limitation, in this work we proposed using a distillation inspired approach [10], i.e., to train the early exits in order to mimic the output of each layer. Perhaps the most straightforward approach to ensure that the features extracted by the early exits $g^{(i)}_{\mathbf{W}_i}(\cdot)$ will reside in the same space as the final representation of the network $\mathbf{y}$ is to minimize the quadratic divergence between these two representations. Therefore, early exit estimators are trained in order to minimize the following loss:

$$\mathcal{L}_i = \frac{1}{N} \sum_{j=1}^{N} ||\mathbf{y}_j - g^{(i)}_{\mathbf{W}_i}(\mathbf{y}^{(i)}_j)||^2_2, \tag{2}$$

where i denotes the early exit that we are training, $|| \cdot ||_2$ denotes the l_2 norm of a vector and the notation $\mathbf{y}_j$ is used to refer to the representation extracted when the j-th sample is fed into the network, i.e., $\mathbf{y}_j = f_{\mathbf{W}}(\mathbf{x}_j, m) \in \mathbb{R}^{N_C}$.

Usually, early exits employ a feature aggregation approach to reduce the dimensionality of the extracted feature maps, e.g., Global Average Pooling, that is then followed by a fully connected layer. However, naive feature aggregation approaches, such as global average/max pooling, have been shown to discard useful information [20]. Therefore, in this work we employ a Bag-of-Features (BoF)-based aggregation layer in order to reduce the dimensionality of the extracted feature maps and extract a compact summary representation that can be further adapted towards the task at hand [21].

BoF-based pooling works as follows. First, we quantize each feature vector extracted from a feature map using a set of N_K codewords. In this work, we use the notation $\mathbf{v}_{ij}$ to denote each codeword, where i is the layer to which the codeword belongs to (a separate codebook is used for each layer) and j refers to a specific codeword (out of N_K possible ones). In this way, we can then extract a membership vector for each feature vector that belongs to the i-th exit as:

$$[\mathbf{u}_{ikl}]_j = \frac{K([\mathbf{y}^{(i)}]_{kl}, \mathbf{v}_{ij})}{\sum_{m=1}^{N_K} K([\mathbf{y}^{(i)}]_{kl}, \mathbf{v}_{im})} \in [0,1]. \quad (3)$$

The notation (k,l) is used to refer to the location of the feature map from which the feature vector is extracted, while $K(\cdot)$ denotes the kernel used for measuring the similarity between codewords and feature vectors. In this work, we use a Gaussian-based kernel to this end:

$$K(\mathbf{x}, \mathbf{v}_{ij}) = \exp(-\frac{||\mathbf{x} - \mathbf{v}_{ij}||^2}{2\sigma_i^2}), \quad (4)$$

where σ_i is a trainable scaling factor that scales the distances between feature vectors and codewords to the appropriate range. For non-trainable BoF models, σ_i is typically set to the average distance between the feature vectors and the codewords.

After extracting the membership vectors $\mathbf{u}_{ikl}$ we can directly extract a compact histogram representation for each exit as:

$$\mathbf{s}^{(i)} = \frac{1}{W_i H_i} \sum_{k=1}^{W_i} \sum_{l=1}^{H_i} \mathbf{u}_{ikl} \in \mathbb{R}^{N_K}, \quad (5)$$

This histogram representation provides a summary of the concepts that appear in the corresponding features. By appropriately tuning the codewords we can *focus* the representation on different concepts. For example, using *k-means* to learn the codebook leads to a generic representation that can be used for any task, while finetuning the whole layer using gradient descent allows for learning task-specific codewords (provided that the BoF layer is part of a network trained for a specific task). Finally, this histogram representation is fed into a linear layer that projects the histogram into the desired space, i.e., $g^{(i)}_{\mathbf{W}_i}(\mathbf{y}^{(i)}) = s^{(i)}(\mathbf{y}^{(i)})\mathbf{W}_i^l$, where $\mathbf{W}_i^l \in \mathbb{R}^{N_K \times M}$. In the case of metric-learning networks this would be the space formed by the output layer of the network. Then, early exits can be trivially trained using gradient descent, minimizing the loss provided in (2).

III. Experimental Evaluation

The proposed method was evaluated using the MS-Celeb-1M [18], Labeled Faces in the Wild (LFW) [22], [23], Cross Pose LFW (CPLFW) [24], Cross Age LFW (CALFW) [25], and VGGFace2 [26] datasets. More specifically, we follow a standard face verification setup [27], where the models are trained on the MS-Celeb-1M dataset and evaluated on the remaining four datasets, i.e., LFW, CPLFW, CALFW and VGGFace2. All images used for the conduncted experiments were resized to 112×112 pixels. For the evaluation procedure we randomly sample 6,000 image pairs that either correspond to face images of the same person or to face images of different persons (equally distributed among the two cases). A face pair is considered to belong to the same person when the distance between the corresponding embeddings is lower than a certain threshold. This threshold is selected to maximize the face verification accuracy on a validation dataset. As a result, we report the average 10-fold cross validation accuracy for all the conducted experiments, i.e., the threshold is selected according to the validation split and the accuracy is reported on the corresponding test set. For all the conducted experiments we used a ResNet-50 network [28], where inverted residual blocks were employed for improving its efficiency [29]. The early exits were placed after the 1st, 2nd and 3nd residual block. The dimensionality of the feature vectors extracted from each of these blocks is 128, 256 and 256 respectively, while the dimensionality of the final representation of the network is 512.

TABLE I
Number of additional parameters required for each method

Method	Exit 1	Exit 2	Exit 3
Raw*	1.2-1.4M	2.4-2.7M	2.4-2.7M
LR	66k	131k	131k
BoF	327k	393k	393k
Proposed	327k	393k	393k

*For the raw method we assume that we employ a database that contains 3,000 feature vectors, which corresponds to the evaluation setup used in this paper.

Three different methods were evaluated along with the proposed one. For the first one we directly extracted the feature vectors from each early exit, we employed global average pooling and then we queried the database using these representations. This approach is called "*raw*" in the rest of this paper, since it relies on directly using the raw feature vectors, as they are extracted from the network. Even though the dimensionality of these feature vectors is lower, this method requires keeping a separate database with the feature vectors extracted from each additional early exit, significantly increasing the storage requirements as shown in Table I. Next, we evaluated a linear regressor (denoted by "LR") that was trained to directly regress the output representation of the network based on the (average) pooled representation extracted from each early exit. The same approach was also repeated using the Bag-of-Feature model, where we used 512 codewords for building the codebook (using the k-means algorithms and the feature vectors extracted from each early exit for building the codebook). This method is denoted as "BoF". Note that both the LR and BoF methods can be regarded as a simplified (ablated) version of the proposed one, since, to the best of our knowledge, neither has been proposed in the literature for constructing early exits. Despite this, they consist a strong baseline, as we demonstrate later in this Section.

Finally, we evaluated the proposed method using again $N_K = 512$ codewords (in order to be directly comparable with the BoF baseline). The number of training iterations was set to 2,000 with a learning rate of 0.001 for the parameters of the BoF model, while the linear regressor was fine-tuned using a learning rate of 0.0001. The Adam algorithm with its default parameters was used for the optimization [30], while the batch size was set to 32. After training the BoF-based layers, the linear regressor was fitted again using the closed form solution to ensure a fair comparison with the LR method (16,000 training samples were used for fitting the regressor). A comparison between the number of parameters required for adding the early exits to the base model are summarized in Table I. All the methods (LR, BoF, Proposed) significantly reduce the number of required parameters over the naive raw baseline. Using the BoF layer increases, to a small extent, the number of required parameters, but as we demonstrate later, this is also accompanied by a corresponding increase in the face verification accuracy. Furthermore, in all cases (except from the raw baseline) the number of parameters is kept within reasonable limits. It is also worth noting that for the BoF/Proposed method the number of the added parameters can be further controlled by decreasing the number of codewords N_K.

The experimental evaluation is provided in Table II. The four evaluated methods are compared on four different datasets using the three different added early exits. In all the cases, using a subsequent early exit increases the obtained verification accuracy as expected. Furthermore, just using a linear regressor (LR) to regress the final representation of the network leads to a significant increase over directly using the raw representation. Indeed, in some cases (e.g., CALFW) the accuracy increases by over 25%. Then, using the BoF model further increases the performance, while employing the proposed method leads to the overall best accuracy in all the evaluated cases. It is worth noting that in some cases, the verification performance is very close to the actual performance of the final output of the network, as reported in Table III.

To further verify that using the proposed method leads to actual performance improvements we evaluated all the employed methods using two embedded platforms, the NVIDIA Jetson TX-2 and the NVIDIA Jetson AGX. The obtained results are summarized in Table IV. Indeed, using early exits leads to a significant speedup , e.g., about $4\times$ for the first exit compared to the final output of the network. Using the proposed method leads to a slight overhead (about 10%) compared to the raw and LR methods. However, it still leads to enormous performance improvements over the final output of the network, while achieving higher verification accuracy, as demonstrated before.

TABLE II
FACE VERIFICATION ACCURACY ON FOUR DIFFERENT DATASETS USING THREE DIFFERENT EARLY EXITS. THE MEAN AND STANDARD DEVIATION OF THE 10-FOLD CROSS-VALIDATION ACCURACY IS REPORTED.

Method	Exit 1	Exit 2	Exit 3
Dataset: LFW			
Raw	68.60 ± 2.25	82.33 ± 1.25	92.80 ± 1.57
LR	75.05 ± 2.00	88.98 ± 1.42	94.15 ± 0.69
BoF	79.93 ± 1.59	92.33 ± 1.22	92.58 ± 1.23
Proposed	$\mathbf{80.98 \pm 2.64}$	$\mathbf{92.70 \pm 1.28}$	$\mathbf{96.58 \pm 0.64}$
Dataset: CPLFW			
Raw	52.75 ± 1.95	51.98 ± 1.74	63.83 ± 2.38
LR	66.80 ± 1.71	75.57 ± 2.04	83.02 ± 1.36
BoF	68.52 ± 1.79	79.98 ± 2.16	81.13 ± 1.05
Proposed	$\mathbf{68.98 \pm 1.18}$	$\mathbf{80.05 \pm 1.70}$	$\mathbf{84.20 \pm 1.56}$
Dataset: CALFW			
Raw	55.00 ± 1.62	61.72 ± 1.50	68.10 ± 1.98
LR	70.28 ± 1.51	82.18 ± 1.89	87.35 ± 1.18
BoF	71.80 ± 1.18	84.55 ± 1.06	84.93 ± 1.51
Proposed	$\mathbf{73.65 \pm 1.82}$	$\mathbf{84.78 \pm 1.50}$	$\mathbf{89.37 \pm 1.37}$
Dataset: VGGFace2			
Raw	56.88 ± 2.07	64.24 ± 1.12	78.88 ± 1.43
LR	67.10 ± 1.74	78.50 ± 1.85	84.16 ± 1.33
BoF	68.86 ± 1.84	81.56 ± 2.69	83.50 ± 1.55
Proposed	$\mathbf{71.34 \pm 2.04}$	$\mathbf{82.32 \pm 2.32}$	$\mathbf{88.10 \pm 1.43}$

TABLE III
FACE VERIFICATION ACCURACY ON FOUR DIFFERENT DATASETS USING THE FINAL OUTPUT OF THE EMPLOYED NETWORK.

Dataset	Accuracy
LFW	99.78 ± 0.22
CPLFW	92.05 ± 1.36
CALFW	95.78 ± 1.20
VGGFace2	94.98 ± 0.71

The mean and standard deviation of the 10-fold cross-validation accuracy is reported.

IV. CONCLUSIONS

In this work we proposed a metric learning-oriented early exit methodology that can be effectively used with any DL model. To this end, we employed the Bag-of-Features model in order to extract compact representations from enormous feature maps, that were then fed into lightweight linear regressors trained to approximate the final representaiton of the model. In this way, it is possible to estimate the final output of a large and complex DL model at various points of its computational graph. As we experimentally demonstrated, this approach allows for effectively adapting the computations to the available resources. At the same time, the proposed method works significantly better than naive approaches that were used until now, such as directly using the representation extracted from a layer and building separate databases for each layer. Furthermore, the proposed method is easy to use and agile, since it can be readily combined with any DL model. At the same time, it can be used to train the models in an end-to-end fashion, in order to better adapt to the task at hand, since it

TABLE IV
SPEED (FPS) COMPARISON BETWEEN DIFFERENT METHODS AND EXITS

Method	Exit 1	Exit 2	Exit 3
Jetson TX-2 ($\approx$ 0.8 FP32 TFLOPS)			
Default		5.6	
Raw	23.4	13.2	9.6
LR	23.1	13.2	9.6
BoF/Proposed	20.8	12.9	9.5
Jetson AGX ($\approx$ 1.4 FP32 TFLOPS)			
Default		8.6	
Raw	41.2	24.5	18.7
LR	40.8	24.3	18.7
BoF/Proposed	38.1	23.9	18.4

Frames Per Second (FPS) are reported. FPS are measured using batches of 4 input images, to ensure more consistent measurements. The average of 100 runs is reported.

relies on a fully differentiable formulation.

In this way, the proposed method paves the way for building more advanced early exit methodologies for representation learning tasks. For example, the early exits at subsequent layers can be trained to regress the residual error [20], instead of the raw final representation. We expect that this will allow for further increasing the performance of the method, since each exit could be finetuned to just refine the previous output. Furthermore, the models can be also used to adapt the model to the difficulty of each sample, in a similar fashion as in [31], allowing for skipping layers of the networks for easier samples. This could allow for further increasing the inference speed and reducing the energy requirements of the models.

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] N. Sünderhauf, O. Brock, W. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford *et al.*, "The limits and potentials of deep learning for robotics," *The Intl. Journal of Robotics Research*, vol. 37, no. 4-5, pp. 405–420, 2018.

[2] B. Gecer, S. Aksoy, E. Mercan, L. G. Shapiro, D. L. Weaver, and J. 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.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[4] Y. E. Wang, G.-Y. Wei, and D. Brooks, "Benchmarking tpu, gpu, and cpu platforms for deep learning," *arXiv preprint arXiv:1907.10701*, 2019.

[5] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for iot big data and streaming analytics: A survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2923–2960, 2018.

[6] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *Proc. Intl. Conf. on Learning Representations*, 2016.

[7] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Intl. Conf. on Computer Vision*, 2017, pp. 5058–5066.

[8] 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.

[9] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2018, pp. 6848–6856.

[10] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[11] N. Passalis and A. Tefas, "Learning deep representations with probabilistic knowledge transfer," in *Proc. European Conf. on Computer Vision*, 2018, pp. 268–284.

[12] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *Proc. Intl. Conf. on Pattern Recognition*, 2016, pp. 2464–2469.

[13] N. Passalis, J. Raitoharju, A. Tefas, and M. Gabbouj, "Adaptive inference using hierarchical convolutional bag-of-features for low-power embedded platforms," in *Proc. IEEE Intl. Conf. on Image Processing*, 2019, pp. 3048–3052.

[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 Conf.*, 2018, pp. 1472–1476.

[15] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *Proc. Intl. Conf. on Neural Information Processing Systems*, 2016, pp. 1857–1865.

[16] H. V. Nguyen and L. Bai, "Cosine similarity metric learning for face verification," in *Proc. Asian Conf. on Computer Vision*, 2010, pp. 709–720.

[17] A. Krizhevsky and G. E. Hinton, "Using very deep autoencoders for content-based image retrieval." in *Proc. European Symposium on Artificial Neural Networks*, vol. 1, 2011, p. 2.

[18] "MS-Celeb-1M."

[19] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. Intl. Conf. on Machine Learning*, 2020, pp. 1597–1607.

[20] N. Passalis, J. Raitoharju, A. Tefas, and M. Gabbouj, "Efficient adaptive inference for deep convolutional neural networks using hierarchical early exits," *Pattern Recognition*, vol. 105, p. 107346, 2020.

[21] N. Passalis and A. Tefas, "Learning bag-of-features pooling for deep convolutional neural networks," in *Proc. IEEE Intl. Conf. on Computer Vision*, 2017, pp. 5766–5774.

[22] 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," University of Massachusetts, Amherst, Tech. Rep. 07-49, October 2007.

[23] G. B. Huang, V. Jain, and E. Learned-Miller, "Unsupervised joint alignment of complex images," in *Proc. Intl. Conf. on Computer Vision*, 2007.

[24] T. Zheng and W. Deng, "Cross-pose lfw: A database for studying cross-pose face recognition in unconstrained environments," *Beijing University of Posts and Telecommunications, Tech. Rep*, vol. 5, 2018.

[25] T. Zheng, W. Deng, and J. Hu, "Cross-age lfw: A database for studying cross-age face recognition in unconstrained environments," *arXiv preprint arXiv:1708.08197*, 2017.

[26] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "VGGFace2: A dataset for recognising faces across pose and age," in *Proc. IEEE Intl. Conf. on Automatic Face & Gesture Recognition*, 2018, pp. 67–74.

[27] Face.evoLVe.PyTorch. [Online]. Available: https://github.com/ZhaoJ9014/face.evoLVe.PyTorch

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[29] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[31] N. Passalis, J. Raitoharju, M. Gabbouj, and A. Tefas, "Efficient adaptive inference leveraging bag-of-features-based early exits," in *Proc. Intl. Workshop on Multimedia Signal Processing*, 2020, pp. 1–6.

8.2 Neural attention-driven Non-Maximum Suppression for person detection

The appended paper follows.

Neural Attention-driven Non-Maximum Suppression for Person Detection

Neural Attention-driven Non-Maximum Suppression for Person Detection

Charalampos Symeonidis, Ioannis Mademlis, Ioannis Pitas and Nikos Nikolaidis
Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece

***Abstract*—Non-maximum suppression (NMS) is a post-processing step in almost every visual object detector. NMS aims to prune the number of overlapping detected candidate regions-of-interest (ROIs) on an image, in order to assign a single and spatially accurate detection to each object. The default NMS algorithm (GreedyNMS) is fairly simple and suffers from severe drawbacks, due to its need for manual tuning. A typical case of failure with high application relevance is pedestrian/person detection in dense human crowds, where GreedyNMS doesn't provide accurate results. This paper proposes an efficient deep neural architecture for NMS in the person detection scenario, by capturing relations of neighbouring ROIs and aiming to ideally assign precisely one detection per person. The presented Seq2Seq-NMS architecture assumes a sequence-to-sequence formulation of the NMS problem, exploits the Multihead Scale-Dot Product Attention mechanism and jointly processes both geometric and visual properties of the input candidate ROIs. Thorough experimental evaluation on three public person detection datasets shows favourable results against competing methods, with acceptable inference runtime requirements and good behaviour for large numbers of raw candidate ROIs per image.**

***Index Terms*—Non-Maximum Suppression, Object Detection, Scaled-Dot Product Attention, Sequence-to-Sequence Learning, Person Detection**

I. Introduction

Non-Maximum Suppression (NMS) is a final refinement step incorporated to almost every visual object detection framework, assigned the duty of merging/filtering any spatially overlapping detected Regions-of-Interest (ROIs), i.e., bounding boxes, which correspond to the same visible object on an image. The problem it attempts to solve arises from the tendency of many detectors to output multiple, neighbouring candidate object ROIs for a single visible object, due to their implicit sliding-window nature. Thus, an NMS algorithm processes the raw object detector outputs identified on an input image and attempts to filter out the duplicate ROIs.

The de facto dominant NMS method for object detection is GreedyNMS. It selects high-scoring detections and deletes less confident neighbours, since they most likely cover the same object. Its simplicity, speed and unexpectedly good behaviour in most cases make it competitive against proposed alternatives, since rapid execution is very important for NMS. An Intersection-over-Union (IoU) threshold determines which less-confident neighbors are suppressed by a detection. This fixed IoU threshold leads GreedyNMS to failure in certain cases. Too powerful a suppression, using a low threshold, may remove detections that cover different spatially overlapped objects, while a too high threshold may be unable to suppress duplicate detections.

Due to these limitations of traditional algorithms, modern Deep Neural Network (DNN)-based methods for performing NMS have emerged during the past few years. While some DNNs are assigned with auxiliary tasks complementing the original NMS scheme (e.g., estimate target density maps in order to apply dynamic suppression thresholding [1]), others provide a more straightforward solution (e.g., outputting a score for each candidate detection, thus indicating whether it corresponds to a "duplicate" detection or not [2]). The latter type of methods relies on building representations for each candidate detection, typically based on their corresponding geometric/spatial relations [2], while ignoring ROI visual appearance. This is either because CNN-based features can blur the boundaries between highly overlapping true positives and duplicates, or due to the difficulties DNNs are faced with when trying to extract accurate representations for highly occluded objects. However, evidence has recently surfaced indicating that appearance-based input may improve the performance of DNN-based NMS methods [3] [4], if that information is properly fused with the geometry-based input.

An additional issue stems from the fact that the NMS problem for object detection purposes is essentially sequential in nature. The output ROIs are sequentially processed by the common object detection evaluation protocols [5] [6], ordered according to the scalar confidence scores assigned to them by the NMS method. Similarly, the input candidate ROIs, i.e., the raw output of the object detector which is fed as input to the NMS algorithm, must also be ordered according to the initial confidence scores assigned to them by the detector. Thus, essentially, an NMS method actually decides whether a candidate ROI is duplicate, or not, based on the decisions it has previously taken for the preceding, higher-scoring candidate ROIs along the input sequence. However, to the best of our knowledge, NMS has not been previously explicitly formulated as a problem of processing sequences, thus related algorithms have not been applied to solving it.

Motivated by these issues of existing neural NMS approaches, this paper offers the following contributions:

- a novel reformulation of the NMS task for object detection as a sequence-to-sequence problem.
- a novel deep neural architecture for NMS, relying on the Scaled Dot-Product Attention mechanism, called *Seq2Seq-NMS*.
- a new, fast, efficient and GPU-based neural implementation of the low-level Frame Moments Descriptor (FMoD) [7], which is employed for feeding the proposed DNN

with appearance-based representations of detected candidate ROIs.

The proposed method is highly applicable to the person/pedestrian detection task, where most NMS algorithms face difficulties in identifying individuals within a crowd, due to various levels of occlusions. The majority of existing NMS methods are oriented towards fast execution, but person detection in human crowds requires a high degree of accuracy; this is critical for ensuring human safety in domains such as autonomous systems [8] [9] [10] [11] [12] [13] [14] [15]. Moreover, the visual appearance representation approach adopted by Seq2Seq-NMS, i.e., FMoD descriptors computed on edge maps of cropped candidate ROIs, is most accurate in cases where the visible silhouette of the target object class remains approximately identical in shape across the training and test images. This is true in the person detection case, bar abnormally extensive viewpoint variations across the employed dataset. Adopting FMoD, which has already proven its worth in NMS for person detection from aerial viewpoints [3], renders the applicability of the proposed method focused to similar scenarios.

Extensive quantitative evaluation using well-known metrics and public person detection datasets indicates favourable results in comparison to several competing NMS methods, both neural and non-neural, leading to state-of-the-art results.

II. Related Work

NMS is the final step of typical object detection pipelines, thus this Section first briefly reviews state-of-the-art detectors. Subsequently, NMS algorithms and related loss functions are presented. Finally, the motivation behind the proposed method is discussed in the context of the existing approaches to NMS.

A. Object Detection

Object detection is a long-standing, fundamental problem in computer vision. Its task is to generate bounding boxes (in 2D pixel coordinates) for objects detected on an image that belong to prespecified object classes and to assign classification scores to them. Most of the early object detection algorithms [16] [17] relied mainly on local handcrafted descriptors and discriminative classifiers. The Deformable Part-based Model (DPM) [18] is a special case, where an object is represented by its component parts arranged in a deformable configuration. In [19], the authors designed a joint person detector, based on the DPM architecture, which overcomes the limitations imposed by frequent occlusions in real-world street scenes.

Object detection has been tremendously improved thanks to Deep Neural Networks (DNNs), with Convolutional Neural Networks (CNNs) being the most relevant architectures. DNN-based object detectors are usually grouped into two categories: two-stage and one-stage object detectors. Typically, the former ones (e.g., [20] [21]) first create object proposals from input images, using a method such as selective search or a separate DNN, and then extract features from these proposals using CNNs. These features are then fed to a classifier that determines the existence and the class of any object in each proposal. Although two-stage detectors achieve state-of-the-art performance, their running speed is typically slow. One-stage object detectors, such as [22] [23] [24] and [25] perform region proposal and object classification in a single, unified DNN. Initial regions are predefined bounding boxes with various scales and ratios placed densely on the image, which are generally referenced as anchors. From the initial anchors, the detectors find those that likely contain objects. Compared to two-stage detectors, their one-stage competitors are usually much faster, but less accurate.

B. Non-Maximum Suppression

The de facto standard in NMS for object detection is GreedyNMS [26]. It selects high-scoring detections and deletes less confident neighbours, since they most likely cover the same object. An Intersection-over-Union (IOU) threshold determines which less-confident neighboring detections are suppressed. It is a simple, well-known, but limited method, leading to several attempts for replacing it with much improved alternatives.

In Soft-NMS [27], a rescoring function decreases the score of neighboring less-confident detections, instead of completely eliminating them, achieving better precision and recall rates compared to GreedyNMS. The authors experiment with Gaussian and linear weighting functions, which both require a hyper-parameter tuning similar to GreedyNMS. In [28], the final coordinates of a detection are being reformulated as the weighted-average of the coordinates of all neighboring detections, given an IoU threshold. GossipNet [2] is a DNN designed to perform NMS, by processing the coordinates and scores of the detections. Overall, it jointly analyzes all detections in the image, so as not to directly prune them, but to rescore them. In [29], the authors replace the classification scores of candidate detections, used in GreedyNMS, with learned localization confidences to guide NMS towards preserving more accurately localized bounding boxes. In [4], an attention module is applied with the task to exploit relations between the input detections, in order to classify them as duplicate or not. [1] proposes Adaptive-NMS, a dynamic thresholding version of GreedyNMS. A relatively shallow neural network predicts a density map and sets adaptive IoU thresholds in NMS for different detections according to the predicted density. An accelerated NMS method has been proposed in [30], allowing higher inference times in exchange for a small performance drop, due to the large number of boxes that are likely to be over-suppressed.

GossipNet was modified in [3], for the specific case of person detection from aerial views, so as to jointly process visual appearance and geometric properties of candidate ROIs. The method exploited handcrafted descriptors encoding statistical ROI appearance characteristics, which were computed on the spatial distribution of edges or interest-points detected within each ROI. These distributions acted as a discriminant factor for identifying complete vs partial object silhouettes, since the silhouette of any person seen from an aerial view is rather similar in shape.

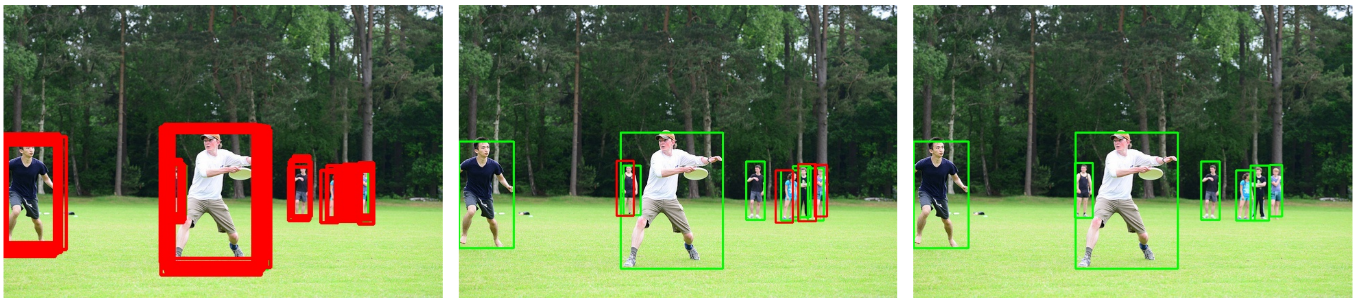

(a) Raw ROIs/detections.
(b) ROIs/detections after applying GreedyNMS at 0.5 IOU.
(c) ROIs/detections after applying the proposed Seq2Seq-NMS method.

Fig. 1: Candidate ROIs/detections from Faster-RCNN in an image from the COCO dataset. Detections matched successfully to humans are colored green, while "incorrect" detections are colored red.

More recently, [31] proposed *Distance-IoU* (DIoU), a new metric which can replace the typical IoU metric in GreedyNMS. This work suggested that the suppression procedure should take into account not only the overlap of two neighboring detections, but also the distances between their centers. Alternatively, Cluster-NMS was proposed in [32], i.e., a technique where NMS is performed by implicitly clustering candidate detections. Cluster-NMS can incorporate geometric factors to improve both precision and recall rates and can efficiently run on a GPU, achieving very fast inference runtimes.

C. Loss Functions for Bounding Box Regression

In DNN-based methods for visual object detection, prediction of spatially accurate ROIs/bounding boxes is enforced by an additional loss term during model training. The regressed ROI parameters are position, shape and scale, in terms of 2D pixel coordinates. These parameters are predicted either directly, or as offsets relative to "anchor boxes", in the case of anchor-based detectors. It is common to use the $\mathcal{L}_n$-norm for calculating the corresponding loss term (e.g., [21] [22] [24]). However, [33] indicates that the correlation between training with such $\mathcal{L}_n$-norm loss terms and improving test accuracy, as measured by the Intersection-over-Union (IoU) metric, is not strong at all. On the other hand, directly incorporating the IoU metric in a loss function would implicitly force the detector itself to also perform a rudimentary degree of NMS, but this is unsuitable for cases where two bounding boxes are non-overlapping, due to their zero loss gradient. Thus, [33] proposes the *Generalized-IoU* (GIoU) loss term, which handles similar scenarios but suffers from slow convergence and inaccurate regression. Thus, in [31], a loss term relying on the DIoU metric was formulated, by adding to the IoU loss a penalty based on the 2D center point coordinates of two bounding boxes. This was shown to converge faster than GIoU. [31] also proposed the *Complete-IoU* (CIoU) loss, an extension of DIoU with an additional term which can be tuned so as to impose aspect ratio consistency between two bounding boxes, thus leading to further increases in test accuracy.

D. Motivation

State-of-the-art object detectors continue to require NMS as a final step [25], even when they use sophisticated loss functions for bounding box regression during training. A typical scenario showcasing the indispensability of a reliable NMS method is when object detection is performed to images depicting dense human crowds, with high levels of occlusions [1] [34]; ironically, this constitutes a challenge even to state-of-the-art NMS algorithms.

Although the geometric properties of candidate ROIs have been considerably exploited by various NMS approaches [2] [32] [31] [30], only a couple of methods [1] [4] [3] have attempted to take advantage of both visual appearance and geometric/spatial ROI information. Therefore, joint exploitation of appearance and geometry for NMS in object detection is underexplored. In addition, despite a vast amount of effort expended towards achieving short inference times [30] [31], since fast execution is an important aspect of NMS, one can easily identify real-world scenarios where a potential improvement in accuracy may equally matter (e.g., pedestrian/person detection in human safety-centric applications).

Despite the sequential nature of the NMS task in object detection, since at least the input candidate ROIs are always ordered according to their confidence score, no previous method has relied on formulating the problem as a sequence-to-sequence task. Thus, the recent rise of self-attention neural modules [35], capable of efficiently capturing interrelations within a sequence, has not yet significantly affected NMS algorithms. To the best of our knowledge, the only relevant method employing self-attention mechanisms is [4], tailored for the duplicate removal task and not for pure NMS. Thus, it does not perform free rescoring: an input candidate ROI which was assigned a low confidence score by the detector (e.g., due to occlusion) cannot be rescored higher by the duplicate removal DNN; only lower. An unconstrained NMS method exploiting the powerful self-attention neural mechanism has yet to emerge.

Based on the above considerations, this paper presents: a) a reformulation of the NMS task as a sequence-to-sequence problem, and b) a novel DNN architecture for solving it, called *Seq2Seq-NMS*. Out of the existing literature, the proposed method is most related to [2] [3] and [4]. Like GossipNet in [2], Seq2Seq-NMS approaches NMS as a rescoring problem. However, an optimized geometric representation for each candidate ROI is proposed here, slightly similar, but different and enriched compared to the GossipNet input descriptor. Like [3], Seq2Seq-NMS jointly processes visual and geometric representations of the input candidate ROIs, using the FMoD descriptor [7] computed on edge maps of cropped detections. However, in this paper, the FMoD descriptor has been re-implemented neurally, leading to significant runtime gains thanks to GP-GPU-based parallel processing, while a novel deep neural architecture is proposed here, so as to exploit the sequence-to-sequence formulation, instead of relying on GossipNet. Finally, similarly to [4], the Seq2Seq-NMS architecture employs the powerful self-attention neural mechanism, but since the proposed method is a complete, free rescoring NMS DNN it is able to search for and fully exploit interrelations between the candidate ROI representations, without being constrained by the original confidence score assigned by the object detector.

III. Attention-driven Non-Maximum Suppression

In this paper, NMS for object detection is first reformulated as a sequence-to-sequence task. This approach is highly related to the evaluation criteria established in object detection [5] [6], where the candidate ROIs identified on an input image are assumed to indirectly form a sequence, based on the scalar confidence score assigned to each of them by the detector (in descending order). Traditionally, evaluating a detector's accuracy on a known dataset involves an analysis of this sequence. At each step, a candidate ROI is processed and matched to a ground-truth object, if and only if: (a) their IoU is higher than a predefined threshold, and (b) that ground-truth object hasn't been previously matched to a higher-scoring candidate detection. In the case where both (a) and (b) are fulfilled, the candidate ROI is marked as "correct", otherwise it is marked as "false". In the special case where only (a) is fulfilled, the candidate detection is marked as "false", due to it being a "duplicate" detection. Thus, the position of a candidate ROI in the sequence can be a significant factor when taking the decision to classify it as a "duplicate" or not.

This emphasis in the ordering is shared with problems traditionally viewed as sequence-to-sequence ones. For instance, in machine translation, a sequence of words from one language must be transformed into a sequence of words in another language. The order of each word (*token*) in the sentence is crucial and can modify its meaning (*context*). Similarly, in object detection evaluation, although a candidate ROI (token) can be successfully matched to a ground-truth object, it can be classified as "duplicate" and therefore as "false", instead of being classified as "correct", due to the fact that a higher-scoring candidate detection, which has been positioned earlier in the sequence, has already been matched with the same ground-truth object.

Motivated by these notions, this paper explicitly formulates the NMS task as a mapping from an input sequence of candidate ROIs to a corresponding output sequence with identical length. Let $\mathbf{R}^{in}$ be the input sequence of candidate ROIs, in descending order based on their scalar confidence scores assigned by the detector:

$$\mathbf{R}^{in} = [\mathbf{r}_1^{in}, ..., \mathbf{r}_N^{in} | r_i^{score_{det}} \geq r_{i+1}^{score_{det}}] \tag{1}$$

where $\mathbf{r}_i^{in} = [r_i^{x_{min}}, r_i^{y_{min}}, r_i^{x_{max}}, r_i^{y_{max}}, r_i^{score_{det}}]$ is an input candidate ROI expressed through its spatial 2D image coordinates, along with its corresponding score assigned by the detector, and N is the number of candidate detections. Let $\mathbf{R}^{out}$ be the output sequence of candidate ROIs, in descending order based on the scores assigned by the NMS method:

$$\mathbf{R}^{out} = [\mathbf{r}_1^{out}, ..., \mathbf{r}_N^{out} | r_i^{score_{NMS}} \geq r_{i+1}^{score_{NMS}}] \tag{2}$$

where $\mathbf{r}_i^{out} = [r_i^{x_{min}}, r_i^{y_{min}}, r_i^{x_{max}}, r_i^{y_{max}}, r_i^{score_{NMS}}]$ is an NMS-rescored candidate ROI. The proposed formulation of the NMS task can be expressed as:

$$\mathbf{R}^{out} = NMS(\mathbf{R}^{in}) \tag{3}$$

Building upon this novel view of the NMS task, the method proposed in this paper, which we call *Seq2Seq-NMS*, receives as input a sequence of candidate ROIs, generated by an object detector, and extracts rich representations regarding their appearance and geometry. Subsequently, these representations are fed to a DNN which processes them in parallel, while mainly paying attention to spatially neighboring, higher-scoring candidates when analyzing each ROI. Finally, it outputs a sequence of scalar scores, each one defining the context of a candidate detection. This is essentially information that determines the final decision of whether the respective ROI should be classified as "correct" or as "potentially suppressed", after the NMS task has been completed. In the proposed formulation, the context of the i^{th} candidate detection is expressed through the corresponding output score, which is a classification probability $p_i : \{p_i \in \mathbb{R} | 0 \leq p_i \leq 1\}$ (1/0 means "correct"/"potentially suppressed", respectively). After the inference stage, simple thresholding can be applied on these output probabilities/scores, in order to decide which candidate detections should be retained. This formulation avoids hard discarding/pruning of ROIs at the inference phase itself, thus allowing us to find a balance in the trade-off between False Positive Rate (FPR) and True Negative Rate (TNR), depending on the application (e.g., using a low threshold in human safety-centric applications such as pedestrian detection).

Seq2Seq-NMS relies on building rich representations for each candidate detection, based on their visual appearance, their geometry and their interrelations, in order to solve the NMS task. Abstractly, it consists of the following three steps:

- Appearance-based ROI representations extraction.
- Geometry-based ROI representations extraction.
- Detections rescoring through the attention-driven NMS DNN.

These steps are detailed below.

A. Appearance-based ROI Representations Extraction

This step can be considered optional, since ROI representations that have been already computed at the intermediate feature extraction layers of the DNN-based object detector itself can be used instead. However, the use of ROI representations computed solely for the NMS procedure makes the NMS DNN less detector-specific and more robust against variations in the effectiveness and the performance of the deployed detector. In [3], where the goal was person detection from aerial views, representations consisting of statistical ROI appearance properties were used, which were computed on the spatial distribution of edges or interest-points detected within each ROI. These distributions acted as a discriminant factor for identifying complete vs partial object silhouettes, since the silhouette of any person seen from an aerial view is rather similar in shape. However, the same argument can be made for people seen from a ground perspective (e.g., pedestrians perceived by an autonomous car), therefore this is a solution applicable to most person detection scenarios.

In [3], a CPU implementation of the low-level FMoD visual descriptor was employed for representing each candidate ROI. FMoD was originally devised in a global [7] and in a local [36] variant (LMoD), respectively applied to movie [37] and activity video [38] [39] [40] summarization via key-frame extraction. Typically, FMoD and LMoD capture informative image statistics from various available image channels (e.g., luminance, color/hue, optical flow magnitude, edge map, and/or stereoscopic disparity), both in a global and in various local scales, under a spatial pyramid video frame partitioning scheme. In both [3] and in this paper, only the luminance channel is employed for the special use-case of describing a ROI interest-point map instead of a typical image/video frame. The intent is to compactly capture the spatial distribution of the interest-points within the ROI interest-point map in a single numerical description vector. However, in [3] ROIs were processed sequentially and not simultaneously, thus achieving very long inference times, nowhere close to real-time. To tackle this limitation in this paper, FMoD was re-implemented neurally so that it can run in parallel on modern GPUs and, given as input a set of candidate ROIs of different shape and scale, simultaneously extract several feature maps before computing in parallel their FMoD descriptors/representations. This fast neural implementation was rendered feasible thanks to the recently proposed operations [20] and [41].

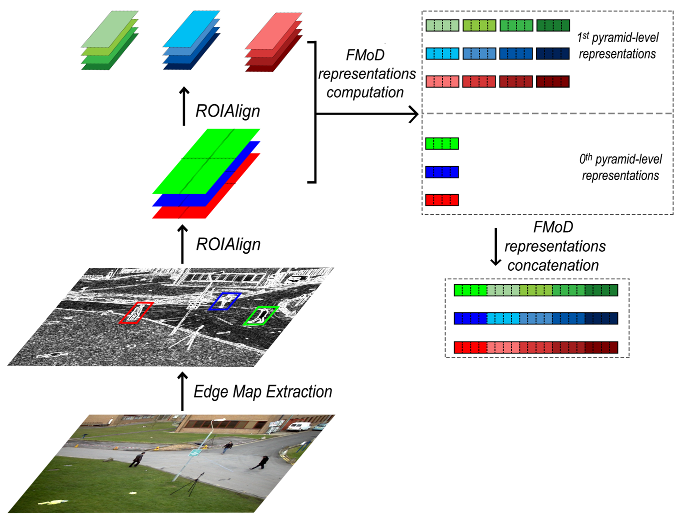
The appearance-based ROI representations extraction process can be divided into three separate operations. The first one involves the computation of edges/interest-point maps of the input RGB image, which is a relatively fast and efficient process. The second step is using the *ROIAlign* [41] operator to extract, in parallel, fixed-size regions across one or multiple maps. Finally, deriving the FMoD representations of these fixed-size maps involves in-parallel computation of the following 15 scalar statistical attributes:

- (1-3) horizontal/vertical/vectorized-block mean values.
- (4-6) horizontal/vertical/vectorized-block standard deviation values.
- (7-9) horizontal/vertical/vectorized-block skew values.
- (10-12) horizontal/vertical/vectorized-block kurtosis values.
- (13-15) horizontal/vertical/vectorized-block signal power values.

Algorithm 1: Appearance-based ROI representations extraction using FMoD

```
Input: (a) an RGB image $\mathbf{I}$
       (b) a set of $N$ ROIs expressed in 2D pixel
       coordinates $\mathbf{B} = [\mathbf{b}_0, \mathbf{b}_1, .., \mathbf{b}_N] \in \mathbb{R}^{N\times 4}$
       (c) FMoD pyramid levels $L$, $L \geq 1$
Output: Appearance-based representations
       $\mathbf{A} \in \mathbb{R}^{N\times 5(4^L-1)}$
1  begin
2      Resize image $\mathbf{I}$ to a fixed size of $W_f \times H_f$.
3      $E(\mathbf{I}) \leftarrow$ Compute the edge/interest-point map of
        image $\mathbf{I}$.
4      Extract in parallel the $0^{th}$-level ROI maps
        $\mathbf{M}^0 = [\mathbf{M}_0^0, \mathbf{M}_1^0, .., \mathbf{M}_N^0]$, where $\mathbf{M}_i^0 \in \mathbb{R}^{1\times W_0\times H_0}$,
        through the ROIAlign operator on $E(\mathbf{I})$.
5      Compute in parallel the $0^{th}$-level FMoD
        representations $\mathbf{A}^0 = [\mathbf{A}_0^0, \mathbf{A}_1^0, .., \mathbf{A}_N^0]$ of $\mathbf{M}^0$,
        where $\mathbf{A}_i^0 \in \mathbb{R}^{15}$.
6      for $j \leftarrow 1$ to $(L-1)$ do
7          Extract in parallel the $j^{th}$-level ROI maps
            $\mathbf{M}^j = [\mathbf{M}_0^j, \mathbf{M}_1^j, .., \mathbf{M}_N^j]$, where
            $\mathbf{M}_i^j \in \mathbb{R}^{4^j\times \frac{W_0}{2^j}\times \frac{H_0}{2^j}}$, through subdivision of
            $\mathbf{M}^0$ ROI maps into four quadrants for $j$ times,
            using the ROIAlign operator.
8          Compute in parallel the $j^{th}$-level FMoD
            representations $\mathbf{A}^j = [\mathbf{A}_0^j, \mathbf{A}_1^j, .., \mathbf{A}_N^j]$ of $\mathbf{M}^j$,
            where $\mathbf{A}_i^j \in \mathbb{R}^{15\times 4^j}$.
9      end
10     Concatenate FMoD representations across all
        pyramid levels $\mathbf{A} \in \mathbb{R}^{N\times 5(4^L-1)}$, where
        $\mathbf{A}_i = [\mathbf{A}_i^0, .., \mathbf{A}_i^L]$.
11 end
```

The corresponding procedure is described on Algorithm 1. Initially, the RGB input image $\mathbf{I}$, of an arbitrary resolution, is resized to a fixed resolution of $W_f \times H_f$ and its corresponding edge/interest-point map $E(\mathbf{I})$ is computed. To make actual inference times even shorter, this operation is carried out here in parallel with the corresponding detector's inference phase. Similarly to [3], the FMoD representations of all ROIs are computed under a spatial pyramid partitioning scheme [42]. At the pyramid base, the 0^{th}-level ROI maps $\mathbf{M}^0 = [\mathbf{M}_0^0, \mathbf{M}_1^0, .., \mathbf{M}_N^0]$, $\mathbf{M}_i^0 \in \mathbb{R}^{1\times W_0\times H_0}$ are extracted in parallel by applying the ROIAlign operator on $E(\mathbf{I})$, assuming that N candidate ROIs (expressed as rectangles in 2D pixel coordinates) have been identified by the object detector for



1st pyramid-level representations
FMoD representations computation
ROIAlign
0th pyramid-level representations
ROIAlign
FMoD representations concatenation
Edge Map Extraction

Fig. 2: Computation of the visual appearance-based candidate ROI representations, by applying the fast FMoD implementation to an image with 3 ROIs and using 2 pyramid levels.

input $\mathbf{I}$. Using $\mathbf{M}^0$, the 0^{th}-level FMoD representations $\mathbf{A}^0 = [\mathbf{A}_0^0, \mathbf{A}_1^0, .., \mathbf{A}_N^0]$, $\mathbf{A}_i^0 \in \mathbb{R}^{15}$ are computed in parallel. Subsequently, the representations at the remaining spatial pyramid levels are computed iteratively, by the in-parallel computation first of $\mathbf{M}^j$ and then of the corresponding partial FMoD descriptors $\mathbf{A}^j$. Once the latter ones have been computed for all L pyramid levels, where L is predefined and fixed, they are concatenated along them. For example, in an image with $N = 5$ candidate ROIs and $L = 2$ pyramid levels, $\mathbf{A} \in \mathbb{R}^{5\times 75}$. This example is illustrated in Figure 2.

B. Geometry-based ROI Representations Extraction

The spatial/geometric interrelations between the various candidate ROIs detected on an input image, based only on their 2D pixel coordinates and not on their visual appearance, is crucial for solving the NMS problem. Such a set of purely geometric attributes has previously proven effective as an input descriptor, in the context of the GossipNet neural architecture [2]. Thus, in this paper, a slightly similar, but enriched set of attributes has been devised, serving as an additional representation for each candidate ROI.

Given a set of N candidate ROIs expressed in 2D pixel coordinates, along with their corresponding detection scores, the tensor $\mathbf{G} \in \mathbb{R}^{N\times N\times 14}$ is computed, where each entry $\mathbf{G}^{ij} \in \mathbb{R}^{14}$ contains the following attributes:

- (1-3) the normalized horizontal/vertical/euclidean distances[1] defined between the centers of the j^{th} and the i^{th} ROI.
- (4-7) the normalized width/height/area/aspect-ratio of the j^{th} ROI.
- (8-11) the width/height/area/aspect-ratio differences (e.g., $\frac{w_j}{w_i}$) between the j^{th} and the i^{th} ROI.
- (12) the detector's confidence score for the j^{th} ROI.
- (13) the detector's confidence score differences between the j^{th} (e.g., $s_j - s_i$) and the i^{th} ROI.
- (14) the IoU between the j^{th} and the i^{th} ROI.

Therefore, each diagonal entry $\mathbf{G}^{ii} \in \mathbb{R}^{14}$ contains the geometric representation of the i-th input candidate ROI/detection.

C. Detetions rescoring through the attention-driven NMS DNN

The goal of the proposed DNN architecture is to perform one-class Non-Maximum Suppression on a set of candidate ROIs/detections through rescoring them, instead of directly pruning them. For a given set of N candidate ROIs detected on an input image, the DNN receives as input a sequence of corresponding representations ($\mathbf{A}$ and $\mathbf{G}$, encoding the appearance and the geometry of all ROIs in the sequence), sorted in a descending order based on the respective scalar detection confidence score.

During inference, these two types of information are fused and each candidate ROI refines its representation by attending to the representations of all detections in the provided set. The Scaled Dot-Product Attention mechanism [35], originally proposed for machine translation tasks, is employed to this end, since it has been proven effective in various applications, such as image classification [43], or image generation [44]. It is briefly described in the next Subsection. In the context of the proposed DNN, the candidate detections used as keys are represented in a relative-to-each-query manner within this attention mechanism. Although this choice leads to slightly increased computational and memory costs, it allows the DNN to more effectively capture the interrelations between the candidate detections.

Finally, the model predicts a new scalar score for each ROI, indicating whether it should be suppressed or not. The output sequence is formed by sorting the candidate ROIs, based on their new scores in descending order.

Multihead Self-Attention Module: The Scaled Dot-Product Attention, also known as self-attention, was presented in [35] and formulated as follows:

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V}, \tag{4}$$

where $\mathbf{Q} \in \mathbb{R}^{N_q\times d_k}$ are the queries, $\mathbf{K} \in \mathbb{R}^{N_k\times d_k}$ are the keys and $\mathbf{V} \in \mathbb{R}^{N_k\times d_v}$ are the values. Each query and each key has a dimension of d_k, while each value has a dimension of d_v. Multihead Attention was also proposed in [35], as a module which allows various attention mechanisms, including self-attention, to run in parallel. This module can be formulated as:

$$Multihead(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\mathbf{h}_1, ..., \mathbf{h}_H]\mathbf{W}^O, \tag{5}$$

where

$$\mathbf{h}_i = Attention(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V). \tag{6}$$

[1]Horizontal and vertical distances are signed distances.

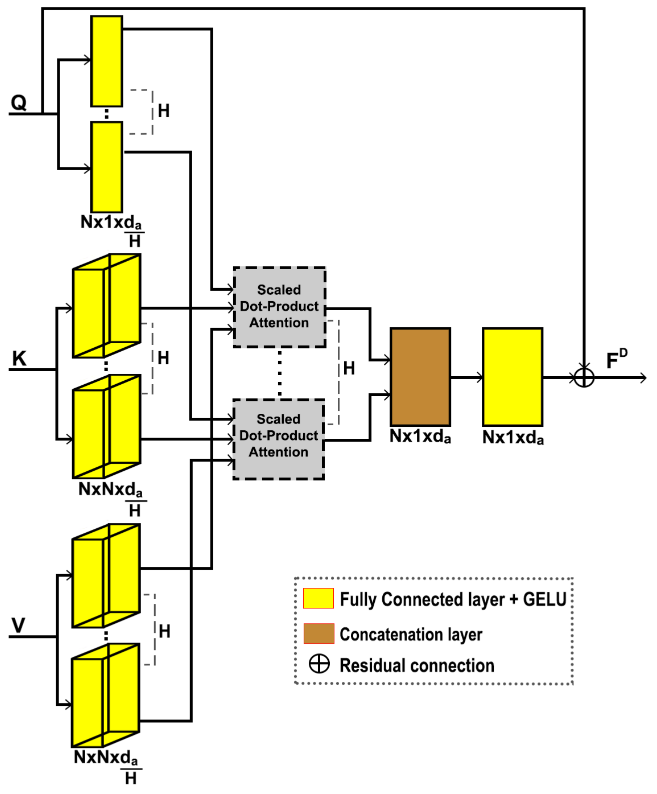

Q
K
V
H
H
H
H
Nx1xda/H
NxNxda/H
NxNxda/H
Scaled Dot-Product Attention
Scaled Dot-Product Attention
Nx1xda
Nx1xda
F^D
Fully Connected layer + GELU
Concatenation layer
Residual connection

Fig. 3: Illustration of the Multihead Self-Attention Module.

In this formulation, $\mathbf{W}_i^Q \in \mathbb{R}^{d_a \times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_a \times d_k}$, $\mathbf{W}_i^V \in \mathbb{R}^{d_a \times d_v}$, $\mathbf{W}_i^O \in \mathbb{R}^{H d_v \times d_a}$ are projection parameter matrices, H is the number of heads, $d_k = d_v = \frac{d_a}{H}$, and the operator [...] implies concatenation.

The proposed DNN architecture relies on these mechanisms in order to identify relations between candidate detections, based both on their visual appearance and on their geometric properties. Such relations can help the model in determining whether a candidate detection should be suppressed or not. For example, the DNN can decide that a higher-scoring candidate ROI should possibly suppress other less-scoring detections having similar appearance and geometric representations.

In [35] the authors introduced *positional encoding* for Natual Language Processing (NLP) tasks, which uses a combination of sines and cosines at multiple frequencies, in order to encode the position of a word in a sequence. In theory, this approach could also be adopted for encoding ROI geometry (e.g., the position of ROI centers in the image along a certain axis). However, this may fail to capture the interrelations of candidate ROIs in a relative manner, as the encoded information in the NMS task is far more complex compared to [35]. As an alternative, in this paper we approached the task by encoding all the representations of the N input candidate detections in a relative-to-each-ROI manner. Thus, the keys and values of the Scale Dot-Product Attention are represented in a relative-to-each-query representation scheme. For example, the j^{th} key may be represented differently for the i^{th} query, compared to its representation for the $(i+1)^{th}$ query. Although this increases the presented method's memory complexity, each query is allowed to represent the keys and the values relatively to itself. Thus, for N candidate detections, $\mathbf{Q} \in \mathbb{R}^{N \times 1 \times d_a}$, $\mathbf{K} \in \mathbb{R}^{N \times N \times d_a}$ and $\mathbf{V} \in \mathbb{R}^{N \times N \times d_a}$, the output is $\mathbf{F}^D \in \mathbb{R}^{N \times 1 \times d_a}$. A residual connection [45] is applied between $\mathbf{Q}$ and $\mathbf{F}^D$. Due to the increased number of dimensions of $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$, batch matrix multiplication is employed in Eq. (4) to speed up the process. The architecture of this module is illustrated in Figure 3.

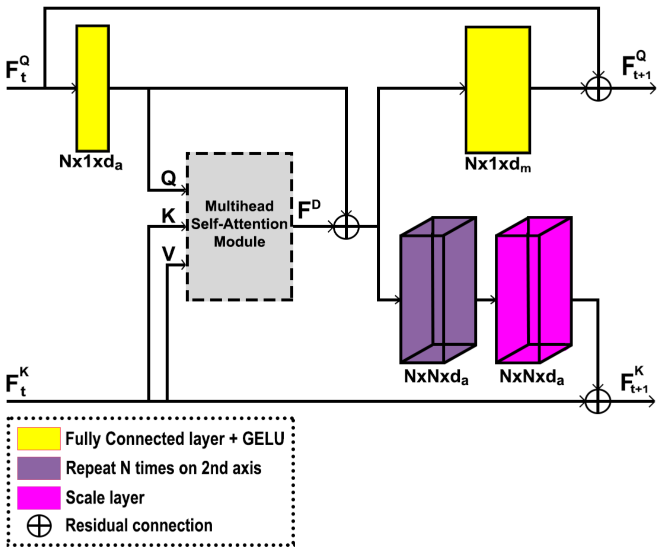


Fig. 4: Illustration of the Joint Processing Module.

Joint Processing Module: In this neural module, the representations of the candidate detections are jointly and simultaneously refined, mainly through the Multihead Self-Attention mechanism. The Joint Processing Module receives as its input $\mathbf{F}_t^Q \in \mathbb{R}^{N \times 1 \times d_m}$, which holds the current representations of all candidate detections, as well as $\mathbf{F}_t^K \in \mathbb{R}^{N \times N \times d_a}$, which holds the current relative-to-each-candidate-detection representations, for all N candidate detection. The architecture of the Joint Processing Module is shown in Fig. 4. The queries and keys are formed as:

$$\begin{aligned} \mathbf{Q} &= \mathbf{F}_t^Q \mathbf{C}^Q, \\ \mathbf{K} &= \mathbf{F}_t^K, \\ \mathbf{V} &= \mathbf{K}, \end{aligned} \tag{7}$$

where $\mathbf{C}^Q \in \mathbb{R}^{d_m \times d_a}$ stands for the weights of a fully connected neural layer. The new representations of the candidate detections, which is the output of this module, are formed as:

$$\begin{aligned} \mathbf{F}_{t+1}^Q &= \mathbf{F}^D \mathbf{C}^D + \mathbf{F}_t^Q, \\ \mathbf{F}^D &= \mathit{Multihead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}), \end{aligned} \tag{8}$$

where $\mathbf{C}^D \in \mathbb{R}^{d_a \times d_m}$ also denotes the weights of a fully connected layer.

Finally, the relative-to-each-candidate-detection representations $\mathbf{F}_K$ are refined as:

$$\mathbf{F}_{t+1}^K = \mathbf{F}_t^K + \mathbf{F}^S \otimes \mathbf{C}^K, \tag{9}$$

where $\mathbf{F}^S$ is derived from $\mathbf{F}^D$, by repeating it N times along its second dimension, and $\mathbf{C}^K$ are learned weights of a *Scale Layer* that we introduce, performing an element-wise multiplication between its weights and an input representation. Its purpose is to select the degree of information which will flow from $\mathbf{F}^S$ to $\mathbf{F}^K_{t+1}$ in each Joint Processing Module.

Masking: A masking approach has been integrated into the self-attention mechanism of the proposed architecture. For N candidate detections, sorted in descending order based on the detector's score values, we mask the values of the input of the softmax function in Eq. (4). Without loss of generality, masking is detailed below for the simplest case, where the number of heads $H = 1$.

Given a candidate ROI $\mathbf{r}_i^{in}$, an its associate ROI $\mathbf{r}_j^{in}$ and $\mathbf{S} = \frac{\mathbf{QK}^T}{\sqrt{d_k}}$, masking is defined as:

$$S_{ij} = \begin{cases} -\infty, & \text{if } IoU(\mathbf{r}_i^{in}, \mathbf{r}_j^{in}) < 0.2 \\ 0.1 \cdot S_{ij}, & \text{if } IoU(\mathbf{r}_i^{in}, \mathbf{r}_j^{in}) \geq 0.2 \text{ and } j > i \\ S_{ij}, & \text{otherwise} \end{cases} \quad (10)$$

This masking operation is employed for two reasons. First, each ROI must be prevented from attending to spatially distant candidate detections. The overlap of ROIs is used to determine whether S_{ij} should be set to $-\infty$, before applying the softmax function. If yes, the attention weight linking $\mathbf{r}_i^{in}$ to $\mathbf{r}_j^{in}$ (after the softmax has been applied) will be zeroed out. Second, an additional motivation is our attempt to replicate the behaviour of Greedy NMS, where a candidate detection is characterized as duplicate, thus marked for suppression, when another, higher-scoring detection spatially covers the same object. In the proposed neural architecture this can be accomplished by forcing (through masking) the internal representation of a candidate detection to be modified by attending mainly to representations that correspond to ROIs higher-scoring than itself.

Network Architecture: For a set of N candidate detections, sorted in descending order based on the detector's confidence score values, the proposed DNN uses as input their corresponding appearance-based representations $\mathbf{A}$ and their geometry-based representations $\mathbf{G}$. FMoD representations of 3 pyramid levels are employed as $\mathbf{A} \in \mathbb{R}^{N\times 1\times 315}$. The extracted geometry-based ROI representations, which is $\mathbf{G} \in \mathbb{R}^{N\times N\times 14}$, are assigned to $\mathbf{G}^K$ as it contains the relative-to-each-candidate-detection representations. Its diagonal, derived from the first two dimensions, forms $\mathbf{G}^Q \in \mathbb{R}^{N\times 1\times 14}$. The representations derived from a fusion between $\mathbf{A}$ and $\mathbf{G}^Q$ form $\mathbf{F}^Q \in \mathbb{R}^{N\times 1\times d_m}$. This fusion is mainly accomplished by concatenating and applying fully-connected layers between the two types of representations. In addition, the representations derived from a fusion between $\mathbf{A}$ and $\mathbf{G}^K$ form $\mathbf{F}^K \in \mathbb{R}^{N\times N\times d_a}$. Both $\mathbf{F}_Q$ and $\mathbf{F}_K$ are used as input to the first Joint Processing Module.

A stack of Joint Processing Modules, sequentially connected, are in charge of refining representations $\mathbf{F}_Q$ and $\mathbf{F}_K$. Finally, after applying two fully connected layers on $\mathbf{F}^Q$, the DNN uses a softmax function to output the final NMS scores. The described architecture of the model is depicted in Fig. 5. The Gaussian Error Linear Unit (GELU) is used as activation function by the network. Layer normalization [46] is applied on the output of residual connections and dropout [47] is used for regularization, similarly to [35].

Training: The weighted binary cross entropy was selected as the training objective of the proposed neural architecture. In particular, the loss function is defined as:

$$L = -\textstyle\sum_{i=1}^{N}(w_1 y_i \log(r_i^{scores_{NMS}}) + w_0(1-y_i)\log(1-r_i^{scores_{NMS}})), \quad (11)$$

where N is the number of candidate detections, $\mathbf{r}^{scores_{NMS}}$ are the output NMS scores, $\mathbf{w}$ are class weights and $\mathbf{y}$ are the labels derived from a matching function, given a specific IoU value. In particular, $y_i \in \{1, 0\}$ indicates whether the i^{th} detection was successfully matched to an object or not. A detection is matched successfully to an object, when the IoU between its ROI and an object's 2D bounding box is higher or equal to a matching threshold, and that specific object hasn't been matched to any higher scoring detection. In this paper, this matching threshold was set to 0.5. The class weights, are defined as:

$$w_j = (1-j)(1-c_j)\frac{\sum_{m=1}^{M}(count(\mathbf{G}_m)+max(0,count(\mathbf{R}_m)-count(\mathbf{G}_m)))}{\sum_{m=1}^{M}(max(0,count(\mathbf{R}_m))-count(\mathbf{G}_m))} + jc_j\frac{\sum_{m=1}^{M}(count(\mathbf{G}_m)+max(0,count(\mathbf{R}_m)-count(\mathbf{G}_m)))}{\sum_{m=1}^{M}count(\mathbf{G}_m)}, \quad (12)$$

where, M is the number of images in a dataset, $\mathbf{R}_m$ are the candidate ROIs in the m^{th} image, $\mathbf{G}_m$ are the ground-truth ROIs in the m^{th} image, and c_1 is a predefined scalar (e.g., 0.1) to balance the class weights.

IV. Experimental Evaluation

The performance of Seq2Seq-NMS was evaluated on three separate datasets for the person detection task. In order to assess its accuracy regardless of the selected object detector, candidate ROIs from three different detectors were employed. These differ significantly in the way they approach the object detection task in general.

The neural architecture used for evaluation consists of 4 Joint Processing Modules. We set $d_m = 256$, and $d_a = \frac{d_m}{2} = 128$. The Multihead Self-Attention module uses $H = 2$ attention heads and thus $d_q = d_k = d_v = \frac{128}{H} = 64$. Appearance-based ROI representations computed from 3-level FMoD were used, with 0^{th} level ROI maps extracted at resolution $W_0 \times H_0$ = 160×160 pixels.

In each dataset, Seq2Seq-NMS was compared against both neural and non-neural NMS algorithms. While some are focused more on achieving state-of-the-art results, others succeed in attaining ultra-fast inference times. The first competing method is a baseline Greedy NMS approach running on GPU. The second is TorchVision's[2] GreedyNMS implemented to run very fast on GPUs. Additionally, Soft-NMS [27] was tested, i.e., a non-neural NMS method widely used as a more accurate replacement for Greedy NMS. Evaluation was conducted

[2]https://pytorch.org/vision/stable/ops.html#torchvision.ops.nms

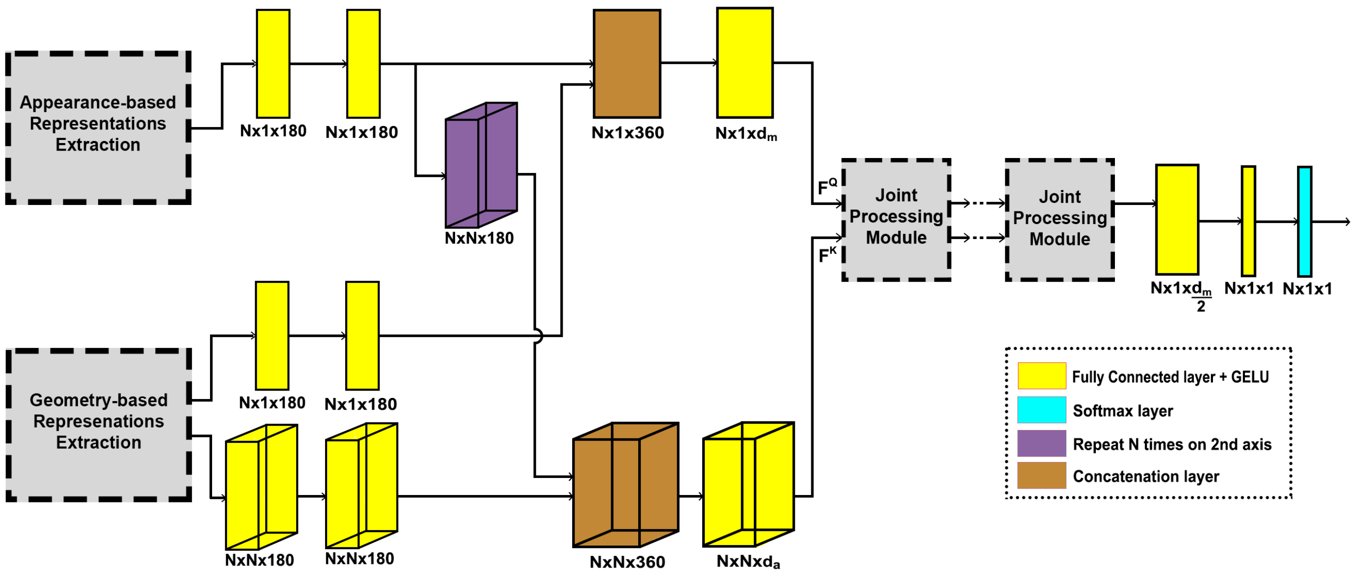


Fig. 5: Seq2Seq-NMS architecture. N is the number of input candidate ROIs/detections.

using both the linear and the Gaussian weighting functions (referred to as Soft-NMS$_\mathrm{L}$ and Soft-NMS$_\mathrm{G}$, respectively). The method was executed on CPU. Another competing algorithm we employed is Fast-NMS [30], a generally faster, non-neural replacement for standard NMS, that suffers a marginal penalty regarding accuracy. Fast-NMS is executed on GPU. Additionally, several variants of Cluster-NMS [32], i.e., a more recent non-neural method, was also selected for comparison purposes. Below, the term Cluster-NMS$_\mathrm{S}$ is used to imply the case where the score penalty mechanism is used, and Cluster-NMS$_\mathrm{D}$ the scenario where the normalized central point distance is added. In the latter case, the method is equivalent to DIoU-NMS [31]. Moreover, the term Cluster-NMS$_\mathrm{S+D}$ is used when both of these mechanisms are utilized. Finally, Cluster-NMS$_\mathrm{S+D+W}$ indicates the case where a weighted strategy similar to [28] is followed. More details regarding these variations can be found in [32]. The last approach selected for comparison purposes is GossipNet [2], a neural NMS method achieving state-of-the-art accuracy.

The hyperparameters of all non-neural methods were tuned so as to report the best achieved results on 0.5 IoU matching threshold. Both for GossipNet and for the proposed method, the class weights are computed through Eq. (12), using a common, dataset-specific c_1 value. Evaluation was performed on a PC using an Intel Core i7-7700 CPU and an NVIDIA GeForce GTX 1070 Ti GPU with 8GB of memory, both for training and inference. The employed evaluation metrics are $\mathrm{AP}_{0.5}$, $\mathrm{AP}_{0.5}^{0.95}$ and inference times. $\mathrm{AP}_{0.5}$ corresponds to the average precision for 0.5 IoU, while $\mathrm{AP}_{0.5}^{0.95}$ to the mean average precision for IoU ranging from 0.5 to 0.95 with a step size of 0.05. Finally, for all methods, all candidate detections were used during the evaluation process, without any thresholding.

A. PETS

PETS [48] is a relatively small dataset, whose images were collected from static surveillance cameras and provide diverse levels of occlusion. The average number of people depicted in an image is about 14. We use candidate detections from [19], a non-neural person detection method, designed specifically to handle occlusions. Due to the fact that the number of ROIs is extremely large in some images, we first apply TorchVision NMS with the relaxed 0.8 IoU threshold on all deployed methods as a typical preprocessing step, commonly utilized in NMS literature. Since the number of candidate detections for the NMS step remains huge even after applying TorchVision NMS, we report results for various ROI retention thresholds, with only a maximum number of higher-scoring detections kept in each case. Typically, candidate ROIs with near-zero confidence scores, as assigned by the detector, are not true positive samples and, thus, do not significantly affect the accuracy of traditional non-neural NMS methods.

The proposed method was trained using the ADAM [49] optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.99$ and $\epsilon = 10^{-9}$ for 8 epochs. The learning rate was set to 10^{-4} for the first 4 epochs, then to 10^{-5} for the next 3 epochs and finally it as to 10^{-6} for the final epoch. At most the 600 highest-scoring candidate detections were only used as the input sequence for each image when training the network, due to memory limitations. Regarding GossipNet, architecture and training process was based on [2].

Table I reports the results of the proposed method, as well as of competing NMS approaches. As mentioned above, the employed object detector [19] outputs a large number of candidate ROIs, thus leading to increased GPU memory consumption for both the proposed method and GossipNet. Typically, most candidate detections that can be successfully matched to ground-truth objects are assigned higher confidence scores by the detector, compared to ROIs with lower scores (e.g., < 0.1) which are mostly false positive samples.

TABLE I: COMPARISON OF DIFFERENT NMS METHODS ON THE PETS DATASET, USING DETECTIONS FROM [19]. BOTTOM LINE REPORTS ON THE PROPOSED METHOD.

Method	**Device**	**Max dets. = 400**			**Max dets. = 600**			**Max dets.= 800**			**Max dets. = 1200**			**Max dets. = All**		
		AP$_{0.5}$	**AP**$_{0.5}^{0.95}$	**Average Inference Time (ms)**	**AP**$_{0.5}$	**AP**$_{0.5}^{0.95}$	**Average Inference Time (ms)**	**AP**$_{0.5}$	**AP**$_{0.5}^{0.95}$	**Average Inference Time (ms)**	**AP**$_{0.5}$	**AP**$_{0.5}^{0.95}$	**Average Inference Time (ms)**	**AP**$_{0.5}$	**AP**$_{0.5}^{0.95}$	**Average Inference Time (ms)**
Original NMS IoU>0.4	GPU	75.4%	31.7%	3.8	76.2%	31.6%	2.5	76.5%	31.3%	4.5	76.3%	31.2%	5.0	76.3%	31.2%	6.3
Original NMS IoU>0.5	GPU	73.1%	31.0%	4.1	73.6%	30.8%	3.6	73.6%	30.8%	5.7	73.6%	30.8%	7.5	73.6%	30.8%	9.8
Original NMS IoU>0.6	GPU	65.7%	28.6%	6.8	65.7%	28.6%	5.5	65.7%	28.6%	10.0	65.7%	28.6%	12.8	65.7%	28.6%	16.0
Original NMS IoU>0.4 (TorchVision)	GPU	75.6%	31.7%	0.4	76.1%	31.6%	0.5	76.4%	31.3%	0.5	76.4%	31.3%	0.8	76.4%	31.3%	0.8
Original NMS IoU>0.5 (TorchVision)	GPU	72.7%	30.8%	0.6	73.2%	30.8%	0.5	73.2%	30.8%	0.5	73.2%	30.8%	0.7	73.2%	30.8%	0.8
Original NMS IoU>0.6 (TorchVision)	GPU	65.1%	28.5%	0.5	65.1%	28.5%	0.5	65.1%	28.5%	0.5	65.1%	28.5%	0.7	65.1%	28.5%	0.8
Soft-NMS$_L$	CPU	76.6%	31.8%	27.1	76.6%	31.5%	50.3	76.5%	31.3%	64.0	76.3%	31.2%	98.5	76.3%	31.2%	143.3
Soft-NMS$_G$	CPU	77.4%	32.7%	23.8	77.2%	32.3	39.2	76.7%	32.0%	54.4	76.0%	31.8%	89.5	75.8%	31.7%	154.7
Fast-NMS	GPU	75.3%	31.5%	3.2	75.2%	31.1%	1.6	75.0%	31.0%	1.5	74.5%	30.8%	2.8	74.4%	30.7%	3.5
Cluster-NMS	GPU	75.9%	31.4%	2.9	76.4%	31.3%	3.6	76.4%	31.3%	3.7	76.4%	31.3%	5.5	76.4%	31.3%	8.0
Cluster-NMS$_S$	GPU	75.4%	31.6%	4.8	74.6%	31.2%	2.9	74.0%	30.8%	3.8	73.6%	30.7%	5.0	73.0%	30.6%	7.3
Cluster-NMS$_D$	GPU	76.2%	31.6%	5.4	76.6%	31.3%	4.1	76.5%	31.2%	5.1	76.5%	31.1%	7.7	76.5%	31.1%	10.0
Cluster-NMS$_{S+D}$	GPU	76.4%	31.9%	3.7	76.3%	31.6%	4.2	75.9%	31.3%	5.4	75.2%	31.0%	8.4	74.9%	30.9%	12.1
Cluster-NMS$_{S+D+W}$	GPU	76.4%	31.9%	28.3	76.3%	31.6%	55.7	75.9%	31.3%	88.2	75.2%	31.0%	171.9	74.9%	30.9%	292.3
GossipNet	GPU	79.6%	34.4%	19.1	81.7%	35.1%	32.5	82.8%	35.6%	48.0	83.6%	35.9%	73.9	83.8%	36.1%	107.2
Seq2Seq-NMS	GPU	80.9%	36.5%	11.1	82.9%	37.1%	11.3	83.9%	**37.3%**	11.9	**84.7%**	**37.3%**	14.7	**-**	**-**	-

Thus, in this paper, we attempt to evaluate whether the lowest scoring detections have an impact on the performance of the proposed and of the competing NMS methods. In Table I, the results of each method are reported using N candidate detections as input, for different values of N. As it can be seen, the performance of the non-neural methods is not improved when the lowest-scoring detections ($N > 800$) are used. In contrast, both GossipNet and the proposed method achieve more accurate results for longer input sequences (more candidate ROIs per image).

This advantage of the neural NMS methods most likely stems from the inability of non-neural algorithms to jointly process the candidate detections in order to rescore/suppress them. Regarding the proposed method, its performance improves as more relations between a corresponding candidate detection and other neighboring ROIs are built. Additionally, it is able to handle "hard" True Positive (TP) candidate detections and classify them as correct ones, even if the employed detector originally assigned them very low confidence scores.

Overall, the proposed method achieved both the best AP$_{0.5}$ and the best AP$_{0.5}^{0.95}$, against all competing approaches, when using an input sequence of 1200 detections. The obtained AP$_{0.5}$ was 84.7%, which is a $+7.3\%$ improvement against Soft-NMS$_G$, the non-neural method with the best AP$_{0.5}$, and a $+0.9\%$ improvement against GossipNet. Notably, when using only a small number of the highest-scoring candidate detections (e.g, N = 400), the proposed method still achieves better results compared to all non-neural NMS algorithms. Regarding inference runtimes, it needs 14.7 ms to run per image when N = 1200, since the required edge maps are computed in parallel with the object detector's inference. Thus it is faster than GossipNet, as well as less affected by the number of candidate detections used as input. However, it is slower than the non-neural methods running on GPU.

B. COCO Person

COCO 2014 is a large dataset consisting 82,783 images for training and 40,504 images for validation/testing. Although it contains 80 labeled classes, only the "person" class was used for evaluating the proposed method. Candidate detections (extracted by Faster R-CNN [21]) and validation set splits from [2] were also employed here. The first data subset, referred to as "minival", contains 5000 images, while the second subset, referred to as "minitest", contains 35000 images. The average ground-truth number of people depicted in an image is about 2.17. However, when taking into account only the images that actually contain visible people, this average ground-truth number is 4.01.

The proposed method was trained using the ADAM optimizer with β_1 = 0.9, β_2 = 0.99 and $\epsilon = 10^{-9}$ for 8 epochs. The learning rate was set to 10^{-4} for the first 4 epochs, then to 10^{-5} for the next 3 epochs and, finally, to 10^{-6} for the final epoch. At most the 600 highest-scoring candidate detections per image were employed as input at the training stage. GossipNet architecture and training again followed [2]. Final model parameters were selected according to achieved accuracy in the minival (validation) set during training, both for the proposed method and for GossipNet.

As it is shown in Table II, the proposed method achieves the best AP$_{0.5}$ against all competing NMS methods, in both the minival and the minitest sets. It leads to an AP$_{0.5}$ = 68.1% in the minival set, which is an $+0.8\%$ improvement against Soft-NMS$_L$, the best non-neural method, and $+0.4\%$ against GossipNet. In the minitest set, it achieved AP$_{0.5}$ = 67.5%, which is an $+0.9\%$ improvement against Soft-NMS$_L$ and $+0.6\%$ against GossipNet. As far as AP$_{0.5}^{0.95}$ is concerned, the proposed method is outperformed only by Cluster-NMS$_{S+D+W}$, achieving $AP_{0.5}^{0.95}$ = 37.2% and $AP_{0.5}^{0.95}$ = 36.9% in the minival and the minitest sets, respectively. These figures are on par with the results of Soft-NMS$_L$. Regarding inference times, the proposed method is close to that of Cluster-NMS$_{S+D+W}$.

TABLE II: COMPARISON OF DIFFERENT NMS METHODS ON THE COCO DATASET, USING DETECTIONS FROM FASTER-RCNN. BOTTOM LINE REPORTS ON THE PROPOSED METHOD.

Method	Device	Minival set		Minitest set		Average Inference Time (ms)
		$AP_{0.5}$	$AP_{0.5}^{0.95}$	$AP_{0.5}$	$AP_{0.5}^{0.95}$	
Original NMS IoU>0.4	GPU	66.1%	35.7%	65.4%	35.5%	4.3
Original NMS IoU>0.5	GPU	66.0%	36.0%	65.3%	35.8%	4.5
Original NMS IoU>0.6	GPU	64.1%	35.9%	63.2%	35.5%	4.8
Original NMS IoU>0.4 (TorchVision)	GPU	66.2%	35.8%	65.4%	35.6%	0.59
Original NMS IoU>0.5 (TorchVision)	GPU	66.0%	36.0%	65.2%	35.8%	0.60
Original NMS IoU>0.6 (TorchVision)	GPU	64.0%	35.9%	63.1%	35.5%	0.60
Soft-NMS$_L$	CPU	67.3%	37.2%	66.6%	36.9%	5.8
Soft-NMS$_G$	CPU	67.0%	37.0%	66.2%	36.7%	6.6
Fast-NMS	GPU	65.0%	35.5%	64.1%	35.3%	2.5
Cluster-NMS	GPU	66.2%	35.6%	65.4%	35.5%	4.9
Cluster-NMS$_S$	GPU	66.0%	36.0%	65.3%	36.0%	4.8
Cluster-NMS$_D$	GPU	66.2%	35.8%	65.5%	35.6%	7.6
Cluster-NMS$_{S+D}$	GPU	66.5%	36.8%	65.8%	36.6%	7.6
Cluster-NMS$_{S+D+W}$	GPU	66.7%	**37.8%**	65.8%	**37.6%**	9.7
GossipNet	GPU	67.7%	36.1%	66.9%	36.0%	6.1
Seq2Seq-NMS	GPU	**68.1%**	37.2%	**67.5%**	36.9%	10.1

C. CrowdHuman

The CrowdHuman dataset has been recently released to specifically target human detection in crowded areas. It contains 15000 images for training, 4370 images for validation and 5000 images for testing. The average number of persons in an image is 22.64, with various types of occlusions. All methods were trained on the CrowdHuman training set and evaluated on its validation set, since annotations weren't provided for the test set.

Candidate detections were extracted using YOLOv4 [25] for this dataset. The detector was trained for 6000 steps, using the Nesterov Accelerated Gradient [50] (NAG) optimizer with a momentum equal to 0.95. Mini-batch size was set to 64 with 64 subdivisions, while images were re-scaled to a resolution of 608×608 pixels. The learning rate was set to 10^{-3}, decreasing by 0.1 at the 4800-th and the 5400-th step.

The proposed NMS DNN was trained using the ADAM optimizer with β_1 = 0.9, β_2 = 0.999 and ϵ = 10^{-8} for 12 epochs. The learning rate was set to 10^{-4} for the first 5 epochs, then to 10^{-5} for the next 5 epochs and finally to 10^{-6} for the final 2 epochs. Once more, at most the 600 highest-scoring candidate detections per image were employed as input at the training stage. During inference, the maximum number of detections per image was set to 800, due to memory constraints. GossipNet was trained for 120000 iterations, with a learning rate set to 10^{-4} and decreasing by 0.1 at the 60000-th and the 90000-th iteration. Due to the very large number of candidate ROIs on several images, TorchVision NMS with the relaxed 0.8 IoU threshold was again applied as a preprocessing step, before feeding the proposed DNN and the GossipNet model with input. This strategy was followed during both the training and the inference stage.

As shown in Table III, the proposed method achieves the best $AP_{0.5}$, along with GossipNet, against all competing NMS methods. In particular, it leads to an $AP_{0.5}$ of 83.7%, which is a $+1.2\%$ improvement against Soft-NMS$_L$. Moving on to the $AP_{0.5}^{0.95}$ metric, the proposed method was slightly outperformed only by Soft-NMS$_L$. Overall, it achieves a $AP_{0.5}^{0.95}$ of 49.1%. Regarding inference runtimes, the proposed method is faster than all non-GPU approaches and than GossipNet, since the latter's required inference time depends strongly on the number of candidate detections per image. The average inference time of the proposed method was 11.1 ms, given that the edge maps needed for the appearance-based ROI representations extraction are computed in parallel with the detector's inference stage.

TABLE III: COMPARISON OF DIFFERENT NMS METHODS ON THE CROWDHUMAN DATASET, USING DETECTIONS FROM YOLOv4. BOTTOM LINE REPORTS ON THE PROPOSED METHOD.

Method	Device	Validation set		Average Inference Time (ms)
		$AP_{0.5}$	$AP_{0.5}^{0.95}$	
Original NMS IoU>0.4	GPU	76.9%	44.8%	2.2
Original NMS IoU>0.5	GPU	80.6%	47.0%	2.9
Original NMS IoU>0.6	GPU	82.0%	48.2%	4.6
Original NMS IoU>0.4 (TorchVision)	GPU	77.1%	44.9%	0.4
Original NMS IoU>0.5 (TorchVision)	GPU	80.7%	47.0%	0.4
Original NMS IoU>0.6 (TorchVision)	GPU	82.0%	48.3%	0.4
Soft-NMS$_L$	CPU	82.5%	**49.3%**	30.8
Soft-NMS$_G$	CPU	81.7%	48.6%	39.8
Fast-NMS	GPU	81.0%	48.0%	1.6
Cluster-NMS	GPU	82.0%	48.2%	4.7
Cluster-NMS$_S$	GPU	80.3%	47.3%	3.4
Cluster-NMS$_D$	GPU	82.2%	48.6%	5.1
Cluster-NMS$_{S+D}$	GPU	81.4%	48.2%	5.5
Cluster-NMS$_{S+D+W}$	GPU	81.4%	48.2%	47.9
GossipNet	GPU	**83.7%**	49.0%	16.1
Seq2Seq-NMS$_{800}$	GPU	**83.7%**	49.1%	11.1

D. Discussion

Overall, the proposed Seq2Seq-NMS DNN achieves top accuracy on the $AP_{0.5}$ metric in all three datasets, being on par with GossipNet in CrowdHuman and outperforming all competing methods in the remaining two datasets. The results show that Seq2Seq-NMS can successfully capture interrelations between candidate detections for the person detection task, based both on their visual appearance and their geometry. The three datasets used for evaluation contain images with a great variety of visible persons density, ranging from images of individual people to photographs of large crowds, indicating that Seq2Seq is suitable for generic person detection.

Besides the measurements reported in Tables I, II and III, an ablation study was also performed regarding the proposed masking operation (as described in Section III-C) of the self-attention mechanism. Omitting masking led to reduced accuracy rates, or to training convergence failures in cases with huge numbers of candidate ROIs per image. Most likely, the importance of masking stems from the fact that it enforces an ordering constraint on how the internal representation of each candidate detection is shaped: thanks to masking, its form is finalized by attending mainly to representations that correspond to ROIs higher-scoring than itself, using the Scaled Dot-Product Attention mechanism. Thus, in our view, this finding supports the validity of the sequence-to-sequence formulation of the NMS task.

Regarding the $AP_{0.5}^{0.95}$ metric, Seq2Seq-NMS outperforms most competing methods but achieves top accuracy only on the PETS dataset. Most likely, this behaviour can be explained by the fact that the labels of the candidate detections for Seq2Seq-NMS training were created based on [5] using an IoU threshold of 0.5.

Moving on to required inference running times, the proposed method is relatively slower than non-neural, mostly less accurate, GPU-executed algorithms. However, when compared against DNN architectures for NMS, such as GossipNet, the inference runtime of Seq2Seq-NMS seems less affected by the input sequence length (number of candidate detections), thus achieving faster inference when processing longer sequences, as shown in Tables I and III.

Finally, evaluation in this paper was performed using candidate detections from: (a) a non-neural detector [19], (b) a two-stage DNN-based detector [21], and (c) a one-stage DNN-based detector [25]. Therefore, it can be claimed that the behaviour of Seq2Seq-NMS is rather invariant to the type of the deployed person detector, achieving an improved $AP_{0.5}$ performance against several competing NMS methods. This can be easily explained by the use of FMoD for visual appearance representation of the cropped candidate ROIs, instead of relying on learnt convolutional representations computed by the DNN-based person detector.

V. Conclusions

Non-Maximum Suppression (NMS) is the last step in a typical object detection system. Heavy occlusions when detecting humans in images of crowded areas imposes great challenges to most NMS methods, despite the importance of such a task for human safety-centric applications. This paper presented Seq2Seq-NMS, a novel deep neural architecture for performing NMS in similar hard cases, which relies on reformulating NMS as a sequence-to-sequence problem. The proposed method relies on the Multihead Scaled Dot-Product Attention mechanism in order to efficiently capture interrelations across the sequence of candidate detections, while also jointly exploiting visual appearance and geometric properties of the input ROIs in order to better represent them. Quantitative evaluation on three public person detection datasets, each one using a different detector, showed that Seq2Seq-NMS can provide state-of-the-art results at the IoU threshold used for annotating its training dataset, with acceptable inference runtime requirements and good behaviour for large numbers of raw candidate ROIs per image. Future extensions may focus on a training strategy suitable for various IoU thresholds, as well as on assessing method performance when using a different modality for describing the visual appearance of the cropped input detections (instead of edge maps). This may enable an extension of the proposed method to multiclass object detection tasks.

VI. Acknowledgments

This project has received funding from the European Unions 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. Liu, D. Huang, and Y. Wang, “Adaptive NMS: Refining pedestrian detection in a crowd,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[2] J. Hosang, R. Benenson, and B. Schiele, “Learning Non-Maximum Suppression,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[3] C. Symeonidis, I. Mademlis, N. Nikolaidis, and I. Pitas, “Improving neural Non-Maximum Suppression for object detection by exploiting interest-point detectors,” in *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2019.

[4] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei, “Relation networks for object detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[5] T.-Y. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: Common objects in context,” in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2014.

[6] M. Everingham, L. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (VOC) challenge,” *International Journal of Computer Vision*, vol. 88, pp. 303–338, 2009.

[7] I. Mademlis, N. Nikolaidis, and I. Pitas, “Stereoscopic video description for key-frame extraction in movie summarization,” in *Proceedings of the EURASIP European Signal Processing Conference (EUSIPCO)*. IEEE, 2015.

[8] L. Chen, S. Lin, X. Lu, D. Cao, H. Wu, C. Guo, C. Liu, and F.-Y. Wang, “Deep neural network-based vehicle and pedestrian detection for autonomous driving: A survey,” *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[9] C. Symeonidis, E. Kakaletsis, I. Mademlis, N. Nikolaidis, A. Tefas, and I. Pitas, “Vision-based UAV safe landing exploiting lightweight deep neural networks,” in *Proceedings of the International Conference on Image and Graphics Processing (ICIGP)*, 2021.

[10] C. Papaioannidis, I. Mademlis, and I. Pitas, “Autonomous UAV safety by visual human crowd detection using multi-task deep neural networks,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[11] E. Kakaletsis, E. Symeonidis, M. Tzelepi, I. Mademlis, T. A., N. Nikolaidis, and I. Pitas, "Computer vision for autonomous UAV flight safety: An overview and a vision-based safe landing pipeline example," *ACM Computing Surveys*, 2021, accepted.

[12] P. Nousi, E. Patsiouras, A. Tefas, and I. Pitas, "Convolutional Neural Networks for visual information analysis with limited computing resources," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2018.

[13] I. Mademlis, V. Mygdalis, N. Nikolaidis, and I. Pitas, "Challenges in autonomous UAV cinematography: an overview," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, 2018.

[14] I. Mademlis, N. Nikolaidis, A. Tefas, I. Pitas, T. Wagner, and A. Messina, "Autonomous unmanned aerial vehicles filming in dynamic unstructured outdoor environments," *IEEE Signal Processing Magazine*, vol. 36, pp. 147–153, 2018.

[15] P. Nousi, I. Mademlis, I. Karakostas, A. Tefas, and I. Pitas, "Embedded UAV real-time visual object detection and tracking," in *Proceedings of the IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 2019.

[16] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[17] P. Viola and M. Jones, "Robust real-time face detection," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2001.

[18] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[19] S. Tang, M. Andriluka, A. Milan, K. Schindler, S. Roth, and B. Schiele, "Learning people detectors for tracking in crowded scenes," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2013, pp. 1049–1056.

[20] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[21] 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*, vol. 39, no. 6, pp. 1137–1149, 2017.

[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, and A. Berg, "SSD: Single shot multibox detector," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2016.

[23] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[24] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *ArXiv*, vol. abs/1804.02767, 2018.

[25] A. Bochkovskiy, C.-Y. Wang, and H. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *ArXiv*, vol. abs/2004.10934, 2020.

[26] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[27] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, "Soft-NMS: Improving object detection with one line of code," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[28] H. Zhou, Z. Li, C. Ning, and J. Tang, "CAD: Scale invariant framework for real-time object detection," in *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2017.

[29] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang, "Acquisition of localization confidence for accurate object detection," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2018.

[30] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT: Real-time instance segmentation," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.

[31] "Distance-IoU loss: Faster and better learning for bounding box regression," *Association for the Advancement of Artificial Intelligence (AAAI)*, vol. 34, no. 07, pp. 12 993–13 000, 2020.

[32] Z. Zheng, P. Wang, D. Ren, W. Liu, R. Ye, Q. Hu, and W. Zuo, "Enhancing geometric factors in model learning and inference for object detection and instance segmentation," *ArXiv*, vol. abs/2005.03572, 2020.

[33] S. H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[34] X. Huang, Z. Ge, Z. Jie, and O. Yoshie, "NMS by representative region: Towards crowded pedestrian detection by proposal pairing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, 2017.

[36] I. Mademlis, A. Tefas, N. Nikolaidis, and I. Pitas, "Compact video description and representation for automated summarization of human activities," in *Proceedings of the INNS Conference on Big Data*. Springer, 2016.

[37] I. Mademlis, A. Tefas, N. Nikolaidis, and I. Pitas, "Multimodal stereoscopic movie summarization conforming to narrative characteristics," *IEEE Transactions on Image Processing*, vol. 25, no. 12, pp. 5828–5840, 2016.

[38] I. Mademlis, A. Tefas, and I. Pitas, "A salient dictionary learning framework for activity video summarization via key-frame extraction," *Information Sciences*, vol. 432, pp. 319 – 331, 2018.

[39] I. Mademlis, A. Tefas, and I. Pitas, "Regularized SVD-based video frame saliency for unsupervised activity video summarization," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2018.

[40] ——, "Greedy salient dictionary learning with optimal point reconstruction for activity video summarization," in *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2018.

[41] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[42] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond Bags of Features: Spatial pyramid matching for recognizing natural scene categories," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[43] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[44] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. S. A. Ku, and D. Tran, "Image transformer," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[46] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016.

[47] 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*, vol. 15, no. 56, pp. 1929–1958, 2014.

[48] J. M. Ferryman and A. Ellis, "PETS2010: Dataset and challenge," in *Proceedings of the IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, 2010.

[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[50] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$," *Doklady AN USSR*, vol. 269, pp. 543–547, 1983.

8.3 Remote Multilinear Compressive Learning with Adaptive Compression

The appended paper follows.

Remote Multilinear Compressive Learning with Adaptive Compression

Dat Thanh Tran*, Moncef Gabbouj*, Alexandros Iosifidis†
*Department of Computing Sciences, Tampere University, Tampere, Finland
†Department of Electrical and Computer Engineering, Aarhus University, Aarhus, Denmark
Email:{thanh.tran,moncef.gabbouj}@tuni.fi, ai@ece.au.dk

***Abstract*—Multilinear Compressive Learning (MCL) is an efficient signal acquisition and learning paradigm for multidimensional signals. The level of signal compression affects the detection or classification performance of a MCL model, with higher compression rates often associated with lower inference accuracy. However, higher compression rates are more amenable to a wider range of applications, especially those that require low operating bandwidth and minimal energy consumption such as Internet-of-Things (IoT) applications. A large majority of communication protocols have supports for adaptive data transmission to maximize the throughput and minimize energy consumption. By developing compressive sensing and learning models that can operate with an adaptive compression rate, we can maximize the informational content throughput of the whole application. In this paper, we propose a novel optimization and formulation that enable such a feature for MCL models. Our proposal not only speeds up the experimentation process required for training and validation but also enables practical implementation of adaptive compressive signal acquisition and inference systems.**

I. Introduction

Nowadays, Internet-of-Things (IoT) technologies can allow us to deploy smart sensors to every corner of the world, even in the most remote areas. This is thanks to the development of low-cost electronics and embedded devices, as well as the advancements in wireless communication technologies such as those in Low Power Wide Area Network (LPWAN). An IoT system is a cyber-physical system in which the physical (client) side often consists of a network of smart sensors and embedded devices deployed in various physical locations, with capability to acquire signals and perform light-weight computation before transmitting them to a network cloud. On the cyber (server) side, the collected data, which is aggregated and analyzed by the network server, can be used for automatic decision making and intelligent services. Since the whole idea of IoT systems is built on information gathering, efficient signal acquisition and transmission play an important role in IoT applications. Among different factors, energy consumption and computational complexity are key factors that determine the efficiency of data collection in IoT services since deployed sensors are often small, low-power embedded devices having small computational capacity and limited energy source.

Under such system requirements, Compressive Sensing [1], which is an efficient signal acquisition technology, is a suitable solution for sensor data collection in an IoT stack. The efficiency in CS devices comes from the fact that signals are sampled, discretized and compressed at the same time, at the hardware level, i.e., on the sensors. This is different from traditional sensors from which we obtain a high amount of discrete samples or raw measurements from the signal, and the compression step is conducted after the acquision step, at the software level. In CS, the signal is measured and compressed at the same time, before being registered on memory during the sampling phase. Because of this, the CS sensor requires a much smaller memory size to store temporary data, and outputs a discrete signal with a significantly lower number of measurements for storage and transmission.

Given an input signal that has been sampled and discretized $\mathbf{y} \in \mathbb{R}^I$, instead of outputing $\mathbf{y}$, a CS device outputs a compressed version of $\mathbf{y}$ by linearly projecting $\mathbf{y}$ onto a lower dimensional space as follows:

$$\mathbf{z} = \mathbf{\Phi}\mathbf{y} \tag{1}$$

where $\mathbf{z} \in \mathbb{R}^M$ denotes the measurement of the input signal and $\mathbf{\Phi} \in \mathbb{R}^{M \times I}$ denotes the projection matrix. The number of measurements obtained by a CS device is often significantly lower than the dimension of $\mathbf{y}$, i.e., $M \ll I$, so $\mathbf{\Phi}$ is a fat matrix. Φ is also referred to as the sensing operator or sensing matrix.

Here we should note that what we obtain from a traditional sensor is $\mathbf{y}$, which is often sampled at a sampling rate higher than the Nyquist rate to ensure perfect reconstruction of the input signal. Since the dimension of $\mathbf{z}$ is lower than $\mathbf{y}$, the signal is undersampled in a CS device. Although the Shanon theorem on ideal sampling specifies that a signal must be obtained at a higher rate than the Nyquist rate to guarantee perfect reconstruction, the undersampled signal outputed by a CS device can still be recovered near perfection if the input signal can be expressed with a sparse representation in some domain and the sensing matrix Φ possesses certain properties [2], [3]. These results are known as the CS theory, which is the foundation and motivation for the developments of compressive sensing and learning methods.

Although signals can be acquired at a very low cost using the CS paradigm, reversing them to the original domains is a daunting task since it involves solving an optimization problem to determine the sparse representation and the corresponding bases of the signal. Under some use cases, signal recovery is necessary and the quality of signal reconstruction plays an important role. For example in medical imaging for health expert diagnosis, the higher the resolution and fidelity of the reconstructed signal are, the better chance of distinguishing between health abnormality and noise generated from the imaging process. However, for many applications, the purpose of acquiring signals is to perform classification or regression of some kind, rather than reconstructing the original signals. For example, in forest fire detection and monitoring via unmanned aerial vehicles, the objective of forest imaging is to automatically detect and locate potential fires. In some applications that involve human such as smart buildings, the reconstruction of data should be avoided since this step can potentially disclose private information. In fact, the majority of IoT services and automation systems gather data mainly for intelligence purposes, rather than high-fidelity reconstruction.

Because of the aforementioned reasons, researchers have proposed machine learning models that are tailored to work directly with the compressed measurements obtained from CS devices without going through a proxy signal recovery step. Methodologies developped under this objective form a research topic known as Compressive Learning (CL). The early works in CL took a similar approach to CS literature, relying on random valued sensing operators and focusing on theoretical guarantees for models operating on compressed measurements [4], [5], [6], [7]. The reliance on random sensing matrices not only exempt us from the problem of joint estimation of model's parameters and the sensing operator but also allows us to take advantage of existing theoretical results on random linear projections [8]. Besides, theoretical results on perfect signal reconstruction in CS were derived from random sensing matrices. Since the ability for high-fidelity signal recovery also implies the preservation of signal content in the compressed measurement, there is an assurance that the machine learning model is estimated with the same degree of informational content in such cases.

With wide adoption of modern stochastic optimization methods in the past decade, more recent works in CL have switched from random sensing matrices to optimized ones, which are jointly estimated with the model's parameters. Although theoretical results for the end-to-end learning approach are yet to be derived, several works have demonstrated its superior performance over prior setups using random sensing operators [9], [10], [11], [12], [13], [14]. Among these end-to-end compressive learning models, Multilinear Compressive Learning (MCL) [13] is the leading solution in both inference performance as well as computational efficiency for multidimensional signals. This stems from the fact that MCL employs sensing and feature extraction modules that are natively designed to operate on tensors using multilinear operations. Since a multidimensional input signal is linearly projected along each tensor mode in MCL, the amount of computation used to compress the given signal is significantly less than the one in Eq. (1) while the multidimensional structure of the input is still retained after the projections.

Regardless of the approach, existing CL formulations require training separate model configurations for different compression rates. This usually results in long experimentation processes in order to determine a suitable trade-off between the compression rate and the inference performance when deploying a CL system. For applications that employ remote compressive sensing, a fixed compression rate for signal acquisition is undesirable. This is because the majority of modern communication standards support adaptive transmission rate to maximize throughput and minimize energy consumption as the transmission environment changes. For example, Adaptive Data Rate (ADR) scheme is a key feature of Long-Range Wide Area Networks (LoRaWAN), an IoT technology. If the signal can be acquired at an adaptive compression rate on the sensor level, hence adaptive degree of signal fidelity, which is adjusted according to the network status, we can further maximize the signal content throughput of a CL system.

In this paper, we propose a novel optimization scheme and deployment setup for MCL that enables training multiple model configurations with different compression rates in one shot. The resulting system can be used to evaluate and benchmark different compressed measurement shapes, which significantly reduces experimentation efforts to find an optimal trade-off between data compression rate and inference performance. In addition, using the proposed optimization scheme, we can obtain sensing operators that produce highly structured compressed measurements, which allow us to implement a CS device that is capable of signal acquisition at an adaptive compression rate, thereby solving the aforementioned shortcoming of existing systems.

The remaining of our paper is organized as follows. Section II reviews the MCL model and related literature. Section III provides details of our proposed optimization scheme and deployment setup. In Section IV, we provide detailed information about our experimental protocol, qualitative and quantitive analyses of the empirical results. Section V concludes our work with remarks on the implication of our contribution.

II. Related Work

Slightly related to our work is the work in [15], which studies a surrogate performance indicator that allows fast estimation of the ranking between different model configurations in MCL. In [15], the authors found out that the mean-squared error obtained during the initialization step in MCL can be used as a performance indicator since this quantity exhibits a high correlation with the final classification error. The work in [16] also bears some similarity to our work in the sense that the method is also capable of learning single neural network that can classify different measurement sizes. However, the method in [16] is different from our work since it was proposed for vector-based CL utilizing a random sensing operator, and trains the classifier via means of data augmentation. In the following, we describe the details of MCL, which are the basis of our method in Section III.

A Multilinear Compressive Learning (MCL) model [13] comprises of three elements: the Multilinear Compressive Sensing (MCS) module, the Feature Synthesis (FS) module and the task-specific neural network N.

Different from the sensing model described in Eq. (1), which only considers input signals of vector form, the Multilinear Compressive Sensing (MCS) model employed in MCL performs separate linear sensing steps along each dimension of the input, given a multidimensional signal. Thus, the sensing is executed via a set of sensing matrices, also known as separable sensing operators. More specifically, let us denote the discretized input signal and the compressed measurement obtained from the MCS module as $\mathcal{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_K}$ and $\mathcal{Z} \in \mathbb{R}^{M_1 \times \cdots \times M_K}$, respectively. In this case, $I_1 \times \cdots \times I_K$ represents the signal resolution that the sensor initially samples and discretizes. The MCS compression model is described by the following equation:

$$\mathcal{Z} = \mathcal{Y} \times_1 \mathbf{\Phi}_1 \times \cdots \times_K \mathbf{\Phi}_K \tag{2}$$

where $\{\mathbf{\Phi}_k \in \mathbb{R}^{M_k \times I_k} \,|\, k = 1, \ldots, K\}$ denotes the set of separable sensing operators and $\times_k$ denotes the mode-k product between a tensor and a matrix. Detailed description of the mode-k product can be found in [17]. Basically, this operation linearly transforms every k-th dimensional slice of the tensor using the matrix.

Here we should note that at deployment the compressive sensing step described in the above equation and also in Eq. (1) is implemented at the hardware level, on the sensing device. What we obtain from this device is $\mathbf{z}$ or $\mathcal{Z}$, the compressed measurement. Thus, the sensing step in Eq. (1) and Eq. (2) should not be viewed as a feature extraction step since they are inherent in the signal acquisition procedure of the CS device. For end-to-end CL approaches, during development and optimization, we often simulate this signal acquisition step at the software level, using the high resolution signal $\mathcal{Y}$ that was sampled above the Nyquist rate using a standard sensor in order to optimize the sensing operators.

Given the compressed measurement $\mathcal{Z}$, MCL synthesizes a high-dimensional tensor feature that is relevant for the learning task by the FS module. In order to preserve the multidimensional structure of the compressed measurement, the FS module proposed in [13] also employs a multilinear transform. However, since the FS module is implemented at the software level, usually on the computing cloud for remote sensing applications, the FS module is not constrained to the use of multilinear operations, as long as the tensor structure of the signal is preserved. For example, the authors in [14] extended the original design of the MCL model in [13] with an FS module that contains several layers of convolution and up-sampling. The choice of the FS module mainly depends on the operating power of the computing server. In this work, we investigate our optimization scheme using the original design in [13] for the FS component, which can be described by the following equation:

$$\mathcal{T} = f_{\mathsf{FS}}(\mathcal{Z}) = \mathcal{Z} \times_1 \mathbf{\Theta}_1 \times \cdots \times_K \mathbf{\Theta}_K \tag{3}$$

where $\mathcal{T} \in \mathbb{R}^{I_1 \times \cdots \times I_K}$ denotes the synthesized feature, $\{\mathbf{\Theta}_k \in \mathbb{R}^{I_k \times M_k} \,|\, k = 1, \ldots, K\}$ denotes the parameters of the FS component, and f_{FS} denotes its functional form.

Finally, $\mathcal{T}$ is fed to a task-specific neural network N to produce a prediction for the given compressed measurement. The architecture of N, which depends on the given problem at hand, is the same architecture that one would use to classify uncompressed signals, i.e., the high-resolution signal $\mathcal{Y}$. For this reason, in Eq. (3), we can see that the dimensions of $\mathcal{T}$ are similar to $\mathcal{Y}$, the high-resolution signal.

In MCL, all model's parameters are optimized in an end-to-end manner by stochastic gradient descent optimizer. Different from the conventional approach where the parameters are initialized with random values, MCL determines the initial values of each component's parameters by solving two optimization objectives: one for the MCS and FS modules and the other for the task-specific neural network. As we have mentioned previously, for methods that optimize the sensing operator, we need a labeled set of high-resolution signals, which is obtained using a standard sensor sensing at higher-than-Nyquist rate, in order to simulate the compressive sensing step during optimization. Let us denote N as the number of samples in the training set. In addition, we also denote the n-th high-resolution sample in the training set as $\mathcal{Y}_n$, its compressed measurement as $\mathcal{Z}_n$, the corresponding synthesized feature as $\mathcal{T}_n$ and the label as $\mathbf{c}_n$. The initial parameters' values of the MCS and FS modules are obtained by solving Higher Order Singular Value Decomposition (HOSVD) [18] using the set of high-resolution signals $\{\mathcal{Y}_n \,|\, n = 1, \ldots, N\}$. That is, if:

$$\mathcal{Y}_n = \mathcal{S}_n \times_1 \mathbf{U}_1 \times \cdots \times_K \mathbf{U}_K \tag{4}$$

denotes the HOSVD decomposition of $\mathcal{Y}_n$, the sensing operators are initialized by setting $\mathbf{\Phi}_k = \mathbf{U}_k^T$, and the FS parameters are initialized by setting $\mathbf{\Theta}_k = \mathbf{U}_k$. Here $\mathcal{S}_n \in \mathbb{R}^{M_1 \times \cdots \times M_K}$ denotes the tensor core that contains the singular values and $\mathbf{U}_k \in \mathbb{R}^{M_k \times I_k}$ $(k = 1, \ldots, K)$ denotes the factor matrices of the decomposition. This initialization scheme of the sensing and the feature extraction matrices helps preserve the energy of $\mathcal{Y}_n$ in $\mathcal{T}_n$.

The parameters of the task-specific neural network N is initialized by training on the high-resolution signals with labels:

$$\underset{\mathbf{\Omega}}{\text{argmin}} \sum_{n=1}^{N} L\big(f_{\mathsf{N}}(\mathcal{Y}_n), \mathbf{c}_n\big) \tag{5}$$

where $\mathbf{\Omega}$ denotes the parameters of N, and $f_{\mathsf{N}}(\mathcal{Y}_n)$ denotes its prediction, given the input $\mathcal{Y}_n$. The learning loss function is denoted by L.

III. Proposed Method

As we have mentioned in the introduction, the main motivation of our work is to develop a remote compressive learning

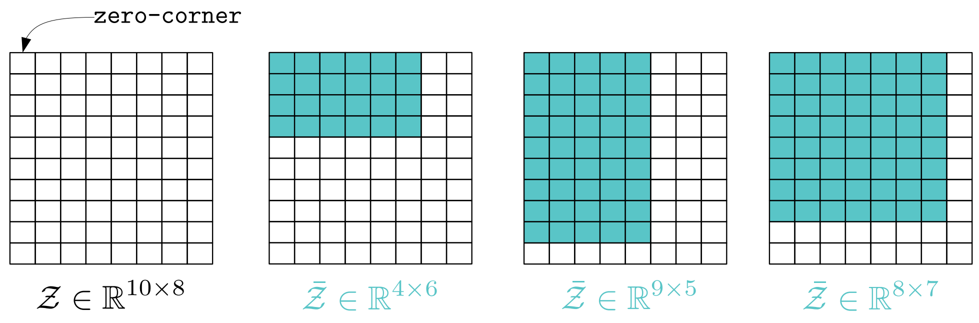

zero-corner
$\mathcal{Z} \in \mathbb{R}^{10\times 8}$
$\bar{\mathcal{Z}} \in \mathbb{R}^{4\times 6}$
$\bar{\mathcal{Z}} \in \mathbb{R}^{9\times 5}$
$\bar{\mathcal{Z}} \in \mathbb{R}^{8\times 7}$

Fig. 1. Illustration of how different compressed measurements of smaller sizes ($\bar{\mathcal{Z}}$) are constructed from $\mathcal{Z}$ from the zero-corner

system that is capable of compressive signal acquisition and learning with an adaptive compression rate. The ability to adaptively adjust the compression rate, hence the degree of signal fidelity and the amount of data transmitted for each signal sample, can significantly enhance the informational content throughput of the remote sensing and learning application. This is because in real-world scenarios, the conditions of data transmission medium, especially in wireless communication, vary from time to time, thus, the majority of communication protocols has support for adaptive transmission rates. This feature from a communication protocol enables a network to maximize its throughput while minimize the energy consumption. However, this feature alone cannot maximize the data content throughput, i.e., the amount of signal information transmitted in a period of time. For example, in video streaming services, in addition to the ability to adjust the transmission rate, a streaming server must be able to send video frames with a resolution that is adaptive to the network strength in order to ensure consistent frames per second, thus smooth view experience.

From the hardware point of view, it is infeasible to implement a MCS sensor with an adaptive compression rate through the use of multiple sets of sensing operators, each of which corresponds to a compression rate. However, with a single set of sensing operators $\{\mathbf{\Phi}_k \,|\, k = 1, \ldots, K\}$, which produces a compressed measurement $\mathcal{Z} \in \mathbb{R}^{M_1 \times \cdots \times M_K}$, we can obtain a compressed measurement of a given size $\bar{\mathcal{Z}} \in \mathbb{R}^{m_1 \times \cdots \times m_K}$ ($m_k \leq M_k, \forall k$) that corresponds to a (higher) compression rate by forming $\bar{\mathcal{Z}}$ from elements in $\mathcal{Z}$:

$$\bar{\mathcal{Z}}[i_1, \ldots, i_K] \in \mathcal{Z}, \quad \forall\, i_k \leq m_k \mid k = 1, \ldots, K \tag{6}$$

where $\bar{\mathcal{Z}}[i_1, \ldots, i_K]$ denotes the element at position $(i_1, \ldots, i_K)$ of $\bar{\mathcal{Z}}$.

In order to construct multiple instances of $\bar{\mathcal{Z}}$, each of which carries the amount of signal information approximately proportional to its size, there must be a way to determine which subset of elements in $\mathcal{Z}$ that corresponds to such level of information. Alternatively, we can optimize the set of sensing operators $\{\mathbf{\Phi}_k \,|\, k = 1, \ldots, K\}$ in such a way that the resulting $\mathcal{Z}$ possesses a predefined semantic structure. In this work, we adopt the latter approach and propose an optimization scheme that induces a predefined semantic structure in $\mathcal{Z}$ that allows us to generate multiple compressed measurements having smaller sizes than $\mathcal{Z}$ by sampling from $\mathcal{Z}$.

Specifically, we aim to learn a set of sensing operators that results in $\mathcal{Z}$ such that elements in $\mathcal{Z}$, which carry the most relevant signal information for the learning task, concentrate around the zero-corner of $\mathcal{Z}$, i.e., the corner at position $(1, \ldots, 1)$. Furthermore, the elements are arranged according to their importance, with elements closer to position $(1, \ldots, 1)$ are more relevant.

With $\mathcal{Z}$ having the aforementioned structure, a another compressed measurement $\bar{\mathcal{Z}}$, which correponds to a higher compression rate, can be obtained by taking the corresponding sub-tensor of the same size from the zero-corner of $\mathcal{Z}$, that is:

$$\begin{aligned} \bar{\mathcal{Z}}[i_1, \ldots, i_K] &= \mathcal{Z}[i_1, \ldots, i_K] \\ \forall\, i_k &\leq m_k \mid k = 1, \ldots, K \end{aligned} \tag{7}$$

The construction of $\bar{\mathcal{Z}}$ is illustrated in Figure 1. One feature of the proposed semantic structure for $\mathcal{Z}$ is the computational efficiency. Since this structure allows the construction of $\bar{\mathcal{Z}}$ from contiguous elements of $\mathcal{Z}$, the indexing operation to create $\mathcal{Z}$ only requires accessing contiguous memory locations, which is more hardware-friendly compared to a set of noncontiguous indices.

Here we should note that $\bar{\mathcal{Z}}$ is constructed on the client side, before being transmitted to the computing server. On the server side, where the FS module and task-specific neural network N are implemented, to make predictions with incoming compressed measurements of different sizes using a single instance of the FS module and N, the FS module must be able to handle variable-size inputs. A simple solution to this requirement is to set the input size of the FS module to the maximum size of incoming compressed measurements, i.e., the size of $\mathcal{Z}$, and the incoming compressed measurements are padded appropriately with zeros to form tensors of a fixed

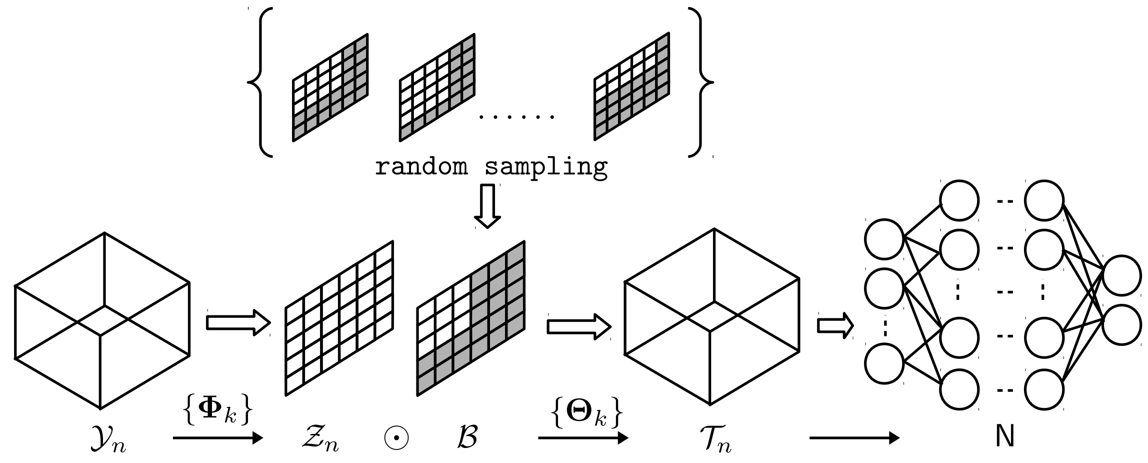


Fig. 2. Illustration of the proposed training method with stochastic binary mask $\mathcal{B}$

size.

To this end, we have defined the following criteria for optimizing a remote MCL system: (i) the sensing step produces $\mathcal{Z}$ that has the proposed semantic structure, and (ii) the server side utilizes a single model instance to make predictions with variable-size compressed measurements. In order to satisfy both criteria using end-to-end training with stochastic gradient descent, we propose to stochastically simulate the effect of variable compression rates via the means of stochastic structural dropout. Dropout [19], which randomly zeroes out intermediate representations in a neural network during optimization, is a regularization technique. This technique can be considered as training a virtual ensemble of sub-networks within the main model, thus, is effective at reducing overfitting. In our case, we use dropout masks having predefined structures not only to train sub-networks that correspond to different compression rates but also to enforce a semantic structure in $\mathcal{Z}$. More specifically, after the initialization steps of MCL described in Section II, a stochastic gradient descent optimizer is used to train all components in MCL with the following objective:

$$\underset{\{\mathbf{\Phi}_k\},\{\mathbf{\Theta}_k\},\mathbf{\Omega}}{\text{argmin}} \quad \sum_{n=1}^{N} L\big(f_{\mathsf{N}}(f_{\mathsf{FS}}(\mathcal{Z}_n \odot \mathcal{B})), \mathbf{c}_n\big) \tag{8}$$

where $\odot$ denotes the element-wise multiplication operator. $\mathcal{B} \in \mathbb{R}^{M_1 \times \cdots \times M_K}$ is a random binary matrix having the following structure:

$$\mathcal{B}[i_1, \dots, i_K] = \begin{cases} 1 & \text{if } i_k \leq m_k^{(r)} \\ 0 & \text{else} \end{cases} \quad \forall k = 1, \dots, K \tag{9}$$

where $m_k^{(r)}$ denotes an integer value randomly drawn from the set $\{M_k^{(\min)}, M_k^{(\min)} + 1, \dots, M_k\}$ for all $k = 1, \dots, K$, with $M_k^{(\min)}$ and M_k denote predefined minimum and maximum values for a given dimension of the compressed measurement.

The proposed training technique with stochastic binary mask $\mathcal{B}$ is illustrated in Figure 2. By applying the binary mask $\mathcal{B}$ to $\mathcal{Z}_n$, we implicitly train the MCL model to perform sensing and learning with a compressed measurement of size $m_1^{(r)} \times \cdots \times m_k^{(r)} \cdots \times m_K^{(r)}$, which is randomly defined during stochastic optimization. To do so, the value of $\mathcal{B}$ is changed in every forward pass for every training sample during stochastic optimization.

Finally, we should remark that in order to implement the adaptive compression rate feature, the server side is not constrained to only use a single instance of the FS module and task-specific neural network. While the client side, i.e., the sensing device has critical limitation in terms of computational power and especially energy, there is no inherent limitation of the computing server. In case there is enough computational power, we can increase the performance of the whole system by running multiple FS modules and task-specific neural networks to make predictions given different compressed measurement sizes. That is, after optimizing Eq. (8), we can fix the sensing operators and only finetune the parameters of server side's components (FS and N) for each compression rate. In deployment, on the sensing device, we still implement a single set of sensing operators, and compressed measurements of different sizes are constructed adaptively with sub-tensors of the output of the sensing device as described previously. On the server side, based on the size of the received compressed measurement, prediction is generated with the corresponding FS and N modules. In Section IV, we will show that this approach can significantly enhance the overall performance of the whole system.

TABLE I
TEST ACCURACY (%) ON CIFAR10

	single -rate	one-shot	
		baseline	adaptive -rate
$4 \times 6 \times 2$	64.92±00.89	19.48±02.69	61.08±00.30
$6 \times 4 \times 2$	64.83±00.21	20.16±00.74	61.25±00.37
$6 \times 8 \times 1$	62.87±00.54	23.27±02.78	61.28±00.13
$8 \times 6 \times 1$	62.40±00.23	23.25±00.56	61.41±00.25
$4 \times 7 \times 2$	66.28±00.16	20.34±03.00	62.92±00.16
$7 \times 4 \times 2$	66.85±00.55	21.38±00.14	63.36±00.21
$7 \times 8 \times 1$	65.02±00.17	26.75±02.89	63.46±00.52
$8 \times 7 \times 1$	64.87±00.13	26.54±01.90	63.86±00.65
$6 \times 6 \times 2$	69.87±00.20	26.42±04.00	68.98±00.15
$8 \times 9 \times 1$	68.17±00.27	33.93±02.57	67.28±00.34
$9 \times 8 \times 1$	67.69±00.41	33.62±02.89	67.52±00.14
$12 \times 6 \times 1$	67.29±00.25	25.91±00.63	65.72±00.23
$9 \times 4 \times 2$	69.20±00.33	22.88±02.03	66.16±00.25
$10 \times 10 \times 1$	71.45±00.20	47.19±01.61	71.42±00.28
$10 \times 5 \times 2$	73.66±00.77	29.83±00.91	72.57±00.48
average	67.02	26.73	65.22
$\sum$epochs	3150	210	210

TABLE II
TEST ACCURACY (%) ON CIFAR100

	single -rate	one-shot	
		baseline	adaptive -rate
$4 \times 6 \times 2$	36.13±00.88	03.33±01.44	33.05±00.71
$6 \times 4 \times 2$	36.69±00.18	03.64±00.91	35.77±00.36
$6 \times 8 \times 1$	30.84±00.26	04.96±00.87	30.53±00.92
$8 \times 6 \times 1$	30.93±00.33	04.37±01.39	30.92±00.08
$4 \times 7 \times 2$	38.26±00.91	04.10±01.47	35.44±01.26
$7 \times 4 \times 2$	38.19±00.27	04.36±02.14	37.66±00.39
$7 \times 8 \times 1$	32.94±00.29	06.22±00.92	32.78±00.16
$8 \times 7 \times 1$	32.65±00.15	05.93±01.64	33.15±00.03
$6 \times 6 \times 2$	39.67±01.42	05.19±02.25	41.08±00.20
$8 \times 9 \times 1$	35.63±00.17	09.45±00.86	36.52±00.13
$9 \times 8 \times 1$	35.63±01.36	09.22±00.87	36.04±00.10
$12 \times 6 \times 1$	34.82±00.30	07.36±01.74	34.59±00.19
$9 \times 4 \times 2$	40.73±01.02	05.40±02.12	40.43±00.15
$10 \times 10 \times 1$	39.82±00.89	16.54±01.37	40.34±00.14
$10 \times 5 \times 2$	44.82±00.53	07.05±00.85	44.86±00.24
average	36.52	06.47	36.21
$\sum$epochs	3150	210	210

IV. EXPERIMENTS

This section provides detailed information about the empirical analysis that we conducted to benchmark our training scheme for MCL models. Information about the datasets and experimental protocol are presented first, followed by the experimental results and discussions.

A. Datasets and Experiment Protocol

Our experimental simulation was developed for object classification and face recognition tasks. Object and face recognition are necessary features in smart buildings and surveillance systems. As we have mentioned before, in order to train end-to-end compressive learning models, we need labeled data that has been collected by standard sensors. For this reason, we used CIFAR [20] and CelebA datasets [21] in our experiments, both of which are publicly available. A brief description of the datasets and our data split are provided below:

- CIFAR dataset [20] is an RGB image dataset, which contains thumbnail-size images of resolution 32×32 pixels. The whole dataset contains 60K images that are divided into train (50K) and test set (10K). The dataset contains two label sets, each of which has 10 and 100 classes. We refer to the two versions as CIFAR10 and CIFAR100. In order to perform proper validation, we randomly selected 5K images from the training set to form the validation set. Images in different classes are uniformly distributed in both CIFAR10 and CIFAR100.
- CelebA [21] is a large-scale face attributes dataset with more than 200K images of about $10K$ different people at varying resolutions. For this dataset, we followed the same experimental setup as in [13] and used a subset of 100 people to train and benchmark the performance in face recognition. The training, validation and test set contain 7063, 2373 and 2400 images, respectively. All images were resized to a fixed resolution, 32×32.

In our experiments, we adopted the same task-specific neural network architecture that was used in the experiments of [13], namely the AllCNN architecture proposed by [22]. AllCNN is a feed-forward architecture that contains only convolution layers, without any residual connection. For the details of AllCNN network, we refer the readers to [13].

For stochastic gradient descent optimization, we used ADAM optimizer [23] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. All models were trained for 230 epochs with an initial learning rate of 0.001, which is reduced by a factor of 10 at epoch 51 and 190, respectively. Weight decay of magnitude 0.00005 was used for regularization. The input pixel values were scaled into the range $[0, 1]$, and we performed a simple data augmentation technique during training by random horizontal flipping and random shifting by 4 pixels in both horizontal and vertical axes. For every experiment configuration, we repeated five times and reported the mean and standard deviation of accuracy measured on the test set.

B. Results

Since both CIFAR10, CIFAR100 and CelebA datasets were set to the same resolution, the size of $\mathcal{Y}_n$ is $32 \times 32 \times 3$ in all of the experiments. In the following, we will denote the results produced by the original training method proposed in

TABLE III
TEST ACCURACY (%) ON CELEBA

	single-rate	one-shot	
		baseline	adaptive-rate
$4 \times 6 \times 2$	49.12±00.17	06.42±00.17	45.60±00.31
$6 \times 4 \times 2$	53.38±00.13	08.02±00.52	51.37±00.17
$6 \times 8 \times 1$	61.00±00.17	09.65±02.66	57.60±00.48
$8 \times 6 \times 1$	63.38±00.88	16.60±00.52	62.12±00.54
$4 \times 7 \times 2$	51.79±00.24	07.06±00.12	49.33±00.74
$7 \times 4 \times 2$	60.17±00.12	11.42±00.13	56.56±00.35
$7 \times 8 \times 1$	68.08±00.37	16.31±01.31	64.21±00.25
$8 \times 7 \times 1$	68.75±00.82	17.58±00.12	64.77±00.56
$6 \times 6 \times 2$	60.40±01.54	13.17±01.37	58.00±00.23
$8 \times 9 \times 1$	73.06±00.19	23.50±00.58	69.44±00.17
$9 \times 8 \times 1$	74.40±00.52	28.35±00.77	71.31±00.39
$12 \times 6 \times 1$	74.48±00.23	30.92±01.46	72.23±00.29
$9 \times 4 \times 2$	66.40±00.19	17.02±00.15	64.44±00.35
$10 \times 10 \times 1$	80.00±00.21	44.71±01.88	76.75±00.29
$10 \times 5 \times 2$	72.92±00.25	22.73±02.02	70.00±00.41
average	65.16	18.23	62.25
$\sum$epochs	3150	210	210

TABLE IV
FINETUNING PERFORMANCE OF ADAPTIVE-RATE$\star$ ON CIFAR10

	single-rate	adaptive-rate	adaptive-rate$\star$
$4 \times 6 \times 2$	64.92±00.89	61.08±00.30	65.22±00.20
$6 \times 4 \times 2$	64.83±00.21	61.25±00.37	65.48±00.89
$6 \times 8 \times 1$	62.87±00.54	61.28±00.13	62.97±00.44
$8 \times 6 \times 1$	62.40±00.23	61.41±00.25	62.73±00.29
$4 \times 7 \times 2$	66.28±00.16	62.92±00.16	67.11±00.78
$7 \times 4 \times 2$	66.85±00.55	63.36±00.21	67.74±00.41
$7 \times 8 \times 1$	65.02±00.17	63.46±00.52	65.22±00.45
$8 \times 7 \times 1$	64.87±00.13	63.86±00.65	65.09±00.79
$6 \times 6 \times 2$	69.87±00.20	68.98±00.15	70.62±00.53
$8 \times 9 \times 1$	68.17±00.27	67.28±00.34	68.61±00.41
$9 \times 8 \times 1$	67.69±00.41	67.52±00.14	68.45±00.82
$12 \times 6 \times 1$	67.29±00.25	65.72±00.23	67.76±00.43
$9 \times 4 \times 2$	69.20±00.33	66.16±00.25	70.35±00.45
$10 \times 10 \times 1$	71.45±00.20	71.42±00.28	72.58±00.91
$10 \times 5 \times 2$	73.66±00.77	72.57±00.48	74.63±00.75
average	67.02	65.22	67.64
$\sum$epochs	3150	210	660

[13] as `single-rate`, and our one-shot training method as `adaptive-rate`.

For the `single-rate` training, we trained different models for the following compressed measurement sizes: $4\times6\times2$, $6\times4\times2$, $6\times8\times1$, $8\times6\times1$; $4\times7\times2$, $7\times4\times2$, $7\times8\times1$, $8\times7\times1$; $6\times6\times2$, $8\times9\times1$, $9\times8\times1$, $12\times6\times1$, $9\times4\times2$; $10\times10\times1$, $10\times5\times2$.

For our `adaptive-rate` training method, with each dataset, we only trained one model with the maximum and minimum dimensions of the compressed measurements set to $15 \times 15 \times 2$ and $4 \times 4 \times 1$. That is, the size of $\mathcal{Z}_n$ in Eq. (8) is $15 \times 15 \times 2$, and $M_1^{(min)} = M_2^{(min)} = 4$ and $M_3^{(min)} = 1$ when sampling $\mathcal{B}$ in Eq. (9). After training, we simply evaluated this model with different compressed measurement sizes that were used to train the `single-rate` models.

In addition, to demonstrate the effectiveness of the stochastic mask $\mathcal{B}$, we also train a MCL model with the compressed measurement of size $15 \times 15 \times 2$, using the original training method in [13]. This model, denoted as `baseline`, is then used to evaluate the target set of compressed measurements mentioned above, using the same evaluation procedure as in `adaptive-rate` models. Thus, `baseline` and our model (`adaptive-rate`) have the same setup that represents a one-shot training setting in which one model was trained for multiple compressed measurement sizes.

The results for CIFAR10, CIFAR100 and CelebA datasets are shown in Table I, II and III, respectively. The average accuracy of each method and the total number of epochs used to train all configurations are shown at the bottom section. In addition, the results are shown in groups according to the compression rate, i.e., compressed measurement sizes that lead to the same compression rate are grouped together.

From Table I, II and III, it is obvious that the results obtained from our method (`adaptive-rate`) are significantly better than the `baseline` method. In fact, from the results of `baseline`, we can see that without any modification to the original training algorithm, the model trained with a large compressed measurement size ($15\times15\times2$ in our case) cannot be used for other configurations with higher compression rates (smaller measurement sizes). The large variations between different runs of the `baseline` indicate that there's no semantic structure existing in the sub-tensors of the default compressed measurement, i.e., the compressed measurement of the maximum size. On the other hand, the results from `adaptive-rate` are more consistent between different experiment runs. This means that from the default compressed measurement of size $15 \times 15 \times 2$, we can directly generate compressed measurements of smaller sizes with consistent performances, indicating the existence of a semantic structure between elements in the compressed measurement trained by the proposed method.

Regarding the comparison with the conventional `single-rate` training method, on average, `adaptive-rate` achieved close performance on CIFAR100 dataset, and performed slightly less accurate on CIFAR10 and CelebA datasets with an average of 1.8% and 2.91% performance degradation, respectively. However, the reductions in accuracy in `adaptive-rate` are compensated with significant computational gains from the experimentation process. For `single-rate` training,

TABLE V
FINETUNING PERFORMANCE OF ADAPTIVE-RATE⋆ ON CIFAR100

	single -rate	adaptive -rate	adaptive -rate⋆
$4 \times 6 \times 2$	36.13±00.88	33.05±00.71	37.27±01.03
$6 \times 4 \times 2$	36.69±00.18	35.77±00.36	38.25±00.95
$6 \times 8 \times 1$	30.84±00.26	30.53±00.92	31.97±00.33
$8 \times 6 \times 1$	30.93±00.33	30.92±00.08	32.10±01.25
$4 \times 7 \times 2$	38.26±00.91	35.44±01.26	39.70±00.76
$7 \times 4 \times 2$	38.19±00.27	37.66±00.39	40.30±00.23
$7 \times 8 \times 1$	32.94±00.29	32.78±00.16	33.88±00.43
$8 \times 7 \times 1$	32.65±00.15	33.15±00.03	34.04±01.09
$6 \times 6 \times 2$	39.67±01.42	41.08±00.20	42.71±00.74
$8 \times 9 \times 1$	35.63±00.17	36.52±00.13	36.97±00.29
$9 \times 8 \times 1$	35.63±01.36	36.04±00.10	36.87±00.17
$12 \times 6 \times 1$	34.82±00.30	34.59±00.19	36.14±00.62
$9 \times 4 \times 2$	40.73±01.02	40.43±00.15	42.94±00.77
$10 \times 10 \times 1$	39.82±00.89	40.34±00.14	40.88±00.31
$10 \times 5 \times 2$	44.82±00.53	44.86±00.24	46.20±00.13
average	36.52	36.21	38.01
$\sum$epochs	3150	210	660

TABLE VI
FINETUNING PERFORMANCE OF ADAPTIVE-RATE⋆ ON CELEBA

	single -rate	adaptive -rate	adaptive -rate⋆
$4 \times 6 \times 2$	49.12±00.17	45.60±00.31	49.92±00.42
$6 \times 4 \times 2$	53.38±00.13	51.37±00.17	53.85±00.15
$6 \times 8 \times 1$	61.00±00.17	57.60±00.48	61.15±00.19
$8 \times 6 \times 1$	63.38±00.88	62.12±00.54	65.04±01.19
$4 \times 7 \times 2$	51.79±00.24	49.33±00.74	52.67±00.49
$7 \times 4 \times 2$	60.17±00.12	56.56±00.35	61.50±00.47
$7 \times 8 \times 1$	68.08±00.37	64.21±00.25	68.17±00.68
$8 \times 7 \times 1$	68.75±00.82	64.77±00.56	68.63±00.33
$6 \times 6 \times 2$	60.40±01.54	58.00±00.23	62.73±01.31
$8 \times 9 \times 1$	73.06±00.19	69.44±00.17	73.04±00.37
$9 \times 8 \times 1$	74.40±00.52	71.31±00.39	74.15±00.19
$12 \times 6 \times 1$	74.48±00.23	72.23±00.29	74.48±00.15
$9 \times 4 \times 2$	66.40±00.19	64.44±00.35	67.67±01.04
$10 \times 10 \times 1$	80.00±00.21	76.75±00.29	79.44±00.96
$10 \times 5 \times 2$	72.92±00.25	70.00±00.41	71.12±00.38
average	65.16	62.25	65.57
$\sum$epochs	3150	210	660

experimenting with 15 different compressed measurements required us to run 3150 epochs of gradient updates. On the other hand, with `adaptive-rate` training, we needed to train only a single model to evaluate all 15 different configurations. As the number of compressed measurement configurations increases, the saving factor increases when using `adaptive-rate` instead of `single-rate` training.

Another benefit from using the proposed training technique is that we can also use `adaptive-rate` training technique to quickly identify the best configurations given a compression rate. For example, in CelebA dataset, even though with the same compression rate, $8 \times 6 \times 1$ and $6 \times 8 \times 1$ yield much better accuracy than $4 \times 6 \times 2$ and $6 \times 4 \times 2$ when each configuration is optimized separately. Using `adaptive-rate` training, we can also observe this phenomenon, using only 25% computation compared to the former case. On CIFAR100, with the same compression rate, the ranking is reversed ($4 \times 6 \times 2$ and $6 \times 4 \times 2$ perform much better than $8 \times 6 \times 1$ and $6 \times 8 \times 1$), which is also seen from the results of `adaptive-rate`.

Up until now, we have shown that with less than 3% of accuracy drop, we can speedily train and deploy a single model instance to achieve the adaptive compression rate feature. However, as we have described at the end of Section III, the server side is not constrained to run a single instance of FS module and task-specific neural network. That is, after optimizing a single model instance using `adaptive-rate` method, we can fix the sensing operators and finetune the FS module and task-specific neural network for each compressed measurement size. To demonstrate this, we performed finetuning for 30 epochs for each compressed measurement size. The results, denoted as `adaptive-rate⋆`, are shown in Table IV, V, and VI. It is clear from these tables that by allowing separate model instances on the server side with `adaptive-rate⋆`, we obtained noticeable enhancements in the overall performances, which are on-par with the performances obtained from `single-rate` training method on CIFAR10 and CelebA datasets, and clearly better on CIFAR100 dataset. Although the training complexity (total number of epochs) using `single-rate⋆` is higher than using a single model instance, it is still far below what is required for `single-rate`.

V. CONCLUSIONS

In this paper, we proposed a novel training method and practical implementation of Multilinear Compressive Learning models that are capable of compressive signal acquisition and prediction with an adaptive compression rate. By enabling such a feature in remote compressive learning systems, the degree of signal fidelity, hence amount of data transmitted for each sample, can be adjusted according to the transmission condition. This can result in major improvements of informational content throughput of a remote sensing and learning application. Our empirical evaluation of the proposed training technique showed that with a few percents of accuracy drop, we can save a significant amount of computation during the training phase. Furthermore, when the server side can handle multiple model instances, we can achieve on-par performances compared to the standard setup while still having the adaptive compression rate feature on the sensing devices and requiring less computation for training. Our work, as well as future works in adaptive sensing and learning systems will complement the developments in IoT technologies, creating a world of intelligent services and systems that make everyday life effortless and comfortable.

VI. Acknowledgement

This project has received funding from the European Unions 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. 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*, 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. Reboredo, F. Renna, R. Calderbank, and M. R. Rodrigues, "Compressive classification," in *2013 IEEE International Symposium on Information Theory*, pp. 674–678, IEEE, 2013.

[8] R. G. Baraniuk and M. B. Wakin, "Random projections of smooth manifolds," *Foundations of computational mathematics*, vol. 9, no. 1, pp. 51–77, 2009.

[9] A. Adler, M. Elad, and M. Zibulevsky, "Compressed learning: A deep neural network approach," *arXiv preprint arXiv:1610.09615*, 2016.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] D. T. Tran, M. Gabbouj, and A. Iosifidis, "Multilinear compressive learning with prior knowledge," *arXiv preprint arXiv:2002.07203*, 2020.

[15] D. T. Tran, M. Gabbouj, and A. Iosifidis, "Performance indicator in multilinear compressive learning," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1822–1828, IEEE, 2020.

[16] Y. Xu and K. F. Kelly, "Compressed domain image classification using a multi-rate neural network," *arXiv preprint arXiv:1901.09983*, 2019.

[17] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[18] 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.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. 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.

[20] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," tech. rep., Citeseer, 2009.

[21] 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.

[22] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

8.4 Continual 3D Convolutional Neural Networks for Real-time Processing of Videos

The appended paper follows.

Continual 3D Convolutional Neural Networks for Real-time Processing of Videos

Lukas Hedegaard Alexandros Iosifidis
Department of Electrical and Computer Engineering, Aarhus University, Denmark
{lhm, ai}@ece.au.dk

Abstract

This paper introduces Continual 3D Convolutional Neural Networks (Co3D CNNs), a new computational formulation of spatio-temporal 3D CNNs, in which videos are processed frame-by-frame rather than by clip. In online processing tasks demanding frame-wise predictions, Co3D CNNs dispense with the computational redundancies of regular 3D CNNs, namely the repeated convolutions over frames, which appear in multiple clips. While yielding an order of magnitude in computational savings, Co3D CNNs have memory requirements comparable with that of corresponding regular 3D CNNs and are less affected by changes in the size of the temporal receptive field. We show that Continual 3D CNNs initialised on the weights from preexisting state-of-the-art video recognition models reduce the floating point operations for frame-wise computations by 10.0−12.4× while improving accuracy on Kinetics-400 by 2.3−3.8%. Moreover, we investigate the transient start-up response of Co3D CNNs and perform an extensive benchmark of online processing speed as well as accuracy for publicly available state-of-the-art 3D CNNs on modern hardware.

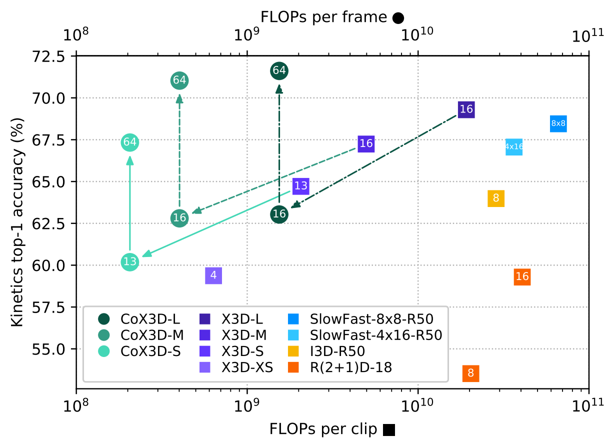


Figure 1: **Accuracy/complexity trade-off** for Continual X3D (*Co*X3D) and recent state-of-the-art 3D CNNs on Kinetics-400 using 1-clip/frame testing. For regular 3D CNNs, the FLOPs per *clip* ■ are noted, while the FLOPs per *frame* ● are shown for the Continual 3D CNNs. The *Co*X3D models used the weights from the X3D models without further fine-tuning. The global average pool size for the network is noted in each point. The diagonal and vertical arrows indicate respectively a transfer from regular to Continual 3D CNN and an extension of receptive field.

1. Introduction

Through the availability of large-scale open-source datasets such as ImageNet [30] and Kinetics [21, 2], deep, over-parameterized Convolutional Neural Networks (CNNs) have achieved impressive results in the field of computer vision. In video recognition specifically, 3D CNNs have lead the recent state-of-the-art [3, 36, 8, 7], generally outperforming RNN-based approaches [5, 17]. Despite their success in competitions and benchmarks where only prediction quality is evaluated, their computational cost and processing time remains a challenge to the deployment in many real-life use-cases with energy constraints and/or real-time needs. To combat this general issue, multiple approaches have been explored. These include computationally efficient architectures for image [13, 42, 34] and video recognition [24, 7, 43], pruning of network weights [4, 10, 11], knowledge distillation [12, 41, 29], and network quantisation [15, 1, 9].

The contribution in this paper is complementary to all of the above. It exploits the computational redundancies in the application of regular spatio-temporal 3D CNNs to a continual video stream in a sliding window fashion (Fig. 2). This redundancy was also explored recently [22, 31] using specialised architectures. However, these are not weight-compatible with regular 3D CNNs. In this work, we present a weight-compatible reformulation of the 3D CNN and its components as a *Continual* 3D Convolutional Neural

Network (*Co*3D CNN). *Co*3D CNNs process input videos frame-by-frame rather than clip-wise and can reuse the weights of regular 3D CNNs, producing identical outputs for networks without temporal zero-padding. To explore the characteristics of Continual CNNs and validate their efficacy, we perform conversions from three state-of-the-art 3D CNNs, each at different points on the accuracy/speed pareto-frontier, and evaluate their frame-wise performance. While there is a slight reduction in accuracy in the conversion due to zero-padding in the regular 3D CNNs, a simple network modification of extending the temporal receptive field recovers and improves the accuracy significantly *without* any fine-tuning at a negligible increase in computational cost. Furthermore, we measure the transient network response at start-up, and perform extensive benchmarking on common hardware and embedded devices to gauge the expected inference speeds for real-life scenarios. We find that Continual 3D CNNs offer a $10.0-12.4\times$ reduction in floating point operations (FLOPs), a $5.9-9.2\times$ speed increase on CPU and a $2.8-3.8\%$ accuracy improvement on Kinetics-400 over regular 3D CNNs.

2. Related Works

2.1. 3D CNNs for video recognition

Convolutional Neural Networks with spatio-temporal 3D kernels may be considered the natural extension from 2D CNNs for image recognition to CNNs for video recognition. Although they did not surpass their 2D CNN + RNN competitors [5, 17] initially [16, 20, 35], arguably due to a high parameter count and insufficient dataset size, 3D CNNs have produced state-of-the-art results [3, 36, 8] since the Kinetics dataset [21] was introduced. Yet, high accuracy comes with a high computational cost, which is prohibitive to many real-life use cases.

In image recognition, efficient architectures such as MobileNet [13], ShuffleNet [42], and EfficientNet [34] attained improved accuracy-complexity trade-offs. These architectures were extended to 3D-convolutional versions 3D-MobileNet [24], 3D-ShuffleNet [24] and X3D [7] (≈3D-EfficientNet) with similarly improved pareto-frontier in video-recognition tasks. While these efficient 3D CNNs work well for offline processing of videos, they are limited in the context of online processing, where we wish to make multiple predictions per second; real-time processing rates can still be achieved only at the price of severely reduced accuracy. At the heart of the limitation of 3D CNNs is the restriction that they must process a whole "clip" (spatio-temporal volume) at a time. When predictions are needed for each frame, this imposes a significant overhead due to repeated computations. In our work, we overcome this challenge by introducing an alternative computational scheme for spatio-temporal convolutions, -pooling, and -residuals, which lets us compute 3D CNN outputs frame-wise (continually) and dispose of the redundancies produced by regular 3D CNNs.

2.2. Architectures for online video recognition

In vision tasks with a temporal dimension, a straightforward and well-explored approach [5, 17, 18, 32] is to let each frame pass through a 2D CNN trained on ImageNet in one stream alongside a second stream of Optical Flow [6]. The outputs can then be integrated using a recurrent network to model long-term temporal dependencies [5, 17] or used for spatio-temporal action detection tasks [18, 32]. It has the advantage that it requires no network modification for deployment in online-processing scenarios, lends itself to caching [40], and is free of the computational redundancies experienced in 3D CNNs. It has the disadvantage that the inclusion of optical flow introduces a large computational overhead.

Another approach is to utilise 3D CNNs for spatio-temporal feature extraction. In [27], spatio-temporal features from non-overlaping clips are used to train a recurrent network for hand gesture recognition. In [23], a 3D CNN operating on a sliding window of the input performs spatio-temporal action detection. These 3D CNN-based methods have the disadvantage of either not producing predictions for each input frame [27] or suffering from redundant computations from overlapping input clips [23].

Exploring modifications of the spatio-temporal 3D convolution operating frame by frame, the Recurrent Convolutional Unit (RCU) [31] replaces the 3D convolution by aggregating a spatial 2D convolution over the current input with a 1D convolution over the prior output. Closest to our work are Dissected 3D CNNs [22] (D3D). They define an architecture that caches the $1 \times n_H \times n_W$ frame-level features of network residual connections and concatenates them with the corresponding features in the next frame. This produces intermediary spatio-temporal features of shape $2 \times n_H \times n_W$, which become inputs to a block of convolutional layers. With kernel sizes $k_T \times k_H \times k_W$ of $2 \times 3 \times 3$ and $1 \times 3 \times 3$, the block produces features of shape $1 \times n_H \times n_W$ to be cached once again. Here, n denotes the feature map size and k the kernel size and subscripts T, H, and W denote the time, height, and width dimensions. An LSTM is then used for late spatio-temporal modelling prior to prediction. Like the Continual Convolutions that we propose, both RCU and D3D are causal and operate frame-by-frame. However, they are incompatible with pre-trained 3D CNNs, and must be trained from scratch. While the idea of caching features is also central to our work, we reformulate spatio-temporal 3D convolutions in a one-to-one compatible manner, allowing us to reuse existing model weights. In fact, all of the results presented in this paper were achieved without any training!

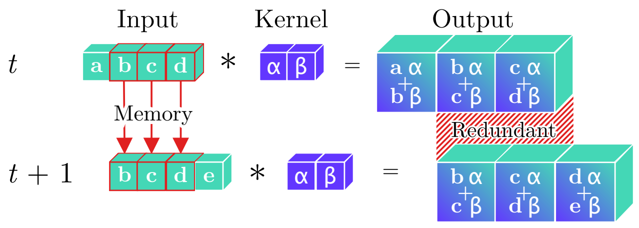


Figure 2: **Redundant computations** for a temporal convolutional layer during online processing of video clips, as illustrated by the repeated convolution of inputs (green $\mathbf{b}, \mathbf{c}, \mathbf{d}$) with a kernel (blue α, β) in the temporal dimension. Moreover, prior inputs $(\mathbf{b}, \mathbf{c}, \mathbf{d})$ must be stored between time-steps for online processing tasks.

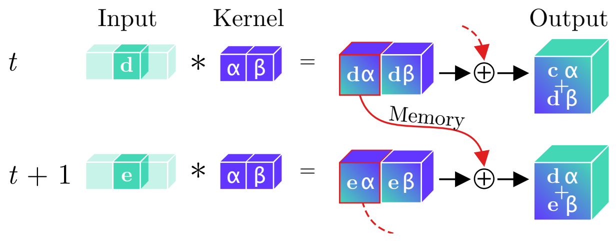


Figure 3: **Continual Convolution**. An input (green $\mathbf{d}$ or $\mathbf{e}$) is convolved with a kernel (blue α, β). The intermediary feature-maps corresponding to all but the last temporal position are stored, while the last feature map and prior memory are summed to produce the resulting output. For a continual stream of inputs, Continual Convolutions produce identical outputs to regular convolutions.

3. Continual Convolutional Neural Networks

3.1. Regular 3D-convolutions lead to redundancy

Currently, the best performing architectures (X3D [7], SlowFast [8], etc.) employ variations on 3D convolutions as their main building block and perform predictions for a spatio-temporal input volume (video-clip). These architectures achieve high accuracy with reasonable computational cost for predictions on clips in the offline setting. They are, however, ill-suited for online video classification, where the input is a continual stream of video frames and a class prediction is needed for each frame. For regular 3D CNNs processing clips of m_T frames to be used in this context, prior $m_T - 1$ input frames need to be stored between temporal time-steps and assembled to form a new video-clip when the next frame is sampled. This is illustrated in Fig. 2.

Recall the computational complexity for a 3D convolution:

$$\mathcal{O}([k_H \cdot k_W \cdot k_T + b] \cdot c_I \cdot c_O \cdot n_H \cdot n_W \cdot n_T), \tag{1}$$

where k denotes the kernel size, T, H, and W are time, height, and width dimension subscripts, $b \in \{0, 1\}$ indicates whether bias is used, and c_I and c_O are the number of input and output channels. The size of the output feature map is $n = (m + 2p - d \cdot (k - 1) - 1)/s + 1$ for an input of size m and a convolution with padding p, dilation d, and stride s. During online processing, every frame in the continual video-stream will be processed n_T times (once for each position in the clip), leading to a redundancy proportional with $n_T - 1$. Moreover, the memory-overhead of storing prior input frames is

$$\mathcal{O}(c_I \cdot m_H \cdot m_W \cdot [m_T - 1])), \tag{2}$$

and during inference the network has to transiently store feature-maps of size

$$\mathcal{O}(c_O \cdot n_H \cdot n_W \cdot n_T). \tag{3}$$

3.2. Continual Convolutions

We can remedy the issue described in Section 3.1 by employing an alternative sequence of computational steps. In essence, we reformulate the repeated convolution of a (3D) kernel with a (3D) input-clip that continually shifts along the temporal dimension as a *Continual* Convolution (*Co*Conv), where all convolution computations (bar the final sum) for the (3D) kernel with each (2D) input-frame are performed in one time-step. Intermediary results are stored as states to be used in subsequent steps, while previous and current results are summed up to produce the output. The process for a 1D input and kernel, which corresponds to the regular convolution in Fig. 2, is illustrated in Fig. 3. In general, this scheme can be applied for online-processing of any ND input, where one dimension is a temporal continual stream. Continual Convolutions are causal [37] with no information leaking from future to past and can be efficiently implemented by zero-padding the input frame along the temporal dimension with $p = \text{floor}(k/2)$. Python-style pseudo-code of the implementation is shown in Listing 1.

In terms of computational cost, we can now perform frame-by-frame computations much more efficiently than a regular 3D convolution. The complexity of processing a frame becomes:

$$\mathcal{O}([k_H \cdot k_W \cdot k_T + b] \cdot c_I \cdot c_O \cdot n_H \cdot n_W). \tag{4}$$

This reduction in computational complexity comes at the cost of a memory-overhead in each layer due to the state that is kept between time-steps. The additional overhead of storing the partially computed feature-maps for a frame is:

$$\mathcal{O}(d_T \cdot [k_T - 1] \cdot c_O \cdot n_H \cdot n_W). \tag{5}$$

However, in the context of inference in a deep neural network, the transient memory usage within each time-step is reduced by a factor of n_T to

$$\mathcal{O}(c_O \cdot n_H \cdot n_W). \tag{6}$$

```
def coconv3d(frame, prev_state = (mem, i)):
    frame = spatial_padding(frame)
    frame = temporal_padding(frame)
    feat = conv3d(frame, weights)
    output, rest_feat = feat[0], feat[1:]

    mem, i = prev_state or init_state(output)
    M = len(mem)

    for m in range(M):
        output += mem[(i + m) % M, M - m - 1]
    output += bias

    mem[i] = rest_feat
    i = (i + 1) % M

    return output, (mem, i)
```

Listing 1: **Pseudo-code** for Continual Convolution.
The full implementation is available at
`https://github.com/lukashedegaard/co3d`.

The benefits of Continual Convolutions thus include the independence of clip length on the computational complexity, state overhead, and transient memory consumption. The change from (non-causal) regular convolutions to (causal) Continual Convolutions has the side-effect of introducing a delay to the output. This is because some intermediary results of convolving a frame with the kernel are only added up at a later point in time (see Fig. 3). The delay for a continual convolution is

$$\mathcal{O}(d_T \cdot [k_T - 1]). \tag{7}$$

3.3. Continual Residuals

The delay from Continual Convolutions has an adverse side-effect on residual connections. Despite their simplicity in regular CNNs, we cannot simply add the input to a Continual Convolution with its output because the *Co*Conv may delay the output. Residual connections to a *Co*Conv must therefore be delayed by an equivalent amount (see Eq. (7)). This produces a memory overhead of

$$\mathcal{O}(d_T \cdot [k_T - 1] \cdot c_O \cdot m_H \cdot m_W). \tag{8}$$

3.4. Continual Pooling

The associative property of pooling operations allows for pooling to be decomposed across dimensions, i.e. $\text{pool}_{T,H,W}(\mathbf{X}) = \text{pool}_T(\text{pool}_{H,W}(\mathbf{X}))$. For continual spatio-temporal pooling, the pooling over spatial dimensions is equivalent to a regular pooling, while the intermediary pooling results must be stored for prior temporal frames. For a pooling with temporal kernel size k_T and spatial output size $n_H \cdot n_W$, the memory consumption is

$$\mathcal{O}([k_T - 1] \cdot n_H \cdot n_W), \tag{9}$$

and the delay is

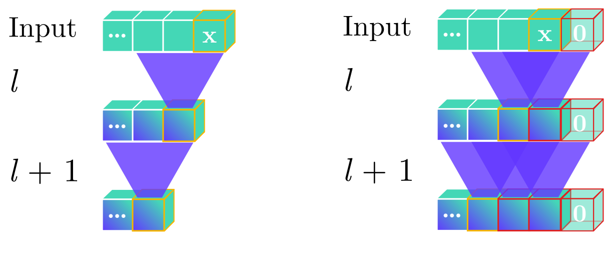
$$\mathcal{O}(k_T - 1). \tag{10}$$

Both memory consumption and delay scale linearly with the temporal kernel size. Fortunately, the memory consumed by temporal pooling layers is relatively modest for most CNN architectures (1.5% for *Co*X3D-M, see Appendix A). Hence, the delay rather than memory consumption may be of primary concern for real-life applications. For some network modules it may even make sense to skip the pooling in the conversion to a Continual CNN. One such example is the 3D Squeeze-and-Excitation (SE) block [14] in X3D, where global spatio-temporal average-pooling is used in the computation of channel-wise self-attention. Discarding the temporal pooling component (making it a 2D SE block) shifts the attention slightly (assuming the frame contents change slowly relative to the sampling rate) but avoids a considerable temporal delay.

3.5. The issue with temporal padding

Zero-padding of convolutional layers is a popular strategy for retaining the spatio-temporal dimension of a feature-map between consecutive layers in a CNN. For Continual CNNs, however, temporal zero-padding poses a problem, as illustrated in Fig. 4. Consider a 2-layer 1D CNN where each layer has a kernel size of 3 and zero padding of 1. For each new frame in a continual stream of inputs, the first layer l should produce two output feature-maps: One by the convolution of the two prior frames and the new frame, and another by convolving with one prior frame, the new frame, and a zero-pad. The next layer $l+1$ thus receives two inputs and produces three outputs which are dependent on the new input frame of the first layer (one for each input and another from zero-padding). In effect, each zero padding in a convolution forces the next layer to retrospectively update its output for a previous time-step in a non-causal manner. As we move through layers, the available computational redundancy to be exploited by Continual Convolutions is gradually reduced. Thus, there is a considerable downside to the use of padding.

This questions the necessity of zero padding along the temporal dimension. In regular CNNs, zero padding has two benefits: It helps to avoid spatio-temporal shrinkage of feature-maps when propagated through a deep CNN, and it prevents information at the boarders from "washing away" [19]. The use of zero-padding, however, has the downside that it alters the input-distribution along the boarders significantly [25, 28]. For input data which is a continual stream of frames, a shrinkage of the feature-size in the temporal dimension is not a concern, and an input frame (which may be considered a border frame in regular 3D CNN) has no risk of "washing away" because it is a middle frame in subsequent time steps. Temporal padding is thus omitted in Continual CNNs. As can be seen in the experimental evaluations presented in the following, this constitutes an approximation error in the conversion from reg-



(a) No padding (b) Zero padding

Figure 4: **Issue with temporal padding**: The latest frame $\mathbf{x}$ is propagated through a CNN with (purple) temporal kernels of size 3 (a) without or (b) with zero padding. Highlighted cubes can be produced only in the latest frame, with yellow boarder indicating independence of padded zero and red boarders dependence. In the zero-padded case (b), the number of frame features dependent on $\mathbf{x}$ following a layer l increases with the number of padded zeros.

ular to Continual 3D CNN if the former was trained with temporal padding.

3.6. Initialisation

Before a Continual CNN reaches a steady state of operation, it must have processed $r_T - 1$ frames where r_T is the receptive field size of the network in the temporal dimension. For example, the Continual X3D-{S,M,L} models have receptive fields of size $\{69, 72, 130\}$. There is thus a transient network response at start-up. This response depends on how internal state variables are initialised. Here, we propose two variants: 1) Initialisation with *zeros* and 2) by repeating a *replicate* of the features corresponding to the first input-frame. The latter corresponds to operating in a steady state for a "boring video" [3] which has one frame repeated in the entire clip.

3.7. Memory consumption: A case study

Where does this leave us with regards to the memory consumption during inference in online-scenarios? To gauge this, we can compare the worst-case memory consumption for the regular X3D-M model [7] with the corresponding Continual X3D-M (*Co*X3D-M) model. Disregarding the storage requirement of the model weights (which is identical for both regular and continual model versions), the memory consumption of storing input-frames between time-steps for the regular model is computed by Eq. (2): For a clip size of $m_T \times m_H \times m_W = 16\times224\times224$, the memory requirement is $3\cdot(16-1)\cdot224\cdot224 = 2{,}257{,}920$ floating point numbers (floats). The final consumption in bytes depends on the floating-point precision. In addition, we need to account for the transient memory consumption, which occurs while storing the intermediary feature-maps. The worst case is the output of the first convolution which is of size $24 \times 16 \times 112 \times 112$ corresponding to 4,816,896 floats. Thus, the regular X3D-M has a worst-case total memory-consumption of 7,074,816 floats.

The continual version of X3D-M (*Co*X3D-M) has no need to store input frames between time-steps, but it must keep state for the Continual Convolutions, Continual Pooling and Continual Residuals. 6,107,568 floats are stored as state within the *Co*X3D-M model (see Appendix A). The worst-case transient memory again occurs while storing the first feature-map, but this time it is only of size $24 \times 1 \times 112 \times 112$, or 301,056 floats. The Continual X3D-M thus has a worst case total memory-consumption of 6,408,624 floats. This corresponds to a 9.4% reduction in total memory-requirements relative to the regular model.

3.8. Design considerations

It should be noted that the result of the comparison in Section 3.7 is highly dependent on the clip size used by the regular model. For a clip size of 4 frames (as in X3D-XS), X3D-M$_4$ would have a worst-case memory consumption of 1,655,808 floats. Its continual counterpart, in which the temporal kernel size of the final pooling layer is reduced to 4, needs 6,403,440 floats. Thus, it may still be worth considering the use of the regular CNN in some embedded systems, which are severely constrained on memory, if lowering the spatial resolution is not an option. On the other hand, using a larger clip size of 64, X3D-M$_{64}$ would consume 28,449,792 floats, whereas *Co*X3D-M$_{64}$ has a worst-case consumption of only 6,429,360 floats. This simple example demonstrates that Continual CNNs utilise longer effective clip sizes much more efficiently than regular CNNs in online processing scenarios. In networks intended for embedded systems or online processing scenarios, we may thus increase the clip size to achieve higher accuracy with minimal penalty in computational complexity and worst-case memory consumption.

Another design-consideration, which has a considerable influence on memory consumption is the temporal kernel size and dilation of *Co*Conv layers. Fortunately, the trend to employ small kernel sizes leaves the memory consumption reasonable for current state-of-the-art networks [3, 36, 8, 7]. A larger temporal kernel size would not only affect the memory growth through the *Co*Conv filter, but also for co-occuring residual connections, since these consume a significant fraction of the total state-memory for real-life networks; in a Continual X3D-M model (*Co*X3D-M) the memory of residual constitutes 31.6% of the total model state memory (see Appendix A).

	Model	Acc. (%)	Params (M)	FLOPs (G)	Speed (evaluations/s)			
					CPU	TX2	Xavier	RTX 2080 Ti
Clip	I3D-R50	63.98	28.04	28.61	0.93 ± 0.04	2.54 ± 0.02	5.37 ± 0.01	77.15 ± 0.88
	R(2+1)D-18$_8$	53.52	31.51	20.35	1.75 ± 0.11	3.19 ± 0.04	6.82 ± 0.01	130.88 ± 0.29
	R(2+1)D-18$_{16}$	59.29	31.51	40.71	0.83 ± 0.06	1.82 ± 0.01	3.77 ± 0.01	75.81 ± 0.21
	SlowFast-8×8-R50	68.45	66.25	50.72	0.34 ± 0.01	0.87 ± 0.00	1.66 ± 0.03	30.72 ± 0.34
	SlowFast-4×16-R50	67.06	34.48	36.46	0.55 ± 0.02	1.33 ± 0.01	2.13 ± 0.05	41.28 ± 0.51
	X3D-L	69.29	6.15	19.17	0.25 ± 0.01	0.19 ± 0.00	0.88 ± 0.00	16.58 ± 0.13
	X3D-M	67.24	3.79	4.97	0.83 ± 0.04	1.47 ± 0.00	3.69 ± 0.02	55.27 ± 0.67
	X3D-S	64.71	3.79	2.06	2.23 ± 0.11	2.68 ± 0.01	8.07 ± 0.12	138.04 ± 1.69
	X3D-XS	59.37	3.79	0.64	8.26 ± 0.11	8.20 ± 0.09	26.37 ± 0.03	430.15 ± 9.29
Frame	*Co*X3D-L$_{16}$	63.03	6.15	1.54	2.30 ± 0.07	0.99 ± 0.00	6.30 ± 0.00	101.38 ± 3.36
	*Co*X3D-L$_{64}$	71.61	6.15	1.54	2.30 ± 0.08	0.99 ± 0.00	6.30 ± 0.01	111.53 ± 4.55
	*Co*X3D-M$_{16}$	62.80	3.79	0.40	7.57 ± 0.14	7.26 ± 0.13	23.70 ± 0.09	335.42 ± 9.91
	*Co*X3D-M$_{64}$	71.03	3.79	0.40	7.51 ± 0.17	7.04 ± 0.03	23.70 ± 0.09	323.56 ± 9.91
	*Co*X3D-S$_{13}$	60.18	3.79	0.21	13.16 ± 0.35	11.06 ± 0.03	40.09 ± 0.04	722.43 ± 56.95
	*Co*X3D-S$_{64}$	67.33	3.79	0.21	13.19 ± 0.37	11.13 ± 0.04	40.10 ± 0.04	726.81 ± 67.38

Table 1: **Benchmark of state-of-the-art methods on Kinetics-400**. The noted accuracy is the single clip or frame top-1 score using RGB as the only input-modality. The performance was evaluated using publicly available pre-trained models without any further fine-tuning. For speed comparison, evaluations per second denote frames per second for the *Co*X3D models and clips per second for the remaining models. Speed results are the mean ± std of 100 measurements.

4. Experiments

The experiments in this section aim to show the characteristics and advantages of Continual 3D CNNs as compared with regular 3D CNNs. One of the main benefits of *Co*3D CNNs is their ability to reuse the network weights of regular 3D CNNs. As such, all *Co*3D CNNs in these experiments use network weights that were pre-trained with Kinetics-400 [21] on regular 3D CNNs without further fine-tuning. Unless stated otherwise, we use the data same pre-processing steps as [8, 7].

The section is laid out as follows: First, we showcase the network performance following weight transfer from regular to Continual 3D. This is followed by a study on the transient response of *Co*3D CNNs at startup. Furthermore, we show how the computational advantages of *Co*3D CNNs can be exploited to improve accuracy by extending the temporal receptive field. Finally, we perform an extensive benchmark of prior state-of-the-art methods and Continual 3D CNNs, measuring the 1-clip/frame accuracy of publicly available models, as well as their inference speed on various computational devices.

4.1. Transfer from regular to Continual CNNs

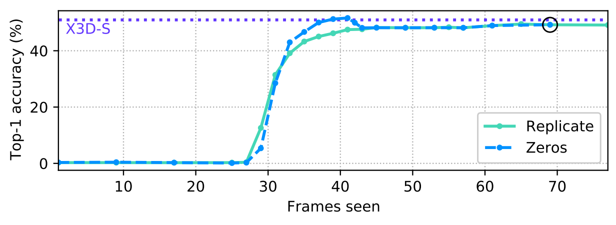
There is a one-to-one correspondence between the weights in a regular CNN and the continual version of the same network. Because Continual CNNs do not utilise zero-padding, however, there is a computational discrepancy between them. To gauge the direct transferability of knowledge to Continual CNNs, we initialise a set of Continual X3D (*Co*X3D) networks with Kinetics-400 pre-trained X3D network weights [7]. The publicly available X3D network variants XS, S, M, and L were evaluated on the Kinetics-400 test set using one temporally centred clip from each video. We omit the XS network in the transfer to *Co*X3D, given that it is architecturally equivalent to S, but with fewer frames per clip. In evaluation of the continual networks, we faced the challenge that the Kinetics videos were all limited to 10 seconds. Due to the relatively long transient response of Continual CNNs (see Section 4.2) and low frame-rate used for training the X3D models ($5.0, 6.0, 6.0$ FPS for S, M, and L respectively), the video-length was insufficient to reach steady-state. As a practical measure to evaluate near steady-state, we repeated the last video-frame for a padded video length of $\approx 80\%$ of the network receptive field as a heuristic choice. The Continual CNNs were thus tested on the last frame of the padded video and initialised with the prior frames.

The results of the transfer are shown in Table 1 and Fig. 1. For all networks, the transfer from regular to Continual 3D CNN results in significant computational savings. For the S, M, and L networks the reduction in FLOPs is 10.0×, 12.4×, and 12.4× respectively. These savings are less than the clip size due to the final pooling and prediction layers, which are in use for each frame in the Continual CNN, but only once per clip in regular CNNs. As a side-effect of the transfer from zero-padded regular CNN to Continual CNN without zero-padding, we see a noticeable reduction in accuracy. This reduction is easily improved by using an extended pooling size for the network (discussed in Section 3.8 and in Section 4.3). Using a global average pooling with temporal kernel size 64, we improve the accuracy of X3D by 2.6%, 3.8%, and 2.3% in the Continual S, M, and L network variants. As noted earlier, the Kinetics dataset did not have sufficient frames to fill the temporal receptive field of the Continual CNNs in these tests. The results in Table 1 are thus lower than what may be expected for online scenarios given enough frames. We explore this further in Sections 4.2 and 4.3.

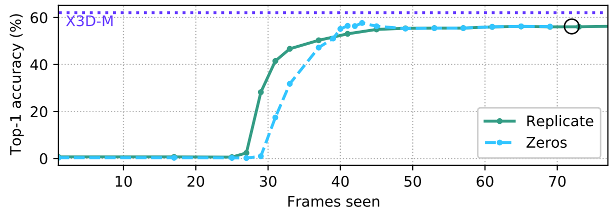
4.2. Transient response of Continual CNNs

As described in Section 3.6, Continual CNNs exhibit a transient response during their up-start. In order to gauge this response, we perform a set of experiments on the Kinetics-400 validation set, this time sampled at 15 FPS to have a sufficient number of frames available. This corresponds to a data domain shift [38] relative to the pre-trained weights, where time advances slower. First, we check the baseline X3D network 1-clip accuracy, which will constitute the expected upper bound. Then we evaluate the *Co*X3D frame accuracy with X3D weights, varying the number of prior frames used for initialisation. Note that temporal center-crops of size $T_{\text{init}} + 1$, where T_{init} is the number of initialisation frames, are used in each evaluation to ensure that the frames seen by the network come from the centre. This precaution is used to counter a data-bias, we noticed in Kinetics-400, namely that the start and end of a video are less informative and contribute to worse predictions than the central part. For an X3D-S network evaluated at different video positions, the results can vary up to 8%. The experiment is repeated for two initialisation schemes, “zeros” and “replicate”, and two model sizes, S and M. The measured transient responses are shown in Fig. 5.

For all responses, the first $\approx$25 frames produce near-random predictions, before rapidly increasing at $25-30$ frames until a steady-state is reached at 49.2% and 56.2% accuracy for S and M. Relative to the regular X3D, this constitutes a steady-state error of -1.7% and -5.8%. Comparing initialisation schemes, we see that the “replicate” scheme results in a slightly earlier rise. The rise sets in later for the “zeros” scheme, but exhibits a sharper slope, overshooting the steady-state accuracy (with peaks at 51.6% and 57.6%) before settling at the exact same values as for the “replication” scheme. This overshoot response was unexpected, and our best conjecture is that the network benefits from some zero states because the regular network, from which is inherited its parameters, was trained using zero padding. While this overshoot could be exploited to inflate the measured accuracy, we abstain from this practice and report only accuracy for steady-state or the latest-possible frame while using “replicate” padding in case of insufficient frames in the video.



(a) *Co*X3D-S



(b) *Co*X3D-M

Figure 5: **Transient response** for Continual X3D-{S,M} on the Kinetics-400 validation set at 15 FPS. The dotted horizontal lines constitute the X3D validation accuracy for 1-clip predictions. The black circles highlight when the frames seen equal the receptive field of the networks.

4.3. Extended receptive field

Continual CNNs experience a negligible increase in computational cost when larger temporal receptive field are used (see Section 3.8). For *Co*X3D networks, this extension can be trivially implemented by increasing the temporal kernel size of the last pooling layer. In this set of experiments, we extend *Co*X3D-{S,M,L} to have temporal pooling sizes 32, 64, and 96, and evaluate them on the Kinetics-400 validation set sampled at 15 FPS. Note, once again, that the network weights were trained on a different sampling rate. The Continual CNNs are evaluated at frames corresponding to the steady state.

Table 2 shows the measured accuracy and floating point operations per frame (*Co*X3D) / clip (X3D) as well as the pool size for the penultimate network layer (global average pooling) and the total receptive field of the network in the temporal dimension. A corresponding visual depiction is found in Fig. 6. As found in Section 4.1, each transfer results in significant computational savings alongside a drop in accuracy. Extending the kernel size of the global average pooling layer increases the accuracy of the Continual CNNs by $11.0-13.3\%$ for 96 frames relative the original 13-16 frames, surpassing that of the regular CNNs. Lying at $0.017-0.009\%$, the corresponding computational increases can be considered negligible.

4.4. Inference benchmarks

Despite their high status in activity recognition leaderboards [39], it is unclear how the state-of-the-art methods of late perform in the online setting, where speed and ac-

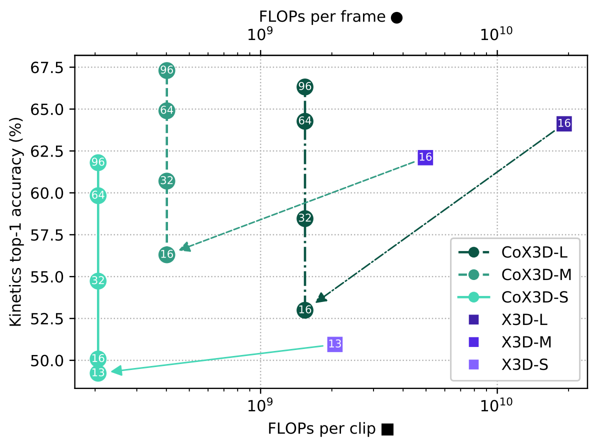


Figure 6: **Accuracy versus FLOPs for extended receptive fields** on the Kinetics-400 validation set at 15 FPS. The global average pool size along the temporal dimension is indicated in each point, and the arrows denote a one-to-one transfer from regular to Continual X3D without fine-tuning.

curacy constitute a necessary trade-off. Specifically, many methods sample multiple clips from different spatial and temporal positions in a video and average the predictions to produce their best results. To avoid the high computational cost of this practice, a reasonable tactic in the online setting is to make predictions from only a single clip. The number of FLOPs as a proxy for inference speed is occasionally noted, but to the best of our knowledge, there has not yet been a systematic evaluation of inference speed for these video-recognition models on real-life hardware. In this set of experiments, we benchmark the 1-clip accuracy of I3D-R50 [3], R(2+1)D-18 [36], SlowFast-8×8-R50 and SlowFast-4×16-R50 [33], as well as the X3D model family [7] alongside the 1-frame accuracy of *Co*X3D. To gauge the achievable speeds at different computational budgets, the networks were tested on four hardware platforms as described in Appendix B.

The results of the benchmark are found in Table 1. Due to the limitation to a single clip, we see that achieved accuracy for the baselines is generally lower than published in the respective works [3, 36, 8, 7]. The speed evaluation results are approximately (inversely log-log) proportional to the measured FLOPs. They are, however skewed to the low side for X3D models, confirming the observation in [26] that FLOPs are not always an accurate proxy for the inference speed on real-life hardware due to differences in memory access cost. Across hardware platforms, the *Co*X3D models with extended receptive fields attain the best accuracy/speed trade-off by a large margin. For example, *Co*X3D-S_{64} achieves an accuracy of 67.3% at 40.1 frames per second on the Nvidia Jetson Xavier, compared to 59.4% accuracy at 26.4 clips per second for X3D-XS.

Model	Size	Pool	Acc.	FLOPs (K)	Rec. Field
X3D	S	13	51.0	2,061,366	13
	M	16	62.1	4,970,008	16
	L	16	64.1	19,166,052	16
*Co*X3D	S	13	49.2	205,933	69
		16	50.1	205,934	72
		32	54.7	205,941	88
		64	59.8	205,955	120
		96	*61.8*	205,968	152
	M	16	56.3	401,829	72
		32	60.7	401,836	88
		64	64.9	401,850	120
		96	*67.3*	401,864	152
	L	16	53.0	1,540,474	130
		32	58.5	1,540,481	146
		64	*64.3*	1,540,495	178
		96	*66.3*	1,540,509	210

Table 2: **Effect of extending pool size**. Note that the model weights were trained at different sampling rates than evaluated at (15 FPS), resulting in a lower top-1 validation accuracy on Kinetics-400 than was originally reported in [7]. *Italic numbers* denote measurement taken within the transient response due to a lack of frames in the video-clip.

5. Conclusion

We have introduced Continual 3D Convolutional Neural Networks (*Co*3D CNNs), a new computational model for spatio-temporal 3D CNNs, which performs computations frame-wise rather than clip-wise while being weight-compatible with regular 3D CNNs. In doing so, we are able dispose of the computational redundancies faced by 3D CNNs in continual online processing, leading to a $10.0{-}12.4\times$ reduction of floating point operations, a $5.9{-}9.2\times$ real-life inference speed-up, and an accuracy improvement of $2.8{-}3.8\%$ on Kinetics-400 through an extension in the global average pooling kernel size.

While this constitutes a substantial leap in the processing efficiency of energy-constrained and real-time video recognition systems, there are still unanswered questions pertaining to the dynamics of *Co*3D CNNs. Specifically, the impact of extended receptive fields on the networks ability to change predictions in response to changing contents in the input video is untested. We leave this as an important direction for future work.

Acknowledgement

This work 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] Z. Cai, X. He, J. Sun, and N. Vasconcelos. Deep learning with low precision by half-wave gaussian quantization. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5406–5414, 2017.

[2] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about kinetics-600. *preprint, arXiv:1808.01340*, 2018.

[3] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733, 2017.

[4] Wenlin Chen, James T. Wilson, Stephen Tyree, Kilian Q. Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *International Conference on International Conference on Machine Learning (ICML)*, page 2285–2294, 2015.

[5] 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.

[6] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Image Analysis*, pages 363–370. Springer Berlin Heidelberg, 2003.

[7] Christoph Feichtenhofer. X3D: Expanding architectures for efficient video recognition. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[8] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[9] Nikolaos Floropoulos and Anastasios Tefas. Complete vector quantization of feedforward neural networks. *Neurocomputing*, 367:55–63, 2019.

[10] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations (ICLR)*, 2016.

[11] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1398–1406, 2017.

[12] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[13] 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. *preprint, arXiv:1704.04861*, abs/1704.04861, 2017.

[14] 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.

[15] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[16] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(1):221–231, 2013.

[17] 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.

[18] V. Kalogeiton, P. Weinzaepfel, V. Ferrari, and C. Schmid. Action tubelet detector for spatio-temporal action localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4415–4423, 2017.

[19] Andrej Karpathy. CS231n convolutional neural networks for visual recognition. URL: `https://cs231n.github.io/convolutional-networks/`. Last visited on 2021/01/26.

[20] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732, 2014.

[21] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *preprint, arXiv:1705.06950*, 2017.

[22] Okan Köpüklü, Stefan Hörmann, Fabian Herzog, Hakan Cevikalp, and Gerhard Rigoll. Dissected 3d cnns: Temporal skip connections for efficient online video processing. *arXiv preprint arXiv:2009.14639*, 2020.

[23] Okan Köpüklü, Xiangyu Wei, and Gerhard Rigoll. You only watch once: A unified cnn architecture for real-time spatiotemporal action localization. 2019.

[24] 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.

[25] Guilin Liu, Kevin J. Shih, Ting-Chun Wang, Fitsum A. Reda, Karan Sapra, Zhiding Yu, Andrew Tao, and Bryan Catanzaro. Partial convolution based padding. pages 1–11, 2018.

[26] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[27] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz. Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4207–4215, 2016.

[28] A. Nguyen, S. Choi, W. Kim, S. Ahn, J. Kim, and S. Lee. Distribution padding in convolutional neural networks. In *International Conference on Image Processing (ICIP)*, pages 4275–4279, 2019.

[29] Nikolaos Passalis and Anastasios Tefas. Learning deep representations with probabilistic knowledge transfer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[30] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (ICCV)*, 115(3):211–252, 2015.

[31] G. SingH and F. Cuzzolin. Recurrent convolutions for causal 3d cnns. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 1456–1465, 2019.

[32] G. Singh, S. Saha, M. Sapienza, P. Torr, and F. Cuzzolin. Online real-time multiple spatiotemporal action localisation and prediction. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3657–3666, 2017.

[33] Vladislav Sovrasov. ptflops. URL: `https://github.com/sovrasov/flops-counter.pytorch`. Last visited on 2021/03/02.

[34] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of Machine Learning Research*, volume 97, pages 6105–6114, 2019.

[35] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, 2015.

[36] 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.

[37] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *arXiv preprint arXiv:1609.03499*, 2016.

[38] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018.

[39] Papers with Code. Kinetics-400 leaderboard. URL: `https://paperswithcode.com/sota/action-classification-on-kinetics-400` Last visited on 2021/02/03.

[40] M. Xu, M. Zhu, Yunxin Liu, F. Lin, and X. Liu. Deepcache: Principled cache for mobile deep vision. *International Conference on Mobile Computing and Networking*, 2018.

[41] J. Yim, D. Joo, J. Bae, and J. Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7130–7138, 2017.

[42] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6848–6856, 2018.

[43] Linchao Zhu, Laura Sevilla-Lara, Yi Yang, Matt Feiszli, and Heng Wang. Faster recurrent networks for efficient video classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:13098–13105, 2020.

Appendix

A. Worst-case memory for *Co*X3D-M

In this section, we provide a detailed overview of the memory consumption incurred by the internal state in a Continual X3D-M (*Co*X3D-M) model. For Continual 3D CNNs, there is no need to store input frames between time steps, though this is the case for regular 3D CNNs applied in an online processing scenario. Intermediary computations from prior frames are kept in the continual layers as state if a layer has a temporal receptive field larger than 1. A continual $k_T \times k_H \times k_W = 1 \times 3 \times 3$ convolution is equivalent to a regular convolution, while a $3 \times 1 \times 1$ is not. The same principle holds for pooling layers. As a design decision, the temporal component of the average pooling of Squeeze-and-Excitation (SE) blocks is discarded. Hence, SE blocks do not incur a memory overhead or delay. Keeping the temporal pooling of the SE block would have increased memory consumption by a modest 85.050 $(+1.4\%)$. We can compute the total state overhead using Eq. (2), Eq. (8), and Eq. (9) by adding up the state size of each applicable layer shown in Table 4. An overview of the resulting computations can be found in Table 3. The total memory overhead for the network state is 6,192,618 floating point operations. In addition to the state memory, the worst-case transient memory must be taken into account. The largest intermediary feature-map is produced after the first convolution in conv_1 and has a size of $24 \times 112 \times 112 = 301{,}056$ floats. The total worst-case memory consumption for *Co*X3D-M (excluding models weights) is thus **6,408,624** floats.

If we were to reduce the model clip size from 16 to 4, this would result in a memory reduction of $5{,}184$ floats (only pool_5 is affected) for a total worst-case memory of $6{,}102{,}384$ floats (-0.08%). Increasing the clip size to 64 would yield an increased state memory of 20,736 floats giving a total worst-case memory of $6{,}128{,}304$ floats $(+0.3\%)$.

Stage	**Layer**		**Mem.** (floats)
conv_1	conv_T	$(5-1) \times 24 \times 112 \times 112 =$	1,204,224
res_2	residual_1	$(3-1) \times 24 \times 112 \times 112 =$	602,112
	residual_{2-3}	$[(3-1) \times 24 \times 56 \times 56] \times 2 =$	301,056
	conv_{1-3}	$[(3-1) \times 54 \times 56 \times 56] \times 3 =$	1,016,064
res_3	residual_1	$(3-1) \times 24 \times 56 \times 56 =$	150,528
	residual_{2-5}	$[(3-1) \times 48 \times 28 \times 28] \times 4 =$	301,056
	conv_{1-5}	$[(3-1) \times 108 \times 28 \times 28] \times 5 =$	846,720
res_4	residual_1	$(3-1) \times 48 \times 28 \times 28 =$	75,264
	residual_{2-11}	$[(3-1) \times 96 \times 14 \times 14] \times 10 =$	376,320
	conv_{1-11}	$[(3-1) \times 216 \times 14 \times 14] \times 11 =$	931,392
res_5	residual_1	$(3-1) \times 96 \times 14 \times 14 =$	37,632
	residual_{2-3}	$[(3-1) \times 192 \times 7 \times 7] \times 6 =$	112,896
	conv_{1-3}	$[(3-1) \times 432 \times 7 \times 7] \times 7 =$	296,352
pool_5	-	$(16-1) \times 432 =$	6,480
Total			**6,107,568**

Table 3: ***Co*X3D-M** state memory consumption by layer.

Stage	**Filters**	**Output size** $(T \times H \times W)$
input	-	$16 \times 224 \times 224$
conv_1	$1 \times 3^2, 24$ $5^* \times 1^2, 24$	$16 \times 112 \times 112$
res_2	res $\begin{bmatrix} 1 \times 1^2, 54 \\ 3 \times 3^2, 54 \\ \text{SE} \\ 1 \times 1^2, 24 \end{bmatrix} \times 3$	$16 \times 56 \times 56$
res_3	res $\begin{bmatrix} 1 \times 1^2, 108 \\ 3 \times 3^2, 108 \\ \text{SE} \\ 1 \times 1^2, 48 \end{bmatrix} \times 5$	$16 \times 28 \times 28$
res_4	res $\begin{bmatrix} 1 \times 1^2, 216 \\ 3 \times 3^2, 216 \\ \text{SE} \\ 1 \times 1^2, 96 \end{bmatrix} \times 11$	$16 \times 14 \times 14$
res_5	res $\begin{bmatrix} 1 \times 1^2, 432 \\ 3 \times 3^2, 432 \\ \text{SE} \\ 1 \times 1^2, 192 \end{bmatrix} \times 7$	$16 \times 7 \times 7$
conv_5	$1 \times 1^2, 432$	$16 \times 7 \times 7$
pool_5	16×7^2	$1 \times 1 \times 1$
fc_1	$1 \times 1^2, 2048$	$1 \times 1 \times 1$
fc_2	$1 \times 1^2, \#\text{classes}$	$1 \times 1 \times 1$

Table 4: **X3D-M model architecture**. When converted to a continual CNN, the highlighted components carry an internal state which results in a memory overhead. *Temporal kernel size in conv_1 is set to 5 as found in the official X3D source code `www.github.com/facebookresearch/SlowFast`.

B. Benchmarking details

This section should be read in conjunction with Section 4.4 of the main paper. To gauge the achievable on-hardware speeds of clip and frame predictions, a benchmark was performed on the following four system: A CPU core of a MacBook Pro (16-inch 2019 2.6 GHz Intel Core i7); Nvidia Jetson TX2; Nvidia Jetson Xavier; and a Nvidia RTX 2080 Ti GPU (on server with Intel XEON Gold processors). A batch size of 1 was used for testing on CPU, while the largest fitting multiple of $2^{\mathbb{N}}$ up to 64 was used for the other hardware platforms which have GPUs and lend themselves better to parallelisation. Thus, the speeds noted for GPU platforms in Table 1 of the main paper should not be interpreted as the number of processed clips/frames from a single (high-speed) video stream, but rather as the aggregated number of clips/frames from multiple streams using the available hardware. The exact batch size and input resolutions can be found in Table 5. In conducting the measurements, we assume the input data is readily available on the CPU and measure the time it takes for it to transfer from the CPU to GPU (if applicable), process, and transfer back to the CPU. A precision of 16 bits was used for the embedded platforms TX2 and Xavier, while a 32 bit precision was employed for CPU and RTX 2080 Ti. All networks were implemented and tested using PyTorch, and neither Nvidia TensorRT nor ONNX Runtime were used to speed up inference.

Model	**Input shape** ($T \times S^2$)	**Batch size** CPU	TX2	Xavier	RTX
I3D-R50	8×224^2	1	16	16	32
R(2+1)D-18_8	8×112^2	1	16	16	32
R(2+1)D-18_{16}	16×112^2	1	8	16	32
SlowFast-8×8-R50	8×256^2	1	8	32	32
SlowFast-4×16-R50	16×256^2	1	16	32	32
X3D-L	16×312^2	1	16	32	32
X3D-M	16×224^2	1	32	64	64
X3D-S	13×160^2	1	64	64	64
X3D-XS	4×160^2	1	64	64	64
*Co*X3D-L	1×312^2	1	8	16	32
*Co*X3D-M	1×224^2	1	32	64	64
*Co*X3D-S	1×160^2	1	32	64	64

Table 5: **Benchmark model configurations**. For each model, the input shape is noted as $T \times S^2$, where T and S are the temporal and spatial input shape.

8.5 Progressive Spatio-Temporal Bilinear Network with Monte Carlo Dropout for Landmark-based Facial Expression Recognition with Uncertainty Estimation

The appended paper follows.

Progressive Spatio-Temporal Bilinear Network with Monte Carlo Dropout for Landmark-based Facial Expression Recognition with Uncertainty Estimation

Negar Heidari and Alexandros Iosifidis
Department of Electrical and Computer Engineering, Aarhus University, Denmark
{negar.heidari,ai}@ece.au.dk

Abstract—**Deep neural networks have been widely used for feature learning in facial expression recognition systems. However, small datasets and large intra-class variability can lead to overfitting. In this paper, we propose a method which learns an optimized compact network topology for real-time facial expression recognition utilizing localized facial landmark features. Our method employs a spatio-temporal bilinear layer as backbone to capture the motion of facial landmarks during the execution of a facial expression effectively. Besides, it takes advantage of Monte Carlo Dropout to capture the model's uncertainty which is of great importance to analyze and treat uncertain cases. The performance of our method is evaluated on three widely used datasets and it is comparable to that of video-based state-of-the-art methods while it has much less complexity.**

I. Introduction

Facial expression recognition (FER) has been widely studied in the past several years and it is of great importance in different areas of computer vision such as social robotics and human-computer interaction (HCI). Although deep learning models have a high ability in feature learning, there are different challenges for employing them in facial expression recognition. The intra-class variability, including variations in age, gender, pose, illumination, face scale and appearance, necessitates the use of complex deep learning models to extract the most useful features for expression recognition [1]. However, existing publicly available datasets are not large and diverse enough to train high-performing deep learning models. Thus, designing compact neural network architectures for real-time facial expression recognition that can achieve high performance is of great importance.

It has been shown that the FER performance can be improved by using localized facial landmarks [2]. The motion of facial landmarks during the execution of a facial expression effectively represents the dynamic motion of the most informative facial parts, such as eyes, nose and mouth, for facial expressions and it is also invariant to illumination conditions and face appearance. However, while it has been shown that multi-modal data fusion based on facial landmarks and images or videos can improve performance of image or video-based FER [3]–[5], deep learning models employing only facial landmark features have been rarely studied.

FER methods can be categorized in *static methods*, which use an image as input to classify the facial expression depicted in it, and *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 work, we focus on dynamic facial expression recognition. Existing deep learning approaches for dynamic FER which utilize facial landmarks, typically concatenate their coordinates over multiple frames to form a sequence of vectors to be used by Recurrent Neural Networks (RNNs) [3], 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 (CNNs) [6]. 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 which are used for human action recognition (HAR) [7], [8], facial landmarks are also non-Euclidean structured data that can be modeled by a graph in which the landmark points are the graph nodes and the relationships between them are the edges connecting graph nodes. Therefore, the Spatio-Temporal Graph Convolutional Networks (ST-GCNs) [7], [8], the Progressive ST-GCN (PST-GCN) [9] which tries to find an optimized ST-GCN architecture, or the recently introduced Spatio-Temporal Bilinear Network (ST-BLN) [10] can be employed to extract informative features from a sequence of graphs, encoding facial landmarks through different time steps, for facial expression recognition.

One aspect of a real FER system that is often neglected by FER methods is that of classification uncertainty. In a real-world scenario, the FER system will analyze the facial expressions of a person and take actions which can take the form of, for example, recommendations to perform an activity. Spurious misrecognized expressions caused either by misclassification due to a low-performing model, or by false identification of an expression (e.g. sadness instead of neutral) due to high uncertainty, would lead to frustration to the user. Thus, for a FER system to be practical, it needs to be based on a high-performing model which can run in real-time and estimate the uncertainty of its predictions.

In this paper, we propose the Progressive Spatio-Temporal Bilinear Network (PST-BLN) method for facial expression recognition. PST-BLN inherits the advantage of ST-BLN [10] to learn graph structures at each layer of the network topology without the requirement of a pre-defined graph structure, allowing for more flexible model design. Moreover, the PST-BLN method automatically defines an optimized, compact and data-dependant network topology without the need of thorough experimentation using user-designed topologies. Moreover, we propose to capture the model's uncertainty by training our PST-BLN model using Monte Carlo Dropout [11] for helping the users of FER system to treat uncertain cases explicitly.

II. Related work

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 achieve good performance in addition to their high efficiency [12].

Recently, a GCN-based method has been proposed in [13] which uses only facial landmarks for facial expression recognition. In [13], the landmark extractor [12] was adopted to extract accurate 2D coordinates of 68 landmark points from each facial image in an image sequence. The extracted landmarks were modeled by a directed spatio-temporal graph which is constructed using landmark points as nodes and triangle meshes among all landmarks, built by Delaunay method, as edges. Inspired by methods recently proposed for skeleton-based human action recognition, like the DGNN [14], the FER method [13] 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.

III. Proposed method

This section describes the proposed PST-BLN method for dynamic landmark-based facial expression recognition. The description starts with the graph construction procedure, followed by the description of the Spatio-Temporal Bilinear Layer (ST-BLL) and the proposed PST-BLN method. The combination of PST-BLN with Monte Carlo Dropout for estimating the model's uncertainty is finally described.

A. Spatio-temporal graph construction

By extracting the facial landmarks of all the images in a sequence, a spatio-temporal graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can be constructed where $\mathcal{V}$ is the node set of 2D coordinates of the facial landmarks and $\mathcal{E}$ is the set of graph edges encoding spatial and temporal connections between the landmarks through different time steps. In this work, we adopted the Dlib's facial landmark extractor [15] to extract accurate 2D coordinates of 68 landmark points from each facial image. It has been shown in [13] that the landmarks of the outer region of the face do not contain informative features for different facial expressions. Therefore, we remove the first 17 facial landmarks of each image and keep only the 51 landmarks carrying features of the key facial parts for facial expression recognition. Facial landmarks in each graph are normalized by subtracting the central landmark (nose). The triangle meshes among all landmarks obtained by Delaunay method make the spatial graph edges. The central node (nose) is set as the master node which is connected to all other graph nodes. The temporal graph edges connect each landmark into its corresponding landmark in its previous and subsequent frames.

We utilize the edge features of the graph which encode the motion of the facial muscles instead of the landmark coordinates. Each graph edge is bounded by two graph nodes and it can be defined as a feature vector representing both the length and direction information. As an example, we define the feature vector of a graph edge with source node $v_i = (x_i, y_i)$ and target node $v_j = (x_j, y_j)$ as $e_{ij} = (x_i - x_j, y_i - y_j)$. Therefore, each image in the sequence is modeled by a graph with E spatial edges and the PST-BLN receives as input a tensor $\mathbf{X} \in \mathbb{R}^{F \times T \times E}$ encoding a sequence of T spatial graphs expressing the connections of the graph edges. F denotes the feature dimension of each edge feature e_{ij}.

B. Spatio-Temporal Bilinear Layer

The ST-BLL is composed of a bilinear transformation and a temporal convolution. The bilinear transformation receives as input the representations for the $E^{(l-1)}$ facial graph edges at layer $l-1$, denoted by $\mathbf{H}^{(l-1)}$, and transforms them by using a learnable weight matrix $\mathbf{W}^{(l)}$ as follows:

$$\mathbf{H}_s^{(l)} = ReLU\left(\mathbf{U}^{(l)}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)}\right), \tag{1}$$

where $\mathbf{U}^{(l)} \in \mathbb{R}^{E^{(l)} \times E^{(l-1)}}$ is a learnable matrix indicating the spatial weighted connections between the facial graph edges. This matrix is initialized randomly and it is optimized in an end-to-end manner jointly with the parameters of the entire network. Unlike GCN layers which use the graph Adjacency matrix in the spatial graph convolution, ST-BLL allows for freely deciding the dimensions of matrix $\mathbf{U}$. This means that ST-BLL allows for aggregating (or expanding) information of the graph edges leading to $E^{(l)} < E^{(l-1)}$ (or $E^{(l)} > E^{(l-1)}$, respectively). In this paper, we chose to keep the number of graph edges constant for all ST-BLLs, and for notation simplicity we use E hereafter.

The spatially transformed feature tensor $\mathbf{H}_s^{(l)} \in \mathbb{R}^{F^{(l)} \times T^{(l)} \times E}$ with $F^{(l)}$ feature dimensions is introduced to the temporal convolution, which captures the motion of the facial muscles taking place in each facial expression by propagating the edge features of each spatial graph through the time domain using a standard 2D convolution with a predefined kernel size $K \times 1$ aggregating edge features in K consecutive frames. The structure of the ST-BLL is shown in Fig. 1. Each layer of the network is euqipped by two residual connections to stabilize the model by adding the input to the output of the bilinear mapping and the temporal convolution. The temporal convolution block is followed by batch normalization and ReLU activation function.

C. Progressive spatio-temporal bilinear network (PST-BLN)

A ST-BLN model is composed of several ST-BLLs for feature extraction and one fully connected layer for classification. A network with l ST-BLLs, employs the global average pooling after the l^{th} layer to produce a feature vector of size $F^{(l)} \times 1$. The feature vector is introduced to a fully connected

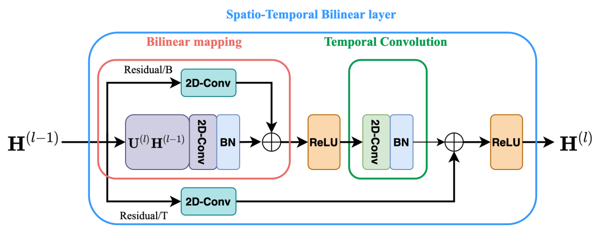


Fig. 1. Illustration of spatio-temporal bilinear layer l. It receives $\mathbf{H}^{(l-1)}$ of size $F^{(l-1)} \times T^{(l-1)} \times E$ as input and applies bilinear projection and temporal convolution to produce the output representation $\mathbf{H}^{(l)}$ of size $F^{(l)} \times T^{(l)} \times E$. The bilinear mapping block and the temporal convolution block are both followed by batch-normalization (BN) and ReLU activation function.

layer which maps features from $F^{(l)}$ to C dimensional subspace to classify features into C different classes.

Let us assume that a ST-BLN with $l-1$ layers has been already built, and the method proceeds in building the l^{th} layer. In practice, the bilinear projection and temporal convolution in 1 are standard 2D convolutions with filters of sizes $F^{(l)} \times 1 \times 1$, and $F^{(l)} \times F^{(l)} \times K \times 1$, respectively. $F^{(l)}$ denotes the number of output channels in the l^{th} layer and K denotes the kernel size in the temporal convolution. The residual connections are also standard 2D convolutions which transform the input data of the layer with filters of size $F^{(l)} \times 1 \times 1$ to have the same dimension as the layer's output. When the method starts building the l^{th} layer, the number of output channels in all the 2D convolutions is set to a predefined fixed number $F^{(l)} = b$ and at each iteration, it is increased by $F^{(l)} = F^{(l)} + b$. While all the model's parameters in the previously built layers are initialized by the finetuned weights, the newly added neurons to the network are initialized randomly and all the model parameters are fine-tuned in an end-to-end manner using back-propagation. The layer's width progression at iteration t is evaluated according to the model's performance in terms of categorical loss value on training data, i.e. $\alpha_w = (\mathcal{L}_{t-1}^{(l)} - \mathcal{L}_{t}^{(l)})/\mathcal{L}_{t-1}^{(l)}$. $\mathcal{L}_{t-1}^{(l)}$ and $\mathcal{L}_{t}^{(l)}$ denote the model's loss value at iterations $t-1$ and t, respectively. If $\alpha_w < \epsilon_w$ with $\epsilon_w > 0$, it shows that increasing the layer's width doesn't improve the model's performance anymore and the method stops progression in that layer. Otherwise, the newly added parameters are saved and the next iteration starts increasing the layer's width by adding b more output channels to the filters of all the 2D convolutions in that layer.

This process repeats iteratively until the performance converges in that layer. After building each layer of the network, the method evaluates the model's depth progression using the rate of improvement in model's performance, i.e. $\alpha_d = (\mathcal{L}^{(l-1)} - \mathcal{L}^{(l)})/\mathcal{L}^{(l-1)}$, in terms of categorical loss value on training data. $\mathcal{L}^{(l-1)}$ and $\mathcal{L}^{(l)}$ denote the model's loss value before and after adding the new layer to the network, respectively. When $\alpha_d < \epsilon_d$ with $\epsilon_d > 0$, the method stops depth progression and the newly added layer is removed. Finally, all the model's parameters are fine-tuned together and the method returns the optimized topology for the ST-BLN model and its performance on training and validation data.

D. *PST-BLN with Monte Carlo Dropout to model uncertainty*

People of different ages, genders and cultural backgrounds have different levels of expressiveness, and they perform or interpret the facial expressions in different ways. Although the output of a classification model (softmax scores) encodes the predictive (pseudo-)probabilities of the model, it has been shown that even models with high softmax outputs can be uncertain about their predictions [16]. Since facial expression datasets are small in size, regularization of the network parameters is needed to prevent overfitting. To address this, we add a dropout layer after each ST-BLL built by the proposed method using a dropout rate p of 0.2. This choice also allows us to use Monte Carlo Dropout [16] to capture the uncertainty of the model during inference. This is very helpful for the users of the FER system to interpret the facial expression of a sample when the model is uncertain about its prediction. The main idea of Monte Carlo Dropout is to use dropout not only in the training phase, but also during the inference. Since dropout randomly switches off a subset of neurons in each layer, it can be interpreted as a Bayesian approximation of the Gaussian process. Every time the model provides classification result with activated dropout layers, its outputs are obtained from slightly different models with different sets of activated neurons and each of these models can be treated as a Monte Carlo sample. By repeating the inference for an input facial spatio-temporal graph with an activated dropout, the outputs of the PST-BLN are combined as an ensemble of different PST-BLN models and the variance in the outputs are used to capture the classification uncertainty.

IV. Experimental results

A. *Datasets*

The preformance of our method has been evaluated on the following three widely used datasets:

CK+ [17], [18]: The Extended Cohn–Kanade (CK+) contains 327 videos of 7 different emotional classes, starting from a neutral expression to peak expression. Similar to most methods using this dataset for evaluation, we select the first frame and the last three frames (including the peak expression) of each sequence for landmark extraction. Besides, the subjects are divided into 10 groups for 10-fold cross-validation.

Oulu-CASIA [19]: The Oulu-CASIA dataset consists of 2,880 image sequences of 80 subjects, captured under three different illumination conditions and using two different imaging systems; near-infrared (NIR) and visible light (VIS). We used the 480 image sequences captured by the VIS system under normal indoor illumination and we divided the subjects into 10 groups for 10-fold cross validation. In each image sequence, we used the last three frames, including the peak expression, and the first frame showing the neutral expression.

AFEW [20], [21]: The Acted Facial Expressions in the Wild (AFEW) dataset is a more challenging dataset for landmark extraction methods compared to the CK+ and Oulu-CASIA. It consists a set of video clips collected from movies with actively moving faces in different illumination and environmental conditions. In some frames of each video where head pose is not frontal, the landmark extraction methods confront challenges to detect the face and extract its landmarks. Therefore, only a subset of video frames which provide meaningful facial landmark features are used. The dataset is divided into three sets, train, validation and test, with labels for the test set not being publicly available. Therefore, models are trained on the training set and evaluated on the validation set.

B. Experimental setup

The experiments are conducted with GeForce RTX 2080 GPUs, SGD optimizer with weight decay of 0.0005 and momentum of 0.9 and cross entropy loss function. The models are trained on AFEW dataset for 300 epochs on 4 GPUs with learning rate of 0.01 and batch size of 64. For CK+ and Oulu-CASIA datasets, the models are trained for 400 epochs with learning rate of 0.1 and batch size of 128. The PST-BLN method is trained with block sizes of 5 and layer/block thresholds of 0.0001 for all three datasets.

Since AFEW dataset is challenging and it does not have sufficient amount of data for training the model, we adopted data augmentation to expand the dataset size by 14 times. First, for each video we extracted the landmarks from 150 frames which are sampled at same time intervals and when the number of frames with meaningful landmarks are less than 150, we repeat the frames by tiling method. After landmark extraction, similar to [3], [13], we added three different Gaussian noises to facial landmarks, then we applied random rotation to the noised data followed by random flipping to each sequence.

C. Performance evaluation

The performance of the proposed method is compared with both video-based and landmark-based state-of-the-art methods on AFEW, Oulu-CASIA and CK+ datasets in Tables I, II, III, respectively. In each table, the state-of-the-art methods are divided into two groups. The first group contains the CNN-based or RNN-based methods which use videos or image sequences as the main data stream for training the model while some of these methods such as [22], [23] also utilize the landmark data in conjunction with video/image sequence to highlight the most important parts of the facial images and improve the performance. The second group contains the GCN-based methods which only use facial landmarks. To the best of our knowledge, the only GCN-based method that has been proposed for facial expression recognition is DGNN [13] which is an extension of [14] method for facial expression recognition. To show the effectiveness of our proposed model compared to other GCN-based networks, we also include in the comparisons the well-known GCN-based methods such as ST-GCN [7] and AGCN [8] to evaluate their performance on the landmark-based facial expression recognition task. Video-based methods train CNN and RNN-based architectures such as VGG16, LSTM, C3D, and they have the best performance on these datasets. However, the number of parameters of some of these methods is not reported in the corresponding papers.

TABLE I
COMPARISON OF VIDEO/IMAGE-BASED AND LANDMARK-BASED METHODS ON THE VALIDATION SET OF AFEW DATASET

Method	Acc(%)	#Params	Data type
SSE-HoloNet [24]	46.48	-	Video
VGG-LSTM [25]	48.60	-	Video
C3D-LSTM [25]	43.20	-	Video
C3D-GRU [26]	49.87	-	Video
ST-GCN [7]	28.17	131.3k	Landmark
AGCN [8]	24.21	143.7k	Landmark
DGNN [14]	32.64	538k	Landmark
ST-BLN w/MCD	**36.11**	**132.3k**	Landmark
ST-BLN wo/MCD	34.13	132.3k	Landmark
PST-BLN w/MCD	**33.33**	**10.8k**	Landmark
PST-BLN wo/MCD	30.15	10.8k	Landmark

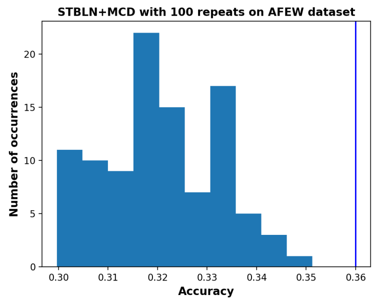


Fig. 2. The distribution of 100 classification accuraceis obtained by the ST-BLN w/MCD method on AFEW dataset. The vertical line in the left side indicates the classification accuracy obtained by the ensembled predictions.

The ST-BLN model is composed of 7 ST-BLLs with output dimensions of $\{8, 16, 16, 32, 32, 64, 64\}$, respectively and a fully connected layer for classification. This model topology is the same as DGNN's topology, and in order to have a fair comparison, we modified the topology of ST-GCN and AGCN models to have the same number of layers and layer dimensions as DGNN and ST-BLN models. While ST-GCN and AGCN methods utilize the landmark coordinates, or graph node features, and the squared Adjacency matrix of the graph in the spatial convolution, ST-BLN and PST-BLN utilize only the edge features of the graph. DGNN utilizes both node features and edge features encoded by a directed graph.

Experimental results on AFEW dataset indicate that ST-BLN outperforms all the baseline GCN-based methods, ST-GCN and AGCN, with a large margin while they have quite similar model complexity in terms of number of parameters. Compared to DGNN, ST-BLN has improved the classification performance by 4% while it has 4 times less number of parameters. PST-BLN which is trained with block sizes of 5 and layer/block thresholds of 0.0001, found an optimized topology for this dataset which is composed of 6 ST-BLLs with output sizes $\{15, 10, 15, 5, 5, 10\}$, respectively. This optimized model outperforms DGNN, ST-GCN and AGCN models with only $10.8k$ parameters which are 49 times less than those of DGNN.

To capture the model's uncertainty, we evaluated both ST-BLN and PST-BLN models with activated dropout layers during the inference and we repeated the inference for 100

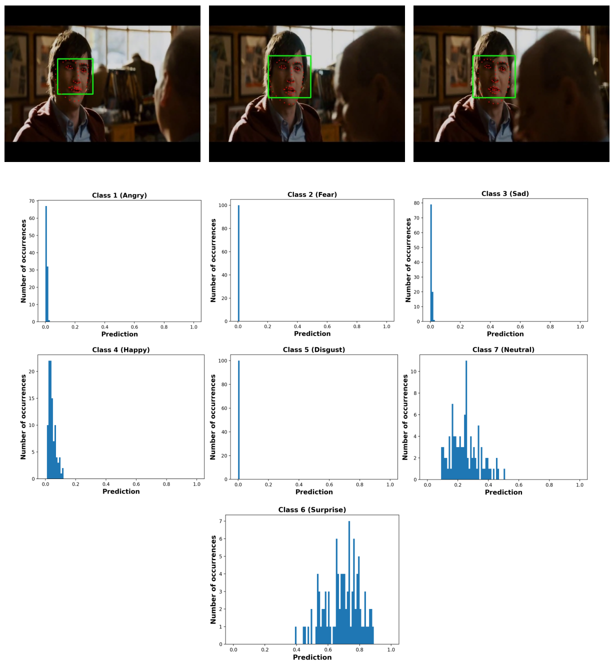


Fig. 3. Illustration of 3 frames of a sample video in AFEW dataset expressing 'Surprise', top row, and the distribution of 100 predictions for each class, obtained by our proposed ST-BLN model.

times on each sample to get 100 different prediction vectors. ST-BLN w/MCD and PST-BLN w/MCD denote the model's classification accuracy obtained as the mean of 100 different predictions and ST-BLN wo/MCD and PST-BLN wo/MCD report the classification accuracy obtained by performing the inference only once. The results show that the model achieves better performance when it ensembles the predictions of 100 models rather than performing the inference only once. To calculate the model's uncertainty on a dataset, we calculate the classification accuracy over 100 runs. Fig. 2, shows the distribution of 100 classification accuracy values of the ST-BLN w/MCD on AFEW dataset. The mean and standard deviation of this distribution are 32.09, 1.18, respectively. The classification accuracy obtained by the ensembled predictions is shown by a vertical line in the left side of the figure which is 36.11% and it is around 4% better than the mean accuracy.

Additionally, our proposed method gives the user the possibility of visualizing the model's uncertainty on an individual sample base. As an example, we evaluated the ST-BLN model on a video sample of class Surprise from the AFEW dataset. Fig. 3 illustrates 3 frames of this video with their extracted facial landmarks and also the prediction distribution for each expression class. This figure shows that the model classifies this sample correctly in the Surprise class with mean probability of 0.69 while it is uncertain about it. Considering the sample frames in the top row of the figure, it can be seen that it is a hard example to classify and based on the prediction distributions, this example can also be classified in Neutral and Happy classes with mean probabilities of 0.24, 0.04, respectively. The variance of the model predictions of

TABLE II
COMPARISON OF VIDEO-BASED AND LANDMARK-BASED METHODS ON OULU-CASIA DATASET USING 10-FOLD CROSS VALIDATION

Method	Acc(%)	#Params	Data type
DTAN [22]	74.38	-	Video
DTGN [22]	74.17	177.6k	Landmark
DTAGN [22]	81.46	-	Video + Landmark
PPDN [27]	84.59	6.8m	Video
PHRNN-MSCNN [23]	86.25	1.6m	Video + Landmark
DCPN [28]	86.23	-	Video
CDLM [29]	91.67	2.7m	Video
ST-GCN [7]	77.08	131.3k	Landmark
AGCN [8]	75.62	143.7k	Landmark
DGNN [14]	81.46	535,7k	Landmark
ST-BLN w/MCD	**83.54**	**132.3k**	Landmark
ST-BLN wo/MCD	82.08	132.3k	Landmark
PST-BLN w/MCD	**79.79**	**7.59k**	Landmark
PST-BLN wo/MCD	78.74	7.59k	Landmark

TABLE III
COMPARISON OF THE VIDEO-BASED AND LANDMARK-BASED METHODS ON CK+ DATASET USING 10-FOLD CROSS VALIDATION

Method	Acc(%)	#Params	Data type
DTAN [22]	91.44	-	Video
DTGN [22]	92.35	177.6k	Landmark
DTAGN [22]	97.25	-	Video + Landmark
PPDN [27]	99.3	6.8m	Video
PHRNN-MSCNN [23]	98.5	1.6m	Video + Landmark
DCPN [28]	99.6	-	Video
CDLM [29]	98.47	2.7m	Video
ST-GCN [7]	93.64	131.3k	Landmark
AGCN [8]	94.18	143.7k	Landmark
DGNN [14]	96.02	535,7k	Landmark
ST-BLN w/MCD	**95.47**	**132.3k**	Landmark
ST-BLN wo/MCD	93.19	132.3k	Landmark
PST-BLN w/MCD	**93.34**	**9.79k**	Landmark
PST-BLN wo/MCD	93.1	9.79k	Landmark

each class can be interpreted as the model's uncertainty on that class. Therefore, the model's uncertainty on classes Surprise, Neutral and Happy is 0.1, 0.9, 0.02, respectively.

The mean classification performance of the models over all folds is reported for Oulu-CASIA and CK+ datasets. Since the PST-BLN method finds a different model topology for each fold of the data, we report the average number of parameters of 10 optimized models. Experimental results on Oulu-CASIA dataset show that the proposed ST-BLN model outperforms all the landmark-based methods while it has 4 times less number of parameters compared to the DGNN method. PST-BLN is trained separately for each of the 10 folds of the data and the average number of parameters is reported which is around 70 times less than DGNN and 17 times less than ST-BLN. Although the PST-BLN does not outperform the state-of-the-art methods, it is competitive while being much more compact.

ST-BLN outperforms the ST-GCN and AGCN on CK+ dataset while it has competitive performance compared to DGNN with around 4 times less number of parameters. The optimized topology PST-BLN achieves similar performance to the ST-GCN and AGCN with around 13 and 14 times less number of parameters. It should be noted that the reported number of parameters in all the tables corresponds only to the neural network models. As we use [15] for landmark detection, other methods such as [22], [23], [29] also utilize IntraFace [30] landmark extractor and [28] employs MTCNN [31] for face detection and alignment as a pre-processing step.

The results of ST-BLN w/MCD and PST-BLN w/MCD on Oulu-CASIA and CK+ datasets also confirm that repeating the inference with activated dropouts and ensembling the results, improves the classification performance.

V. Conclusion

In this paper, we proposed a method which builds an optimized and compact spatio-temporal bilinear network topology for facial expression recognition by employing the localized facial landmarks instead of videos or image sequences. While our method has achieved comparable performance to more complex state-of-the-art methods, it captures the model's uncertainty using Monte Carlo Dropout technique which allows the user to analyze the model's prediction for different cases and take desired action.

Acknowledgment

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] Michel F Valstar, Marc Mehu, Bihan Jiang, Maja Pantic, and Klaus Scherer, "Meta-analysis of the first facial expression recognition challenge," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 4, pp. 966–979, 2012.

[2] Ali Mollahosseini, David Chan, and Mohammad H. Mahoor, "Going deeper in facial expression recognition using deep neural networks," in *IEEE Winter Conference on Applications of Computer Vision*, 2016.

[3] Heechul Jung, Sihaeng Lee, Junho Yim, Sunjeong Park, and Junmo Kim, "Joint fine-tuning in deep neural networks for facial expression recognition," in *IEEE International Conference on Computer Vision*, 2015.

[4] 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*, pp. 1–1, 2020.

[5] Behzad Hassani and Mohammad H. Mahoor, "Facial expression recognition using enhanced deep 3d convolutional neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017.

[6] Jingwei Yan, Wenming Zheng, Zhen Cui, Chuangao Tang, Tong Zhang, and Yuan Zong, "Multi-cue fusion for emotion recognition in the wild," *Neurocomputing*, vol. 309, pp. 27–35, 2018.

[7] Sijie Yan, Yuanjun Xiong, and Dahua Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *AAAI Conference on Artificial Intelligence*, 2018.

[8] 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.

[9] Negar Heidari and Alexandras Iosifidis, "Progressive spatio-temporal graph convolutional network for skeleton-based human action recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 3220–3224.

[10] Negar Heidari and Alexandros Iosifidis, "On the spatial attention in spatio-temporal graph convolutional networks for skeleton-based human action recognition," *arXiv preprint arXiv:2011.03833*, 2020.

[11] Joost van Amersfoort, Lewis Smith, Andrew Jesson, Oscar Key, and Yarin Gal, "Improving deterministic uncertainty estimation in deep learning for classification and regression," vol. abs/2102.11409, 2021.

[12] Xuanyi Dong, Yan Yan, Wanli Ouyang, and Yi Yang, "Style aggregated network for facial landmark detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[13] Quang Tran Ngoc, Seunghyun Lee, and Byung Cheol Song, "Facial landmark-based emotion recognition via directed graph neural network," *Electronics*, vol. 9, no. 5, pp. 764, 2020.

[14] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu, "Skeleton-based action recognition with directed graph neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[15] Vahid Kazemi and Josephine Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[16] Yarin Gal and Zoubin Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *International conference on machine learning*, 2016, pp. 1050–1059.

[17] Patrick Lucey, Jeffrey F Cohn, Takeo Kanade, Jason Saragih, Zara Ambadar, and Iain 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.

[18] Takeo Kanade, Jeffrey F Cohn, and Yingli Tian, "Comprehensive database for facial expression analysis," in *IEEE International Conference on Automatic Face and Gesture Recognition*. IEEE, 2000, pp. 46–53.

[19] Guoying Zhao, Xiaohua Huang, Matti Taini, Stan Z Li, and Matti PietikäInen, "Facial expression recognition from near-infrared videos," *Image and Vision Computing*, vol. 29, no. 9, pp. 607–619, 2011.

[20] Abhinav Dhall, Roland Goecke, Simon Lucey, and Tom Gedeon, "Collecting large, richly annotated facial-expression databases from movies," *IEEE Annals of the History of Computing*, vol. 19, no. 03, pp. 34–41, 2012.

[21] Abhinav Dhall, Roland Goecke, Jyoti Joshi, Karan Sikka, and Tom Gedeon, "Emotion recognition in the wild challenge 2014: Baseline, data and protocol," in *International conference on multimodal interaction*, 2014, pp. 461–466.

[22] Heechul Jung, Sihaeng Lee, Junho Yim, Sunjeong Park, and Junmo Kim, "Joint fine-tuning in deep neural networks for facial expression recognition," in *IEEE International Conference on Computer Vision*, 2015.

[23] Kaihao Zhang, Yongzhen Huang, Yong Du, and Liang Wang, "Facial expression recognition based on deep evolutional spatial-temporal networks," *IEEE Transactions on Image Processing*, vol. 26, no. 9, pp. 4193–4203, 2017.

[24] Ping Hu, Dongqi Cai, Shandong Wang, Anbang Yao, and Yurong Chen, "Learning supervised scoring ensemble for emotion recognition in the wild," in *ACM International Conference on Multimodal Interaction*, 2017.

[25] Valentin Vielzeuf, Stéphane Pateux, and Frédéric Jurie, "Temporal multimodal fusion for video emotion classification in the wild," in *ACM International Conference on Multimodal Interaction*, 2017.

[26] Min Kyu Lee, Dong Yoon Choi, Dae Ha Kim, and Byung Cheol Song, "Visual scene-aware hybrid neural network architecture for video-based facial expression recognition," in *IEEE International Conference on Automatic Face & Gesture Recognition*, 2019.

[27] Xiangyun Zhao, Xiaodan Liang, Luoqi Liu, Teng Li, Yugang Han, Nuno Vasconcelos, and Shuicheng Yan, "Peak-piloted deep network for facial expression recognition," in *European Conference on Computer Vision*, 2016.

[28] Zhenbo Yu, Qinshan Liu, and Guangcan Liu, "Deeper cascaded peak-piloted network for weak expression recognition," *The Visual Computer*, vol. 34, no. 12, pp. 1691–1699, 2018.

[29] Chieh-Ming Kuo, Shang-Hong Lai, and Michel Sarkis, "A compact deep learning model for robust facial expression recognition," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018.

[30] Xuehan Xiong and Fernando De la Torre, "Supervised descent method and its applications to face alignment," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[31] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.

8.6 Learning to ignore: rethinking attention in CNNs

The appended paper follows.

Learning to ignore: rethinking attention in CNNs

Firas Laakom*, Kateryna Chumachenko*, Jenni Raitoharju, Alexandros Iosifidis, and Moncef Gabbouj

Abstract—Recently, there has been an increasing interest in applying attention mechanisms in Convolutional Neural Networks (CNNs) to solve computer vision tasks. Most of these methods learn to explicitly identify and highlight relevant parts of the scene and pass the attended image to further layers of the network. In this paper, we argue that such an approach might not be optimal. Arguably, explicitly learning which parts of the image are relevant is typically harder than learning which parts of the image are less relevant and, thus, should be ignored. In fact, in vision domain, there are many easy-to-identify patterns of irrelevant features. For example, image regions close to the borders are less likely to contain useful information for a classification task. Based on this idea, we propose to reformulate the attention mechanism in CNNs to learn to ignore instead of learning to attend. Specifically, we propose to explicitly learn irrelevant information in the scene and suppress it in the produced representation, keeping only important attributes. This implicit attention scheme can be incorporated into any existing attention mechanism. In this work, we validate this idea using two recent attention methods Squeeze and Excitation (SE) block and Convolutional Block Attention Module (CBAM). Experimental results on different datasets and model architectures show that learning to ignore, i.e., implicit attention, yields superior performance compared to the standard approaches.

Index Terms—Computer vision, CNNs, attention mechanisms, CBAM, SE

1 Introduction

Inspired by the properties of the human visual system, attention mechanisms have been recently applied in the field of deep learning, resulting in improved performance of the existing models across multiple applications. In the context of computer vision, learning to attend, i.e., learning to highlight and emphasize relevant attributes of images, have led to development of novel approaches [1], [2] in Convolutional Neural Networks (CNNs), improving their capabilities in many tasks [3], [4], [5].

Related to the concept of attention, recent studies in neuroscience suggest that the ability of humans to successfully perform visual tasks is related to the ability to ignore and suppress distractive information [6], [7], [8]. For example, the authors of [7] show that differences in visual working memory capacity, i.e., ability to remember visual features of multiple objects, are specifically related to distractor-suppression activity in visual cortex. This idea is reinforced in [8], where the authors provide evidence on an inhibitory mechanism of suppression of salient distractors aimed at preventing them from capturing attention and being further processed by humans. Additional studies [9] report that ignoring the irrelevant information is a powerful learning tool for human cognition with ubiquitous effectiveness. Inspired by these findings, we investigate the intuition of learning to explicitly ignore irrelevant information in the field of computer vision and reformulate attention mechanisms commonly utilized in CNNs under the framework of learning to ignore rather than learning to attend.

Existing attention mechanisms used in CNNs learn the attention masks by directly optimizing for the high response of attributes of the image that are important for the prediction and, thus, should be focused on more. The learned attention masks are applied to feature representations, leading to higher emphasis put on the attributes of interest, and, therefore, resulting in implicit ignoration of the irrelevant features. In our work, we propose to rethink this logic and instead explicitly focus on ignoring irrelevant regions, hence achieving the attention to important regions implicitly. We argue that learning of features that should be ignored is an easier task than learning to attend and, therefore, optimization with such an objective leads to better training. Arguably, discriminative features of samples of different classes are harder to capture and often require more advanced feature learning. On the other hand, irrelevant attributes or attributes common between classes are often related to easy-to-identify patterns, such as borderline locations on the image or background features that can already be learned at early stages of training. Following this intuition, we design our method to explicitly optimize which attributes of the image should be ignored, and based on this, the important attributes that should be attended are derived implicitly. We validate this idea using two recent attention methods Squeeze and Excitation (SE) block and Convolutional Block Attention Module (CBAM) and show that indeed our intuition holds and explicitly learning features to ignore leads to better model performance.

Our contributions can be summarized as follows:

- We propose a new perspective on attention in computer vision where the main aim is to learn to ignore instead of learning to attend.
- We propose an implicit attention scheme which explicitly learns to identify the irrelevant parts of the scene and suppress them. The proposed approach can be incorporated into any existing attention mechanism.
- We validate this idea using two attention mech-

** Equal contribution*

F. Laakom, K. Chumachenko and M. Gabbouj are with Department of Computing Sciences, Tampere University, Tampere, Finland, Tampere University, Tampere, Finland.

J. Raitoharju is with the Programme for Environmental Information, Finnish Environment Institute, Jyväskylä, Finland.

A. Iosifidis is with the Department of Electrical and Computer Engineering, Aarhus University, Aarhus, Denmark.

anisms. Specifically, we reformulate Squeeze-and-Excitation (SE) block and Convolutional Block Attention Module (CBAM) using our paradigm, i.e., *learn to ignore*, and show the superiority of such an approach.

2 Related work

Attention mechanisms in vision. The idea of attention in vision tasks stems from the properties of selective focus in the human visual system, i.e., that humans do not perceive images as a whole, but rely on certain salient parts of them. This property gave rise to a variety of attention-based learning mechanisms aimed to enhance the performance in computer vision domain [3], [4], [10], finding its applications in a variety of tasks, including sequence learning [11], image captioning [5], and others [12], [13]. A subset of attention-driven methods is directed at CNNs and aims at selecting and highlighting relevant attributes in the feature space during training [1], [2]. Conventionally, this is achieved by learning attention masks over feature representations that encode the importance of different attributes in form of weights and applying these masks on intermediate feature representations. This results in higher influence of features relevant for decision making in subsequent layers.

Other tasks adjacent to this line of research include saliency estimation, image segmentation, and weakly-supervised object localization. In saliency estimation, the goal is to estimate salient, i.e., significant regions of the scene without any prior knowledge on the scene in unsupervised [14], [15] or supervised manner [16], [17], [18]. In image segmentation, the task is to partition a given image into a set of segments, based on either semantics (semantic segmentation) or individual objects (instance segmentation) [19]. In weakly-supervised object localization, the goal is to predict the location of the object given only image-level labels [20].

Within the attention mechanisms utilized in CNNs, two of the notable ones include Squeeze-and-Excitation block (SE) [1] and Convolutional Block Attention Module (CBAM) [2]. In SE, an attention mask is learned channel-wise based on global average-pooled features of intermediate representations and applied at multiple layers of the ResNet architecture [21]. A further extension is the CBAM method that enriches the SE mechanism by additional max-pooled input and learns spatial attention in addition to channel-wise one. The learned attention weight masks are then applied channel-wise or pixel-wise to corresponding feature maps. These methods were shown to lead to superior performance across various domains and can be incorporated in any CNN architecture.

Learning by ignoring. Learning by ignoring is a powerful learning paradigm, which has been used in various machine learning applications [22], [23], [24]. It has been leveraged in the context of saliency estimation [14], [23], [25], [26]. For example, the authors of [14] propose an unsupervised graph-based saliency estimation approach, where auxiliary variables are used to encode prior knowledge on regions to be ignored, such as dark regions, as it is assumed that they are less-likely to contain salient object. A similar approach was proposed for the color constancy problem [27]. In the context of machine translation, it has been shown that learning to ignore spurious correlations in the data can improve the performance of neural networks in zero-shot translation [22]. In the context of domain adaptation, a learning framework assigning and learning an 'ignoring' score for each training sample and re-weighting the total loss based on these scores was proposed in [24].

3 Learning to ignore in CNNs

Attention in CNNs is generally formulated in a form of a learned attention mask that emphasizes relevant information in a feature map. Formally, given a feature map $\mathbf{F}$, attention can be defined as follows:

$$\mathbf{F}' = \mathbf{F} \otimes f_\theta(\mathbf{F}), \tag{1}$$

where $\mathbf{F}'$ is the attended feature map output, $\otimes$ is the element-wise multiplication and $f_\theta(\cdot)$ is an attention function with learnable parameters θ, which takes as input a feature map $\mathbf{F}$ and returns an attention mask $f_\theta(\mathbf{F}) \in [0, 1]$. This mask is then element-wise multiplied with the original map $\mathbf{F}$ in order to produce the output map $\mathbf{F}'$. The mask $f_\theta(\mathbf{F})$ is expected to identify relevant spatial or channel information and output the 'importance score' for each attribute, producing high response for most relevant regions and smaller values for regions of lesser interest. This can be seen as an explicit attention mechanism, where the model $f_\theta(\cdot)$ learns to directly identify and highlight relevant information.

In this work, we develop a new formulation of the concept of attention in CNNs, where the main target is learning to ignore instead of learning to attend. By training the model to predict irrelevance of features, rather than their importance, we expect to simplify the training objective and, hence, to improve the learning of the model. Our approach consists of a function which learns to identify irrelevant or confusing parts of the feature map in order to suppress them, followed by inversion of predicted irrelevance scores. Formally, this can be formulated as follows:

$$\mathbf{F}' = \mathbf{F} \otimes T(g_\theta(\mathbf{F})), \tag{2}$$

where $g_\theta(\cdot)$ is a function with learned parameters θ that is expected to learn to highlight information in the feature map that is irrelevant or confusing for the prediction. This can be seen as an *ignoring mask* that outputs high values for attributes and regions that should be suppressed in the feature map. The function $T(\cdot)$ is a function with an output $T(x)$ inversely proportional to x, hence flipping the learned ignoring mask and transforming it into an attention mask. Similarly to Eq. (1), the final feature map $\mathbf{F}'$ is obtained by element-wise multiplication of the input map $\mathbf{F}$ and the flipped ignoring mask $T(g_\theta(\mathbf{F}))$.

Given an ignoring mask $g_\theta(\mathbf{F})$, the function $T(\cdot)$ can be any function satisfying the condition of being inversely proportional to its input and bounded between $[0, 1]$. In this work, we propose three variants:

$$T_1(x) = 1 - \alpha x, \tag{3}$$

$$T_2(x) = sigmoid(\frac{1}{x}), \tag{4}$$

$$T_3(x) = sigmoid(-x). \tag{5}$$

The first variant $T_1(\cdot)$ linearly converts the ignoring mask to an attention one, and α is a hyper-parameter controlling this linear scaling. The extreme case $\alpha = 0$ corresponds to the extreme case $\mathbf{F}' = \mathbf{F}$, i.e., none of the features are emphasized or suppressed. For the second and third variants T_2 and T_3, a sigmoid function is applied to ensure that the output is bounded between $[0, 1]$.

We argue that formulating the objective as learning of irrelevant features that should be ignored, as opposed to focusing on important features, is beneficial, as optimization of a model with such an objective is easier. This is due to potential presence of many easy-to-identify patterns of irrelevant attributes, such as borderline pixel locations, color and lighting perturbations, or background properties that are not correlated with the groundtruth labels. At the same time, information responsible for predictions is generally label-specific and harder to capture. Moreover, learning of discriminative attributes that can be regarded as important often requires learning of complex feature representations that can be achieved only at latter stages of training, while patterns irrelevant for decision making can often be identified already at the early stages.

It can be argued that standard attention, i.e., Eq. (1), is also learning to ignore as it is expected to indirectly assign smaller values for less important regions. However, function $f_\theta(\cdot)$ is optimized directly for highlighting relevant information and, hence, this can be seen as an implicit and indirect strategy of learning to ignore. In our approach, Eq. (2), the model $g_\theta(\cdot)$ is explicitly optimized for identifying the irrelevant or confusing parts and the function $T(\cdot)$ suppresses them. This can be seen as an implicit learning to attend approach and explicit learning to ignore approach, as opposed to the standard attention which has an explicit learning to attend formulation.

As can be seen, the main difference between implicit and explicit attention formulations is the presence of a flipping function $T(\cdot)$. It can be seen from Eq. (1) and Eq. (2) that $f_\theta(\cdot)$ can be directly replaced by $T(g_\theta(\cdot))$. This makes it straightforward to reformulate any existing explicit attention method to learn to ignore instead of learning to attend by applying an inversion function $T(\cdot)$ on top of the learned mask. This way, the model $g_\theta(\cdot)$ can be learned as the model $f_\theta(\cdot)$ in conventional attention methods, while its parameters will be optimized to detect irrelevant or confusing regions instead of relevant ones. In this paper, for the choice of the function $f_\theta(\cdot)$, we consider two state-of-the-art attention mechanisms, namely SE [1] and CBAM [2] , and we show how to reformulate them using our paradigm in the following subsections.

3.1 Ignoring with Squeeze-and-Excitation blocks

Squeeze-and-Excitation (SE) block [1] presents a mechanism to learn channel-wise attention, focusing on which features of the representation are important for prediction. This is achieved by squeezing the spatial information into a channel representation, followed by an excitation operation that highlights important channels via a bottleneck block. Formally, given a feature map $\mathbf{F}$, this is defined as follows:

$$f_\theta(\mathbf{F}) = \sigma(\mathbf{W}_2\delta(\mathbf{W}_1 GAP(\mathbf{F}))), \tag{6}$$

where $GAP(\cdot)$ denotes Global Average Pooling, δ is a ReLU activation, σ is the sigmoid function, $\mathbf{W}_1 \in \mathbb{R}^{c\times\frac{c}{r}}$ and $\mathbf{W}_2 \in \mathbb{R}^{\frac{c}{r}\times c}$ are linear layers, c is the number of channels in $\mathbf{F}$, and r is the reduction rate in the bottleneck block. Given the output $f_\theta(\mathbf{F})$, the attended feature map is obtained by applying the learned mask element-wise between corresponding channels.

To incorporate our ignoring paradigm into SE, we apply $T(\cdot)$ to the output $f_\theta(\mathbf{F})$, hence transforming its objective into learning features that should be ignored. Specifically, we define the three variants as: $f^1_\theta(\mathbf{F}) = 1 - \alpha\sigma(\mathbf{W}_2\delta(\mathbf{W}_1 GAP(\mathbf{F})))$; $f^2_\theta(\mathbf{F}) = \sigma(\frac{1}{\sigma(\mathbf{W}_2\delta(\mathbf{W}_1 GAP(\mathbf{F})))})$; $f^3_\theta(\mathbf{F}) = \sigma(-\mathbf{W}_2\delta(\mathbf{W}_1 GAP(\mathbf{F})))$ using the definitions of T_1, T_2, and T_3, respectively. As can be noticed, in the first two variants $T(\cdot)$ is applied directly on $f_\theta(\mathbf{F})$, while in the third case it is applied on pre-sigmoid output to ensure sufficiently wide range for attention scores.

3.2 Ignoring with Convolutional Block Attention Modules

Following the approach of SE, Convolutional Block Attention Module (CBAM) [2] extends it to incorporate spatial attention as well as to enrich channel attention with an additional input representation. Under the definition of attention in Eq. (1), this is formulated as follows:

$$\begin{gathered} f^{ch}(\mathbf{F}) = \sigma(\mathbf{W}_2\delta(\mathbf{W}_1(GAP(\mathbf{F}))) + \mathbf{W}_2\delta(\mathbf{W}_1(GMP(\mathbf{F})))), \\ \mathbf{F}^{ch} = \mathbf{F} \otimes f^{ch}(\mathbf{F}), \\ f^{sp}(\mathbf{F}^{ch}) = \sigma(Conv^{7\times7}(GAP(\mathbf{F}^{ch}) \frown GMP(\mathbf{F}^{ch}))), \end{gathered} \tag{7}$$

where f^{ch} and f^{sp} denote channel and spatial attention, respectively, $GAP(\cdot)$ and $GMP(\cdot)$ correspond to Global Average Pooling and Global Max Pooling, respectively, δ is a ReLU activation, σ is the sigmoid activation, $\mathbf{W}_1 \in \mathbb{R}^{c\times\frac{c}{r}}$ and $\mathbf{W}_2 \in \mathbb{R}^{\frac{c}{r}\times c}$ are linear layers, c is the number of channels in $\mathbf{F}$, and r is the reduction rate in the bottleneck block, similarly to SE. $\mathbf{F}^{ch}$ is the channel-wise attended feature map, $Conv^{7\times7}$ denotes a convolutional layer with 7×7 kernel, and $\frown$ denotes concatenation.

As can be seen, channel and spatial attention masks are applied sequentially and channel-attended feature representations are used as input to compute spatial attention. Following this, we transform CBAM for ignoring by addition of inversion function $T(\cdot)$ on top of both channel function $f^{ch}(\cdot)$ and spatial function $f^{sp}(\cdot)$ to reformulate their objectives as learning of features and regions to ignore. In both cases, variants of $T_1(\cdot)$ and $T_2(\cdot)$ are applied directly on the output of corresponding functions, and $T_3(\cdot)$ is applied on pre-sigmoid output.

4 Experimental Results

4.1 CIFAR10 & CIFAR100

We start by validating our approach on image classification task using CIFAR10 and CIFAR100 [28] datasets. To show invariance of the proposed approach to specific model architectures, we evaluate two state-of-the-art CNNs, namely, ResNet50 [21] and DenseNet [29] architectures. We report the results of standard models with no attention, models with applied CBAM and SE attention blocks, and models with our proposed ignoring approach with both CBAM and SE variants with the three inversion function variants presented in Section 3.

All the models are optimized using Stochastic Gradient Descent (SGD) [30] with a momentum of 0.9 [31], weight decay of 0.0001 [32], and a batch size of 128. The initial learning rate is set to 0.1 and is then decreased by a factor of 5 after 60, 120, and 160 epochs, respectively. The models are trained for 200 epochs with the best performance on the validation set used for testing. Each experiment is repeated three times and the average performance is reported. 40k images are used for training and 10k for validation. Standard data augmentation is used [33], [34].

In Table 1, we report the experimental results of the standard model, i.e., no attention, SE, and our different SE-based variants, namely, SE-Ign$_i$ where i indicates the flipping function used (T_1 or T_2 or T_3). For the first variant, i.e., SE-Ign$_1$, we experiment with three different values of hyper-parameter α: 1, 0.8, and 0.5. We note that for both architectures applying an explicit or implicit attention mechanism consistently outperforms the standard model. On CIFAR10, the best performance is achieved using our third variant, i.e., SE-Ign$_3$, which improves the results by 1% compared to standard and +0.3% compared SE using ResNet50 architecture. On CIFAR100, the lowest top1-% error rates are achieved by SE-Ign$_3$ and SE-Ign$_{1(\alpha=0.5)}$ for ResNet50 and DenseNet architectures, respectively. In fact, on this dataset our third variant boosts the accuracy by 4% compared to the standard and 1.85% compared to SE. This can be explained by the fact that for this dataset only 500 training samples per class are available, thus making it hard to directly learn the relevant visual features for each class. At the same time, the irrelevant features are more universal and typically independent of the class, thus making them easier to learn in a scarce data context.

In Table 2, we report the empirical results for the different CBAM-based variants. As can be seen, the results with this attention variant are consistent with our findings using SE. For both datasets and for both architectures, learning to ignore yields better performance compared to both the standard model and the SE attention. The top performance is achieved by either by CBAM-Ign$_{1(\alpha=0.5)}$ or CBAM-Ign$_{1(\alpha=0.8)}$ variant. More results can be found Supplementary material Table 1.

4.2 ImageNet

To further validate the effectiveness of our learning to ignore framework, we perform additional experiments on ImageNet dataset [35] using ResNet50. For training on ImageNet, optimization is done with SGD with the same weight decay and momentum as used for CIFAR datasets. The initial learning rate is set to 0.1 and reduced by a factor of 10 after 30, 60, and 80 epochs, respectively. The models are trained for 90 epochs with batch size of 256 with the results reported on the validation set.

Table 3 shows the results on ImageNet dataset, where Top-1 and Top-5 errors are reported. As can be seen, our results are consistent with findings on CIFAR10 and CIFAR100 datasets. Specifically, we find that applying attention, whether explicit or implicit, outperforms standard model. At the same time, the proposed framework based on ignoring outperforms the conventional attention in a vast majority of cases. In SE variant, SE-Ign$_{1(\alpha=1)}$ and SE-Ign$_3$ outperform the conventional approach, while other variants report competitive results with minimal gap. Best result of SE-Ign$_3$ outperforms the standard model by 1.1%. In CBAM, all variants of CBAM-Ign$_1$ outperform conventional approach on both Top-1 and Top-5 metric, and CBAM-Ign$_2$ and CBAM-Ign$_3$ outperform conventional CBAM on Top-5 metric, while being competitive on Top-1 metric. More results can be found Supplementary material Table 2.

4.3 NTU-RGBD

To further demonstrate the effectiveness of our approach, we additionally evaluate the proposed method in the multimodal fusion setting. Here, we rely on the Multimodal Transfer Module (MMTM) [36] architecture for our evaluation. MMTM is a method for fusing information from multiple modalities in multiple-stream architectures, which has recently shown good performance in a variety of tasks, including activity recognition, gesture recognition, and audiovisual speech enhancement.

The method relies on an architecture inspired from Squeeze-and-Excitation blocks placed between network branches. Specifically, considering a two-stream scenario, intermediate feature representations from two network branches corresponding to two modalities are first spatially squeezed into channel descriptors by applying global average pooling in each branch. The squeezed representations are subsequently concatenated and projected into a joint lower-dimensional space. The resulting features are transformed with two projection matrices corresponding to each of the two modalities to the spaces of original dimensionalities, and sigmoid activation is then applied to obtain attention masks. The masks are further multiplied element-wise with original feature representations in each branch.

As can be seen, the fusion module is essentially a multimodal SE-block with joint squeeze and modality-specific excitation operations, to which we apply our ignoring framework as described in Section 3.1. We perform experiments on NTU-RGBD dataset [37] for human action recognition, where we fuse the skeleton and RGB modalities, similarly to MMTM [36]. We follow our ignoring paradigm and replace the SE attention mask in each branch with our proposed approach. The rest of the architecture and training protocol follows that of MMTM. We initialize the model from ImageNet+Kinectics pretrained weights, finetune for 10 epochs with batch size 8, and report the test set performance of the model that performed best on validation set. The results are reported in Table 4. As can be seen, the proposed ignoring approaches outperform the baseline in the vast majority of cases.

4.4 Discussion

As can be seen from the experimental results in previous sections, learning to ignore consistently yields superior performance compared to the baselines. We argue that this

		CIFAR 10	CIFAR 100	
		Top-1 Error%	Top-1 Error%	Top-5 Error%
ResNet50	Standard	08.27 ± 0.54	34.06 ± 1.02	10.97 ± 0.54
	SE	07.63 ± 0.37	32.80 ± 0.11	09.97 ± 0.50
	SE-Ign$_{1(\alpha=1)}$	07.42 ± 0.29	32.50 ± 0.26	09.92 ± 0.37
	SE-Ign$_{1(\alpha=0.5)}$	07.61 ± 0.46	31.40 ± 0.68	**09.39 ± 0.19**
	SE-Ign$_{1(\alpha=0.8)}$	07.76 ± 0.73	32.71 ± 1.15	10.07 ± 0.64
	SE-Ign$_2$	07.66 ± 0.13	32.78 ± 0.77	10.11 ± 0.56
	SE-Ign$_3$	**07.28 ± 0.17**	**30.95 ± 0.08**	09.49 ± 0.36
DenseNet	Standard	07.07 ± 0.33	29.25 ± 0.10	08.26 ± 0.12
	SE	06.96 ± 0.05	29.43 ± 0.44	08.36 ± 0.33
	SE-Ign$_{1(\alpha=1)}$	06.94 ± 0.07	29.17 ± 0.07	08.22 ± 0.13
	SE-Ign$_{1(\alpha=0.5)}$	06.69 ± 0.04	**27.64 ± 0.30**	**07.30 ± 0.10**
	SE-Ign$_{1(\alpha=0.8)}$	06.95 ± 0.14	27.73 ± 0.41	07.39 ± 0.07
	SE-Ign$_2$	06.80 ± 0.09	28.08 ± 0.35	07.39 ± 0.23
	SE-Ign$_3$	**06.41 ± 0.08**	27.77 ± 0.54	07.65 ± 0.20

TABLE 1
Results of SE variants on CIFAR10 and CIFAR100 datasets.

		CIFAR 10	CIFAR 100	
		Top-1 Error%	Top-1 Error%	Top-5 Error%
ResNet50	Standard	08.27 ± 0.54	34.06 ± 1.02	10.97 ± 0.54
	CBAM	08.04 ± 0.03	31.46 ± 0.20	09.32 ± 0.15
	CBAM-Ign$_{1(\alpha=1)}$	07.78 ± 0.28	31.03 ± 0.25	09.28 ± 0.27
	CBAM-Ign$_{1(\alpha=0.5)}$	**07.17 ± 0.05**	30.58 ± 0.20	09.25 ± 0.23
	CBAM-Ign$_{1(\alpha=0.8)}$	07.40 ± 0.23	**30.28 ± 0.39**	**09.08 ± 0.33**
	CBAM-Ign$_2$	07.53 ± 0.29	31.42 ± 0.58	09.27 ± 0.21
	CBAM-Ign$_3$	07.60 ± 0.10	30.88 ± 0.22	09.38 ± 0.32
DenseNet	Standard	07.07 ± 0.33	29.25 ± 0.10	08.26 ± 0.12
	CBAM	07.21 ± 0.23	30.63 ± 0.23	08.90 ± 0.14
	CBAM-Ign$_{1(\alpha=1)}$	07.19 ± 0.26	29.63 ± 0.46	08.37 ± 0.39
	CBAM-Ign$_{1(\alpha=0.5)}$	06.53 ± 0.14	27.92 ± 0.19	07.58 ± 0.27
	CBAM-Ign$_{1(\alpha=0.8)}$	**06.40 ± 0.14**	**27.11 ± 0.08**	**07.33 ± 0.19**
	CBAM-Ign$_2$	06.80 ± 0.02	27.88 ± 0.59	07.62 ± 0.05
	CBAM-Ign$_3$	06.68 ± 0.05	27.94 ± 0.10	07.78 ± 0.21

TABLE 2
Results of CBAM variants on CIFAR10 and CIFAR100 datasets.

	Top-1 Error%	Top-5 Error%
Standard	23.73	06.85
SE	22.70	06.35
SE-Ign$_{1(\alpha=1)}$	22.60	**06.29**
SE-Ign$_{1(\alpha=0.5)}$	23.03	06.58
SE-Ign$_{1(\alpha=0.8)}$	22.88	06.30
SE-Ign$_2$	23.16	06.55
SE-Ign$_3$	**22.59**	06.32
CBAM	22.91	06.58
CBAM-Ign$_{1(\alpha=1)}$	**22.84**	06.50
CBAM-Ign$_{1(\alpha=0.5)}$	**22.84**	06.52
CBAM-Ign$_{1(\alpha=0.8)}$	**22.84**	06.40
CBAM-Ign$_2$	23.02	**06.39**
CBAM-Ign$_3$	23.10	06.44

TABLE 3
Results of CBAM and SE with variants of ignoring on ImageNet dataset

stems from the fact that learning irrelevant information is easier than identifying what should be attended. For example, in order to learn features that should be attended to, the model needs to first learn to extract patterns such as lines and edges and make associations with the class labels in order to produce a meaningful attention mask. On the other hand, irrelevant patterns, such as background textures and borderline pixels, are often shared across the dataset, are persistent and independent of the class labels, which makes them easier to learn. Therefore, it should be possible to learn them already in the early stages of training. Figure 1 shows the validation loss curves of the baseline attention methods and the best-performing ignoring methods with ResNet50 on CIFAR100 dataset (more training curves can be found in supplementary material). As can be seen, especially at the earlier stages of training, our approach results in lower loss with less fluctuations and more stable training, hence supporting our claim. From an optimization point of view, in the case of α=1, only the gradient of the attention blocks are flipped, and thus in the back-propagation, when they are summed with the gradient of the main block (which are not flipped), the total feedback carried to the earlier layers is different and does not correspond to a flipped version of the total sum of the standard attention. Thus, this yields different feedback and leads to a different optimal solution in the end of the training (Figure 7 in supplementary material).

Moreover, in Figure 2, we provide visual results of the class activation maps [38] produced by the different models on three different samples from validation set of ImageNet. As can be seen, the learning to ignore formulation leads to different attention maps compared to the explicit attention, i.e., learning to attend. Noticeably, standard CBAM attention tries to capture the relevant parts of the image directly, leading to the prediction being made based on the small part of the input that is considered by the model as the most important. This leads to the possibility that the model can miss some important parts of the class of interest on the image. As an example, only one of the plants on the lower

	MMTM	$\text{Ign}_{1(\alpha=1)}$	$\text{Ign}_{1(\alpha=0.5)}$	$\text{Ign}_{1(\alpha=0.8)}$	Ign_2	Ign_3
NTU-RGBD	89.98	89.99	**90.52**	88.70	90.21	90.36

TABLE 4
Accuracy on NTU-RGBD dataset

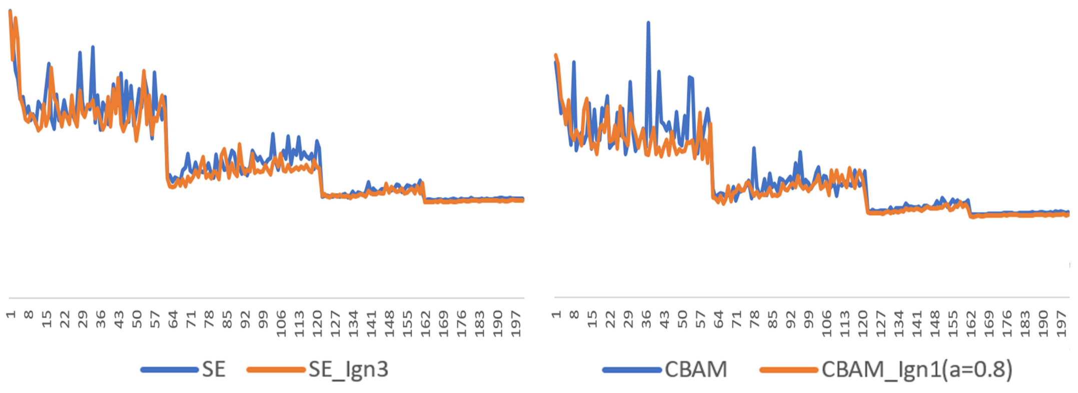


Fig. 1. Validation loss curves of ResNet50 on CIFAR100 using the different attention approaches.

figure is considered in CBAM model, as well as only a side of the bus in the middle image. On the other hand, our approach by learning to identify the non-relevant background regions first and subsequently suppressing them, simplifies the problem and typically results in an attention mask that is broader and captures the object of interest better, hence reducing the risk of suppressing relevant attributes of it.

5 Conclusion

In this paper, we provide a new perspective on attention in CNNs where the main target is learning to ignore instead of learning to attend. To this end, we propose an implicit attention scheme with three variants which can be incorporated into any existing attention mechanism. The proposed approach explicitly learns to identify the irrelevant and confusing parts of the scene and suppresses them. In addition, we reformulate two state-of-the-art attention approaches, namely SE and CBAM, using our learning paradigm. Experimental results on three image classification datasets show that learning to ignore, i.e., implicit attention consistently outperforms standard attention across multiple models.

Acknowledgments

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR), and the NSF-Business Finland Center for Visual and Decision Informatics (CVDI) project AMALIA. The authors wish to acknowledge CSC – IT Center for Science, Finland, for computational resources.

References

[1] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[2] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[3] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 20, no. 11, pp. 1254–1259, 1998.

[4] M. Jiang, Y. Yuan, and Q. Wang, "Self-attention learning for person re-identification." in *British Machine Vision Conference*, 2018, p. 204.

[5] 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*. PMLR, 2015, pp. 2048–2057.

[6] J. D. Cosman, K. A. Lowe, W. Zinke, G. F. Woodman, and J. D. Schall, "Prefrontal control of visual distraction," *Current biology*, vol. 28, no. 3, pp. 414–420, 2018.

[7] J. M. Gaspar, G. J. Christie, D. J. Prime, P. Jolicœur, and J. J. McDonald, "Inability to suppress salient distractors predicts low visual working memory capacity," *Proceedings of the National Academy of Sciences*, vol. 113, no. 13, pp. 3693–3698, 2016.

[8] N. Gaspelin and S. J. Luck, "The role of inhibition in avoiding distraction by salient stimuli," *Trends in cognitive sciences*, vol. 22, no. 1, pp. 79–92, 2018.

[9] C. A. Cunningham and H. E. Egeth, "Taming the white bear: Initial costs and eventual benefits of distractor inhibition," *Psychological science*, vol. 27, no. 4, pp. 476–485, 2016.

[10] H. Larochelle and G. E. Hinton, "Learning to combine foveal glimpses with a third-order boltzmann machine," *Advances in neural information processing systems*, vol. 23, pp. 1243–1251, 2010.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[12] G. Wu, X. Zhu, and S. Gong, "Spatio-temporal associative representation for video person re-identification." in *British Machine Vision Conference*, 2019, p. 278.

[13] F. Zhang, B. Ma, H. Chang, S. Shan, and X. Chen, "Relation-aware multiple attention siamese networks for robust visual tracking." in *British Machine Vision Conference*, 2019.

[14] C. Aytekin, A. Iosifidis, and M. Gabbouj, "Probabilistic saliency estimation," *Pattern Recognition*, vol. 74, pp. 359–372, 2018.

[15] J. Zhang, T. Zhang, Y. Dai, M. Harandi, and R. Hartley, "Deep unsupervised saliency detection: A multiple noisy labeling perspective," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9029–9038.

[16] N. Liu, J. Han, and M.-H. Yang, "Picanet: Learning pixel-wise contextual attention for saliency detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3089–3098.

[17] N. Liu, N. Zhang, K. Wan, J. Han, and L. Shao, "Visual saliency transformer," *arXiv preprint arXiv:2104.12099*, 2021.

[18] N. Liu and J. Han, "Dhsnet: Deep hierarchical saliency network for salient object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 678–686.

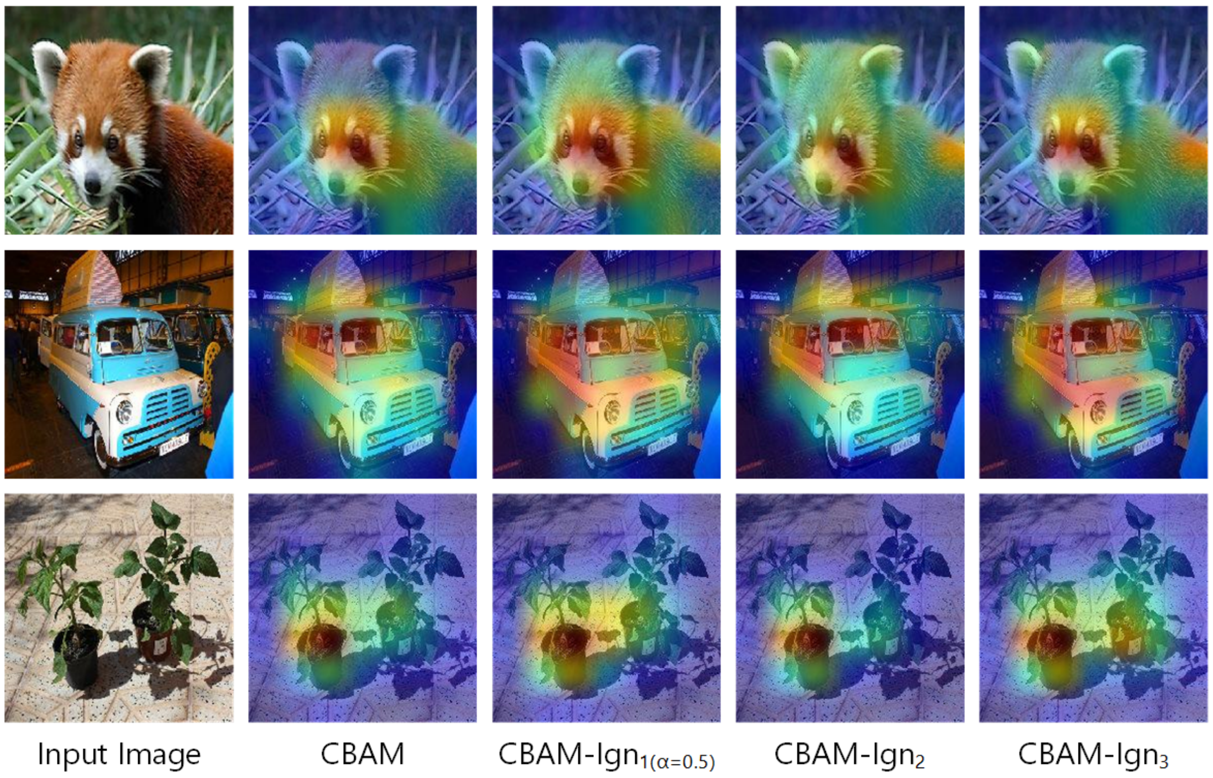


Fig. 2. Visual results of different CBAM-based attention mechanisms on three different samples from validation set of ImageNet. The attention masks are obtained as in [38].

[19] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[20] D. Zhang, J. Han, G. Cheng, and M.-H. Yang, "Weakly supervised object localization and detection: A survey," *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[22] J. Gu, Y. Wang, K. Cho, and V. O. Li, "Improved zero-shot neural machine translation via ignoring spurious correlations," *arXiv preprint arXiv:1906.01181*, 2019.

[23] B. Jiang, L. Zhang, H. Lu, C. Yang, and M.-H. Yang, "Saliency detection via absorbing markov chain," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 1665–1672.

[24] X. Zhao, X. He, and P. Xie, "Learning by ignoring, with application to domain adaptation," *arXiv preprint arXiv:2012.14288*, 2020.

[25] X. Li, H. Lu, L. Zhang, X. Ruan, and M.-H. Yang, "Saliency detection via dense and sparse reconstruction," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 2976–2983.

[26] W. Zhu, S. Liang, Y. Wei, and J. Sun, "Saliency optimization from robust background detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 2814–2821.

[27] F. Laakom, J. Raitoharju, A. Iosifidis, U. Tuna, J. Nikkanen, and M. Gabbouj, "Probabilistic color constancy," in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 978–982.

[28] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[29] 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*, 2017, pp. 4700–4708.

[30] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[31] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[32] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Advances in neural information processing systems*, 1992, pp. 950–957.

[33] 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*, 2017, pp. 4700–4708.

[34] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *International Conference on Learning Representations 2018*, 2018.

[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[36] H. R. V. Joze, A. Shaban, M. L. Iuzzolino, and K. Koishida, "Mmtm: Multimodal transfer module for cnn fusion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 289–13 299.

[37] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "Ntu rgb+ d: A large scale dataset for 3d human activity analysis," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1010–1019.

[38] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.