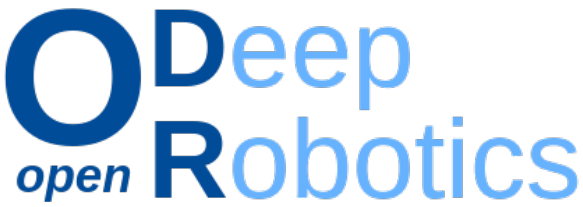


OpenDR — Open Deep Learning Toolkit for Robotics

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

Deliverable D3.3: Third report on deep human centric active perception and cognition

Date of delivery: 31 December 2022

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	**D3.3: Third report on deep human centric active perception and cognition**
Project	**OpenDR** (ICT-10-2019-2020 RIA)
Nature	Report
Dissemination Level:	**PU**blic
Authors	Moncef Gabbouj (TAU), Kateryna Chumachenko (TAU), Anton Muravev (TAU), Efstratios Kakaletsis (AUTH), Charalampos Symeonidis (AUTH), Pavlos Tosidis (AUTH), Theodoros Manousis (AUTH), Anastasios Tefas (AUTH), Nikolaos Nikolaidis (AUTH), Paraskevi Nousi (AUTH), Maria Tzelepi (AUTH), Nikolaos Passalis (AUTH), Lukas Hedegaard Morsing (AU), Negar Heidari (AU), Alexandros Iosifidis (AU)
Lead Beneficiary	TAU (Tampere University)
WP	3
Doc ID:	OPENDR_D3.3.pdf

Document History

Version	Date	Reason of change
v1.0	1/9/2022	Deliverable structure template ready
v2.0	25/11/2022	Contributions from partners finalized
v3.0	23/12/2022	Final version ready

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 third 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 worked towards developing active perception models using a multitude of methods, including active perception for face recognition (Sections 2.1 and 2.2, Section 2.3), high resolution pose estimation (Section 2.4), as well non-maximum suppression suitable for person detection methods (Section 2.5). To this end, AUTH finalized its active face recognition approach that utilizes synthesized facial views (Section 2.1). Moreover, AUTH developed 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 appropriate control commands accordingly to acquire better results (Section 2.2). It also designed a two-step pipeline that initially predicts the direction of movement and then regresses towards the full object removal from robot's point of view, enabling efficient active perception (Section 2.3). Finally, AUTH proposed two variants of its novel non-maximum suppression method for person detection (Section 2.5). The first variant was demonstrated to be more robust to visual data distribution-shifts, while the second was able to utilize existing feature maps of DL-based object detectors in an efficient manner, for computing appearance-based RoI representations, achieving top results while providing short inference times.

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, as well proposed and implemented Continual Inference Networks (CINs), which perform efficient step-by-step online processing. A brief of this work is given below.

In online tasks demanding frame-wise predictions, a considerable computational redundancy is observed in state-of-the-art 3D-CNN, ST-GCN, and Transformer architectures which receive as input a spatio-temporal clip/skeleton and perform predictions by processing overlapping clips/skeletons. To alleviate this redundancy, AU researchers have proposed a new class of 3D convolution, the *continual* 3D convolution, which can perform identical computations to those in a regular 3D convolution while considering only one time-step at a time. Existing 3D convolutional neural network architectures and weights can thus be reused and accelerated through a continual inference mode. This principle was also used to reduce the per-prediction

floating point operations of Spatio-Temporal Graph Convolutional Neural Networks (ST-GCNs) for skeleton-based action recognition by two orders of magnitude (Section 3.3). 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 18$\times$ for ST-GCNs.

Furthermore, a continual formulation of the Transformer Encoder has been proposed, which enables redundancy-free sequential processing of tokens (Section 3.1). Similarly to *continual* 3D-CNNs and *continual* ST-GCNs, *continual* Transformer Encoders achieve remarkable speedups compared to similar non-continual models, while retaining their predictive performance. 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.

The class of Continual Inference Networks (CINs), which includes the *Co*3D CNNs, the *Co*Trans, and the *Co*ST-GCNs, was formally described and accompanied with an efficient implementation (Section 3.2).

Finally, AU proposed Structured Pruning Adapters (SPAs) (Section 3.4), an alternative to fine-tuning with pruning, which adapts pre-trained weights with an extremely compressed parameter set and no modification to the source weights, while utilizing structured pruning to reduce computational complexity. Compared to fine-tuning with structured pruning, our proposed channel-SPA improves accuracy by an average of 6.9% under 90% pruning while learning half the parameters. At 70% pruning, it can learn adaptations with 17× fewer parameters with only 1.6% lower accuracy. Similarly, the proposed block-based SPA requires far fewer parameters than fine-tuning.

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

Facial expression as a fundamental natural signal for human social communication plays an important role in different applications of artificial intelligence, such as Human Computer Interaction (HCI), and healthcare. AU developed a novel methodology and its corresponding tool for image-based facial expression recognition. The work conducted by AU is briefly summarized below.

Deep Convolutional Neural Networks (CNNs) have led to considerable progress in automatic Facial Expression Recognition (FER) on laboratory-controlled datasets. However, these methods confront challenges for in-the-wild datasets where facial images come with illumination, occlusion and pose variations causing considerable change in facial appearance. AU researchers proposed a mechanism for improving the generalization ability of the state-of-the-art models on unseen samples by learning diversified facial feature representations and encouraging the learner to extract diverse spatial and channel-wise features Section 4.1. The proposed optimization mechanism is incorporated into two state-of-the-art models and improved their classification performance on three benchmark in-the-wild datasets.

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

TAU contributed a tool for heart anomaly detection (specifically, classification of atrial fibrilation from electrocardiograms) by developing a self-attention based attention mechanism for Neural Bag-of-Features formulation allowing to learn joint attention maps for over codeword

and temporal dimensions, hence improving the performance of the underlying NBoF model. The details are provided in Section 5.1.

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

Emotion recognition is one of the important tasks in human-robot interaction and multi-modal methods can be employed in this task to make use of largely available data of different types. Most of existing methods rely on pre-extracted features and assume idealistic inference environment, while such conditions are not necessarily guaranteed at inference time. To this end, TAU has contributed a tool for robust audiovisual emotion recognition, described in Section 6.1. As part of the method, a new modality fusion approach based on self-attention is proposed, resulting in improved performance compared to competing methods. Additionally, a training approach, referred to as modality dropout, is introduced, leading to significantly increased robustness of the model to incomplete or noisy data of one of the modalities.

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*

AU proposed and developed two methodologies for efficient continual human action recognition based on ST-GCNs (Section 3.3) and Transformer networks Section 3.1), which allow to reduce the number of computations compared to the standard approach of sliding window-based classification. Besides, AU formally described the class of Continual Inference Networks (CINs) for efficient online inference, and made available public implementation (Section 3.2). AU also proposed and developed an efficient image-based facial expression recognition method (Section 4.1) which uses a mechanism to diversify the features extracted by CNN layers of state-of-the-art facial expression recognition architectures for encouraging the model to learn discriminative features. The proposed method improves the model performance on in-the-wild facial expression recognition datasets. Finally, AU proposed Structured Pruning Adapters (Section 3.4), which can learn new tasks (given available pre-trained source weights) with an order of magnitude fewer parameters than fine-tuning with pruning while achieving similar predictive and computational performance.

AUTH proposed two variants of Seq2Seq-NMS (Section 2.5), a DNN-based Non-Maximum Suppression method capable of improving the performance of person detection methods. The first variant is able to operate using only geometric properties of candidate detections, while the second is able to utilize the information-rich intermediate feature maps of DL-based object detectors, for extracting appearance-based representations of the candidate detections.

TAU proposed and developed a methodology for audiovisual emotion recognition that focuses on improving the robustness of the model towards incomplete data (Section 6.1). The proposed model does not require separate feature extraction unlike the majority of the existing methods in the field, and instead relies on an end-to-end architecture that is based on lightweight video and audio branches. The proposed method outperforms competing multimodal emotion recognition methods. TAU also proposed an extension to its previously introduced Neural Bag-of-Features framework, adding a number of self-attention mechanisms, which have shown improvements in analyzing the ECG biosignal data (Section 5.1).

O1b *To provide real-time deep learning tools for robotics visual perception on high-resolution data* AUTH worked for developing high resolution pose estimation models (Section 2.4).

O2 *To leverage AI and Cognition in robotics: from perception to action*

O2a *To propose, design, train and deploy models that go beyond static computer perception, towards active robot perception*
AUTH worked towards developing active perception models for face recognition (Sections 2.1 and 2.2, Section 2.3).

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

2.1 Using Synthesized Facial Views for Active Face Recognition

2.1.1 Introduction, state of the art and work performed so far

Active vision exploits the ability of robots to interact with their environment, towards increasing the quantity / quality of information obtained through their sensors and, therefore, improving their performance in perception tasks. Active face recognition is largely understudied in recent literature. Examples of the very few active recognition methods include the simple method described in [83] that comprises of a neural network-based face recognizer along with a rather naive decision making controller that decides for the viewpoint changes and the deep learning-based active perception method for embedding-based face recognition described in [89]. During the 3rd year of the project, AUTH continued and finalized its work on active face recognition which was described in D3.2 (M24). The proposed active approach utilizes facial views produced by photorealistic facial image rendering. Essentially, the robot that performs the recognition selects the best among a number of candidate movements around the person of interest by simulating their results through view synthesis. This is accomplished by feeding the robot's face recognizer with a real world facial image acquired in the current position, generating synthesized views that differ by $\pm\theta^\circ$ from the current view and deciding, based on the confidence of the recognizer, whether to stay in place or move to the position that corresponds to one of the two synthesized views, in order to acquire a new real image with its sensor. During the current reporting period thorough experimentation and testing led to a number of improvements and fine-tuning of the method that resulted in enhanced performance. For example, a new face recognizer was utilized and the algorithm was extended to operate for a number of steps, if the obtained confidence is not acceptable. Also, instead of having the algorithm decide on the person's identity based on the last robot location, it was modified so as to decide using the position and the corresponding acquired image that provided the largest confidence among all positions that have been visited.

2.1.2 Performance evaluation

Extensive experimental performance evaluation was conducted in three facial image datasets namely the HPID dataset [30], the Queen Mary University of London Multi-view Face Dataset (QMUL)[104] and a Synthetic Dataset (SD) it has created (Section 2.3 of D6.3). These datasets consist of cropped face images of different numbers persons (15, 48 and 33 respectively) captured from various camera positions. For example, in the QMUL dataset the images cover a viewsphere of $-90^\circ ... +90^\circ$ in pan and $-30^\circ ... +30^\circ$ in tilt in 10° increments. The images of all subjects were divided into two non-overlapping subsets: a database subset G that the face recognizer uses to decide upon the ID of the query image through the nearest neighbor classifier and a query (test) subset T (these are meant to be the images captured by the robot camera in its initial position). This was done by choosing images with different pan ranges for G and T. With this setup we simulated active recognition where the robot is moving only in the pan direction. Experimental results in all three datasets verified the superior performance of the proposed method compared to the respective "static" approach, i.e. the approach where the same face recogniser is fed only with the initial image captured by the "robot". For the HPID and SD

datasets the best performance is obtained for 4 steps of the algorithm and the absolute increase of accuracy with respect to the static version is 15.61% and 13.05% respectively, whereas for the QMUL dataset the best performance is obtained for 2 steps (increase of 15.69% compared to the static approach). Moreover, comparisons were conducted with approaches that involve face frontalization. In more detail, the view synthesis algorithm was applied on the input facial image in order to synthesize a frontal view of the subject. This synthetic frontal image was then given as input to the same face recogniser. Experiments showed that the results of the proposed approach correspond to an absolute increase in accuracy (with respect to the frontalization approach) of 7.35%, 9.62% and 13.9% for the HPID, QMUL and SD datasets respectively. In addition, the proposed approach was compared to a method that uses only synthesized views generated for pan values around that of the input image. However this approach provided results inferior even to those of the static case. Finally the method was compared to the state of the art active method in [89]. The proposed approach provides (in the 4 steps setup) results that are better than that of [89] by (absolute increase in accuracy) 26.48%, 16.48% and 18.40% for the HPID, QMUL and SD datasets respectively.

Two papers describing this work were recently submitted to Springer Machine Vision and Applications Journal and the 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2023). Both can be found in Appendices 8.9 and 8.10 respectively.

- E. Kakaletsis and N. Nikolaidis, "Using Synthesized Facial Views for Active Face Recognition", *submitted to Machine Vision and Applications, Springer, 2022*, *Preprint Available at SSRN: http://dx.doi.org/10.2139/ssrn.4241482*
- E. Kakaletsis and N. Nikolaidis, "Active Face Recognition through View Synthesis", *submitted to 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2023)*

2.2 Active Vision Control Policies for Face Recognition using Deep Reinforcement Learning

2.2.1 Introduction, objectives and summary

Recent advances in Deep Learning (DL) led to a number of spectacular applications, ranging from self-driving cars and robots that outperform humans in various tasks [57]. Despite the enormous success in these areas, DL methods operate in a static fashion, i.e., they do not typically provide means for interacting with the environment in order to better perceive it. There have been several recent attempts to integrate active perception principles into DL models [83, 89]. Most of them focused on robotics tasks, where they attempt to appropriately manipulate a camera and/or a 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 [83], or use multi-view datasets to simulate some of the effects of active perception feedback [89]. 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 [89]. Another closely related line of work employs Deep Reinforcement

Learning (DRL) algorithms to perform a specific control task [78, 100, 79], e.g., acquire a frontal view of a person [119]. 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 [98]. At the same time, the lengthy training time of DRL methods further limits their applications in robotics. As a result, despite their enormous potential for developing active perception approaches their 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 appropriate control commands accordingly to acquire better results. To train and evaluate the proposed method, we developed a simulation environment using the Webots simulator [77] and generated several 3D human models using the MakeHuman software [9]. 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 by issuing the appropriate control commands. 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.

The full paper, along with the full results can be found in Appendix 8.11:

- [114] P. Tosidis, N. Passalis, and A. Tefas. "*Active Vision Control Policies for Face Recognition using Deep Reinforcement Learning*", European Signal Processing Conference 2022.

2.3 Active Perception for Occlusion Removal in Face Recognition

2.3.1 Introduction, objectives and summary

Face recognition challenges can be split into four main categories, the natural process of aging, which is uncontrollable in the sense that each person passes through different aging patterns; pose invariance, which can be potentially tackled with advanced alignment methods; severe illumination changes; and partial occlusions [1]. Further analyzing, partial occlusions refer to obstacles in the query image that block the face area by an occlusion less than 50%. The occluder object can be sunglasses, scarf, hair, masks, another person/object or even severe shadows. Partial FR is an active research topic that often uses face patches. More specifically, dynamic feature matching uses face patches and face images of a subject for reducing the intra-class variation [35]. On the other hand, newer approaches modify the network architecture using attention modules [43] that drive the model to focus on relevant parts of the occludee face. All the aforementioned solutions try to tackle the partial FR problem using a single static image representation. However, a robot agent acts in dynamic environments and has the ability to further explore its surroundings to get a better understanding.

"We do not only see, we look", is written in an early active perception and exploratory robots work [5]. More recent works argue that one of the main reasons to use active control is to see a portion of the visual field otherwise hidden due to occlusion [6]. While object detection has undergone tremendous advancements, it is still rational that problems considered ill-posed for a passive observer can be simplified when managed by an active agent [3]. Applications on active perception that enforce deep learning often use reinforcement learning [[82], [103], [8]] and are being applied to various tasks, such as image classification or adversarial scenarios detection.

Enforcing the ability of robot agents to move around, we aim to tackle the occluded face recognition problem as an active perception vision task. We propose an active perception pipeline that intents on simultaneously moving towards a direction that gives a clear face view for the robot and performing face recognition. Our proposed pipeline is two-step, initially the robot captures the face image and decides the direction of the movement towards removing the object in the most efficient way. Later, given a new face view, the agent decides how much further it should move to fully clear out the occlusion, as a ratio of the initial movement. Both the direction decision network and the regression module are implemented using efficient deep learning networks.

One could easily question why two different networks are needed, when it is fairly easy to train a feature extraction network with two heads, one predicting the movement direction and the other the movement distance. This is a rational pipeline if the problem we aimed to solve would stay in the 2D image world or we had a 3D view of the environment as an extra input. However, we consider an agent that can only acquire color information of its surroundings, as a two dimensional image. Thus, given only an RGB image as input, there is no out-of-the-box way of knowing the actual distance between the agent and the object that causes the occlusion. We aim to solve this issue by first performing a pre-defined movement towards the direction that is picked by our network. This pre-defined movement will result in different image views based on three dimensional factors such as the robot's positioning with regards to the object and the human. Our second network is trained so that it recognizes the impact of the initial movement and regresses a factor showing how much more the network should move.

A second aspect of our work that needs to be noted is that our simulation modules operate in pixel values and not three-dimensional ones. We pick this formulation as we want to make full usage of the plethora of image datasets that exist and not restrict ourselves in three dimensional models and slow simulation modules. However, our pipeline is exactly constructed for this reason, the network's output is never expected to be in pixel or distance metrics. On the contrary, we force the network to predict the remaining movement as a *ratio of the first pre-defined movement*. We also conduct a detailed mathematical analysis on why the ratio that is predicted is analogous to distance metric values in the world. Our contributions can be summed up as follows:

- We consider a real-world problem, formally describe it, and construct an active perception pipeline to solve it.
- We design a two-step pipeline that initially predicts the direction of movement and then regresses towards the full object removal from robot's point of view. We manage to use the image data after the first pre-define movement, in order to understand objects that are placed in various positions from the agent, without the need of any depth sensor.
- We follow a classic deep learning training procedure with two dimensional data, but

specifically target into a pixel agnostic solution.

- We propose a pipeline which is currently applied in face recognition, but all the modules included can be plugged in a plethora of object identification/recognition problems.

The technical report, along with the full results can be found in Appendix 8.12:

- V. Dimaridou, N. Passalis, and A. Tefas. "*Active Perception for Occlusion Removal in Face Recognition*", Technical Report (AUTH), 2022.

2.4 Active Perception for enabling Efficient High Resolution Pose Estimation

2.4.1 Introduction, objectives and summary of state of the art

Human Pose Estimation is the method of recognizing and locating the joints of humans inside a picture or a video frame. It is a computer vision problem that over the years has been challenging the science community in different ways e.g. Multi-person Pose estimation, Lightweight methods, Pose estimation in high resolution images [13],[48], [62], [86], [109]. In this section, we propose a methodology for high resolution pose estimation that allows a pose estimator to run on a high resolution image without slowing down the procedure and keeping the accuracy of poses in high levels. More specific, it works in an analogy with the human vision, where humans first take a quick look from their environment and then focus on the object that they want to. The proposed methodology (Fig. 1) is approaching pose estimation problem using an active technique where the algorithm takes advantage of a rough heatmap of the humans in the image and finds an optimal bounding box which is going to be used for further pose estimation. More specific, the proposed methodology resizes the initial image in a desired resolution by reducing the details that the image contains, and with a quick view it creates a rough heatmap with the approximate locations of humans. On the next step the method uses only the part of the image that is useful for pose estimation by using the details that provided from high resolution images without using anything extra from the environment.

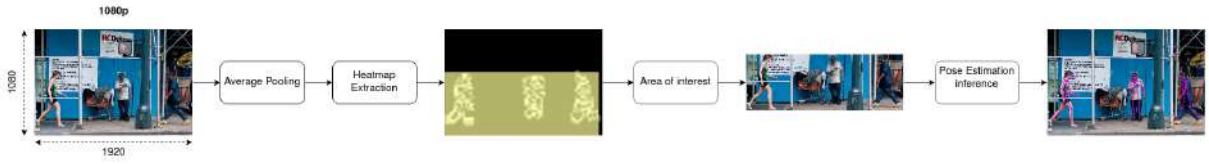


Figure 1: Proposed method pipeline

2.4.2 Summary of state of the art

In recent years there has been a significant increase in research for human pose estimation methods especially in bottom-up approaches. In the Multi-person pose estimation problem, it had been shown that bottom-up approaches are more efficient and faster than top-down techniques leading a big part of research community on this way. These methods are trying to estimate each human's pose by grouping the keypoints per person with a variety of techniques. For example, in OpenPose pose estimator [13] the research team, except for the human body's joints they also try to estimate the Part Affinity Fields (PAFs) of humans. PAFs are a set of 2D vector

fields that encode the location and orientation of limbs over the image domain. Using PAFs and keypoints for the corresponding joints it manages to makes an association for the keypoints and tries to connect them when the association metric is maximizing. Moreover, in order to make it run faster there has been implemented a lightweight version of OpenPose [86] which uses more lightweight backbones and dilated convolution and manages to reach 28 FPS running in CPU.

2.4.3 Description of work performed so far

In this section we describe the proposed method pipeline that had been done so far. Our methodology is a bottom-up approach for pose estimation and it aims to improve the results of pose estimation algorithms either in accuracy or in inference time running when it comes to high resolution images.

According to the two approaches (top-down, bottom-up), in bottom-up algorithms the whole image is passing through the algorithm in order to estimate all the joints that exist inside the image and then to append them to each person separately. However, in high resolution data each image contains a great amount of pixels and information that may not be valuable for human pose estimation. This problem could be fixed by using a top-down technique where the algorithm detects the humans, separates them and applies pose estimation on the detected bounding boxes. Following this procedure the problem of the multi-person pose estimation is still existing and the inference time is increasing as the number of humans is growing since the pose estimation is applied in each detected bounding box. Our proposed methodology is using a bottom-up approach where the pose estimation algorithm applies in a downscaled lower resolution version of the input image and creates a draft heatmap about the possible locations of the humans in the frame. Taking into account the information about humans from the generated heatmap we crop the image in the original resolution and keep only the region of interest that the estimator will be applied. At this point, we have separated the important parts of the input image, the pose estimation algorithm is applied on the new image and after finding the poses they should be a backwards projection with mathematical functions in order to fit in the original size image.

To evaluate our method we need some high resolution image datasets. Since our method is using as pose estimator the Lightweight OpenPose, which is using the COCO2017 dataset, we created three high resolution datasets from COCO2017 in 720p, 1080p and 1440p. The images from COCO2017 were patched on a new empty frame in the appropriate dimensions, this procedure created images in the desired resolution without upscale the original image and lose information from the quality decrease through upscaling. For the COCO2017 data there is an annotation file provided in which there is the ground truth keypoint coordinates for each human and each image. As it is described before after the extract of the actual poses the keypoints are projected back to the original image resolution respectively and we also provide a transformation that allows the evaluation of our predicted keypoints with the ground truth inside the annotations file provided.

This proposed method is focusing on managing the high resolution images for pose estimation and is divided in 5 steps. The first part of the methodology is the resizing of the input image. At the beginning it was used a downscale on the input image that was reducing the image size by a specific factor (0.9,0.8,0.7,0.6, etc.). From the results it is shown that there was a specific value (0.7) that the method was working more efficient both in increasing the accuracy and working in higher FPS. Following the results, it was obvious that there was a standard resolution in which the estimator was working properly. Thus, the proposed methodology is

passing the input image from an average pooling layer in order to reduce the size of the image. The kernel of the pooling layer varies according to the size of the input image, and it transforms the image into a fixed size -base height 1. The experiments that took place set this variable to 180, 360, 540, 720 pixel. During the resizing procedure the aspect ratio of the image is being preserved the same as the original image.

In the second part of the procedure the method is using the part of the Lightweght OpenPose algorithm that creates the confidence map with the joints and the part affinity fields of all the humans detected inside the low resolution image. In this way the proposed method is taking a rough look on the image about where humans are located without spending resources for a detailed examination and it produces the corresponding heatmap.

Continuing in the third part of the proposed method's procedure, in this step it is extracted the region of interest of the image. As it is shown in Fig. 2, the heatmap has highlighted the human figures and they are easily distinguishable from the background. Using the OpenCV's library for creating contours we crop the image, using a simple but efficient method, by keeping the coordinates of the top-left and bottom-right corners of the contours created. Since the previous procedure has applied on the low resolution resized image, the coordinates for the area of interest are transformed into teh original coordinate system and then the image is cropped. It should be mentioned that there might be cases that the cropped and the original image have similar size e.g., people scattered inside the frame, but besides these exceptions the method seems to outperform the Lightweight OpenPose (LwOP).

In the fourth part of the proposed method we try to extract the human poses. In this step there is an additional resizing on the image that comes as an input (cropped image) before the pose estimation takes place. The resizing step is implemented as the previous one and it is tested in the same height values (180, 360, 540, 720) as before for all the possible combinations. To complete the experiments, there was also tested one more case, in which the cropped image didn't resized and passed through the pose estimator on its fixed resolution.

The last step of the procedure is the backwards projection of the predicted keypoints to the original coordinates. Since the pose estimation was applied on a cropped and resized image, the keypoints that were predicted should projected on the original image. This procedure contains a mathematical transformation going back step by step on the changes we made on the original image.

2.4.4 Performance evaluation

For the evaluation of the proposed High Resolution Pose Estimation, all the experiments conducted on the COCO2017 dataset or in some variations of COCO2017 in 720p,1080p,1440p as it explained before. The dataset contains 5000 images and around 54% (2693) of them are including at least one human. The experiments that took place were a combination of the resizing scale in first and in second inference. Moreover, since our proposed method is based in the preparation of the high resolution images in order to be used from the pose estimator, there is no need of further training. At first it was needed to have some baseline results to compare this high resolution method. We used the Lightweight OpenPose on each dataset and gathered the results for the average precision 0.5:0.95 and the FPS as it can be seen in Table 1.

From the previous results it is shown that the pose estimator can maintain its average precision while dropping dramatically the inference time if it is not using any resizing on the input images and on the other hand it can maintain its inference time close to real-time but with a big cost in average precision.

Figure 2: Generated heatmaps from input images

Table 1: Lightweight OpenPose on various datasets with and without resizing input image in base_height 368 pxl

	Average Precision 0.5:0.95	
Dataset	**LwOP with resizing**	**LwOP without resizing**
COCO	0.4	0.425
720p	0.288	0.422
1080p	0.172	0.424
1440p	0.111	cuda memory error

	FPS	
Dataset	**LwOP with resizing**	**LwOP with resizing**
COCO	35	24.51
720p	27	6.601
1080p	23	2.76
1440p	20	cuda memory error

The experiments that conducted to evaluate this method were using the same datasets and we were tracking the same metrics (average precision and FPS). The results of the experimental process are shown in Table 2.

From the Table 2, we keep only the best combinations of downscaling resolutions that combine high average precision scores and FPS. The best results are shown in the Table 3

Table 2: Evaluation of the proposed method on different resolutions. During the experiments different combinations of image height have been applied for the first and the second part of the procedure.

Avg_precision 0.5:0.95
COCO

		base height 2			
		180	**360**	**540**	**720**
base height 1	**180**	0.253	0.387	0.368	0.311
	360	0.268	0.415	0.399	0.33
	540	0.215	0.395	0.431	0.393
	720	0.203	0.393	0.439	0.413

FPS
COCO

		base height 2			
		180	**360**	**540**	**720**
base height 1	**180**	68	49	30	18
	360	50	35	20	11
	540	44	28	16	9
	720	41	27	15	8

Avg_precision 0.5:0.95
720p

		base height 2			
		180	**360**	**540**	**720**
base height 1	**180**	0.175	0.313	0.34	0.31
	360	0.164	0.359	0.425	0.428
	540	0.16	0.352	0.426	0.427
	720	0.16	0.352	0.426	0.427

FPS
720p

		base height 2			
		180	**360**	**540**	**720**
base height 1	**180**	89	65	43	26
	360	47	31	17	9.7
	540	23	18	12	7
	720	23	18	12	7

Avg_precision 0.5:0.95
1080p

		base height 2			
		180	**360**	**540**	**720**
base height 1	**180**	0.102	0.186	0.204	0.195
	360	0.127	0.312	0.393	0.408
	540	0.148	0.348	0.434	0.441
	720	0.099	0.275	0.386	0.43

FPS
1080p

		base height 2			
		180	**360**	**540**	**720**
base height 1	**180**	97	78	61	40
	360	54	40	25	14
	540	31	23	14	8
	720	12	11	8	6

Avg_precision 0.5:0.95
1440p

		base height 2			
		180	**360**	**540**	**720**
base height 1	**180**	0.067	0.106	0.113	0.105
	360	0.114	0.274	0.334	0.342
	540	0.117	0.311	0.407	0.445
	720	0.117	0.311	0.407	0.445

FPS
1440p

		base height 2			
		180	**360**	**540**	**720**
base height 1	**180**	124	106	82	57
	360	67	54	38	24
	540	23	18	13	8
	720	23	18	13	8

2.4.5 Conclusions and Future Work

To summarize, the proposed high resolution pose estimation method seems to manage to increase the average precision and the FPS when input images are in high resolutions. A detail that needs to be explained, is that it can be seen that images in 1440p run faster than 1080p. This anomaly is due to the bigger difference in size between the original and the resized one.

Table 3: Comparing the proposed method to different baselines

	Average Precision 0.5:0.95			
Dataset	**LwOP**	**Proposed method base_height 1,2 360,360**	**Proposed method base_height 1,2 360,540**	**Proposed method base_height 1,2 360,720**
COCO	0.4	0.415	0.399	0.33
720p	0.288	0.359	0.425	0.428
1080p	0.172	0.312	0.393	0.408
1440p	0.111	0.274	0.334	0.342

	FPS			
Dataset	**LwOP**	**Proposed method base_height 1,2 360,360**	**Proposed method base_height 1,2 360,540**	**Proposed method base_height 1,2 360,720**
COCO	35	35	20	11
720p	27	31	17	9.7
1080p	23	40	25	14
1440p	20	54	38	24

When larger images are down-scaled the objects in the frame become smaller and the algorithm can not detect the human figures, so the image is over-passed without any further processing and this is increasing the speed of the algorithm. As future work, we plan to develop more efficient approaches for processing high resolution images for generating the initial heatmap generator, e.g., by using models that can handle high resolution images in real time [118], as well as develop approaches to more precisely localize the area of interest.

2.5 Feature Selection for Attention-based Non-Maximum Suppression

2.5.1 Introduction objectives and summary of state of the art

Non-Maximum Suppression (NMS) is a post-processing step incorporated in almost every visual object detector, tasked with rapidly pruning the number of overlapping detected candidate rectangular Regions-of-Interest (RoIs) and replacing them with a single, more spatially accurate detection (in pixel coordinates). The problem it attempts to solve arises from the tendency of many detectors to output multiple, neighbouring candidate object RoIs for a single given visible object, due to their implicit sliding-window nature. NMS methods typically rescore the raw candidate detections/RoIs outputted by the detector, before thresholding these modified scores so that, ideally, only a single RoI is finally retained for each visible object. A typical case where most NMS methods struggle to perform is when they operate on images depicting objects in complex scenes, where several in-between occlusions appear. This occurs frequently when detecting persons in crowded scenes. This is a very important scenario for security- or safety-critical applications. The vast majority of existing methods only exploit geometric properties/interrelations between the candidate RoIs, in the form of geometric features.

2.5.2 Description of work performed so far and performance evaluation

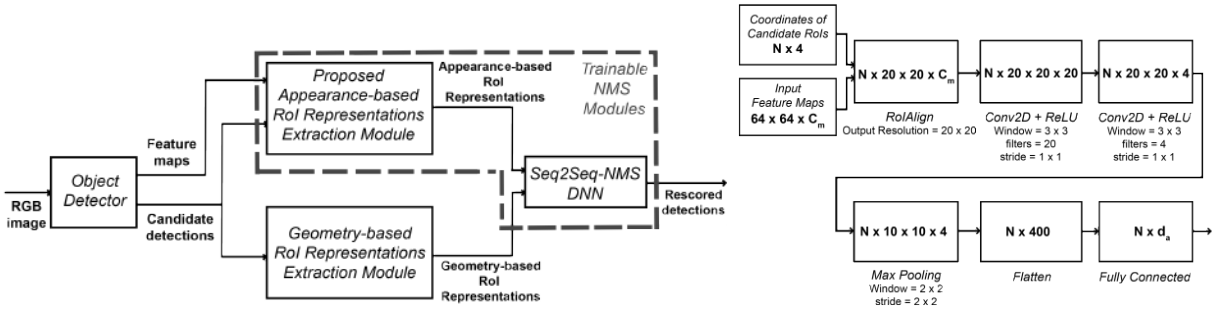
In Section 2.3 of D6.2 (M24), AUTH presented a deep neural architecture which approached NMS as a sequence-to-sequence problem. The presented method, called Seq2Seq-NMS, extracts RoI representations based on geometric *and* visual appearance properties of the input candidate RoIs. An efficient implementation of FMoD [73] was employed for visual RoI description. These RoI representations are then refined by the Seq2Seq-NMS, by capturing relations of neighboring RoIs and aiming to ideally assign precisely one detection per person. In [111], Seq2Seq-NMS was able to achieve top precision results on three separate person detection datasets.

In this reporting period, AUTH examined whether Seq2Seq-NMS is susceptible to visual data distribution shifts, due to the fact that the method extracts RoI representations based on their visual appearances, in contrast to other NMS methods which incorporate only geometry-based RoI representations. A visual data distribution shift scenario was simulated, in which the depicted environments of the test set were visually dissimilar from those on the training set. As expected, this distribution shift had a major negative effect to the performance of Seq2Seq-NMS, compared to the performance of the methods that incorporate only geometry-based features that are affected to a lesser extent. A new variant of Seq2Seq-NMS was proposed for such cases, exploiting only the geometric properties of the candidate RoIs. This variant, named Seq2Seq-NMS$_{geom}$, was implemented by feeding the DNN a zero vector for each ROI, as a dummy appearance-based representation vector. In the aforementioned visual data distribution shift scenario, Seq2Seq-NMS$_{geom}$ attained improvements of +0.3% and + 1.1% against the original Seq2Seq-NMS in $AP_{0.5}$ and $AP_{0.5}^{0.95}$ respectively.

A conference paper describing this work was presented at IEEE ICIP 2022 and can be found in Appendix 8.13:

- [112] C. Symeonidis, I. Mademlis, I. Pitas and N. Nikolaidis, "*AUTH-PERSONS: A Dataset for Detecting Humans in Crowds from Aerial Views*", IEEE International Conference on Image Processing (ICIP), 2022

In addition AUTH also proposed a new variant of Seq2Seq-NMS, which was named FSeq^2-NMS. The proposed variant is able to harness the information-rich intermediate feature maps of DL-based object detectors. These intermediate feature maps are used to derive learned, high-level, semantically meaningful RoI representations, which are then exploited *instead of* handcrafted visual descriptors (such as [73]). The efficacy achieved by the internal/latent image representations of state-of-the-art detectors allows the method to discriminate duplicate RoIs from a set of densely sampled and heavily occluded candidate detections, a problem commonly encountered when detecting humans in crowded scenes. FSeq^2-NMS can be easily plugged on top of any DL-based detector, and trained as a separate sub-module. The structure of the overall object detection framework, in which FSeq^2-NMS is employed, is depicted in Figure 3a. An appearance-based ROI representations extraction module was implemented for processing the detector's feature maps. An illustration of this module is depicted on Figure 3b. As input, the module receives $\mathbf{B} = [\mathbf{b}_0, \mathbf{b}_1, .., \mathbf{b}_N] \in \mathbb{R}^{N\times 4}$, i.e., the coordinates of N candidate RoIs, as well as $\mathbf{M} \in \mathbb{R}^{64\times 64\times C_m}$, namely a set of features maps, extracted from an in-between layer of the deployed detector and resized to a fixed 64×64 resolution (C_m is the number of channels of the corresponding feature maps). Using the *RoIAlign* operator [34], initial RoI maps can be in-parallel extracted in a fixed 20×20 spatial resolution. Then two convolutional layers, with the Rectified Linear Unit (ReLU) as activation function, followed by a max-pooling layer are



(a) (b)

Figure 3: a) The pipeline of the overall object detection framework in which FSeq^2-NMS is employed. b) The appearance-based RoI representations extraction module.

applied on the extracted RoI maps. The final RoI representations $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, .., \mathbf{a}_N] \in \mathbb{R}^{N \times d_a}$ are computed by flattening the RoI maps and applying a fully connected layer using ReLU as activation function (d_a corresponds to the dimension of the final appearance-based RoI representations).

Experiments were conducted on two public person detection datasets, widely used in the task of detecting humans in crowded scenes. In the testing set of PETS dataset, FSeq^2-NMS surpassed all competitive NMS methods by +0.5% and +0.1% in $AP_{0.5}$ and $AP_{0.5}^{0.95}$ respectively. In the more challenging CrowhHuman dataset, the proposed method surpassed all competitive NMS methods by +2.9% and +1.9% in $AP_{0.5}$ and $AP_{0.5}^{0.95}$ respectively. In both datasets, the proposed implementation achieved faster than real-time inference times (>120fps). This work has been submitted to ICASSP 2023 and can be found in Appendix 8.14

- C. Symeonidis, I. Mademlis, I. Pitas and N. Nikolaidis, "*Efficient Feature Extraction for Non-Maximum Suppression in Visual Person Detection*", submitted to IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2023)

2.6 Real-time synthetic-to-real human detection for robotics applications

2.6.1 Introduction, objectives and summary of state of the art

During the 3rd year of OpenDR's project research, AUTH also worked on synthetic-to-real-human detection focusing on robotics applications. Generally, the use of synthetic data is accompanied by various benefits linked with their low-cost nature and ability to meet specific requirements imposed by the application, which may not be feasible in real data. Therefore, synthetic data have been utilized in a wide spectrum of robotics applications, e.g., [139]. A key issue associated with the successful use of synthetic data in robotics is the gap between the generated data and their deployment considering real data, that is, the so-called synthetic-real gap. The need for bridging this gap has fueled a new research area [129]. In this work, AUTH first created a synthetic dataset for discriminating between humans and non humans, and then utilized it to train a lightweight fully convolutional model, capable of operating in real-time (about 25 frames per second) on a low-power GPU for high resolution input [118]. The target was to use the model to provide semantic heatmaps of human presence on real data. That is, AUTH trained the real-time model on the synthetic data, and tested the model on unseen images that

contain real humans, producing semantic heatmaps, as explained in [118]. A main objective of this work was to assess the generalization of the model to real data, and investigate the effect of using real images in the training phase. As it was demonstrated in the experimental evaluation the use of even few real training examples can considerably ameliorate the performance of training merely with synthetic data, while this was also reflected in the qualitative evaluation through the produced heatmaps.

This work has been presented to IISA 2022 and can be found in Appendix 8.15:

- M. Tzelepi, C. Symeonidis, N. Nikolaidis and A. Tefas. "*Real-time synthetic-to-real human detection for robotics applications*", IEEE International Conference on Information, Intelligence, Systems and Applications (IISA) , 2022.

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

3.1 Continual Transformers

3.1.1 Introduction and objectives

Originally proposed for Natural Language Processing tasks, Transformers [120] have emerged as powerful architectures for computer vision [24, 88, 84, 4] and time-series processing [124, 23]. Due to the mutual information exchange between input tokens as they are processed in the network, their application to online processing of time-series has been limited to the processing of a sliding window with significant computation cost. In our work, we have identified the computational redundancies of Transformer Encoders in the online setting, and rectified the shortcomings with our proposed Continual Transformer Encoders. For one- and two-block architectures, our method produces identical results to the original Transformer Encoder formulation, and greatly accelerate inference through novel formulations of the Scaled Dot-product Attention (SDA), which reduce the per time-step computational complexity from $\mathcal{O}(n^2d)$ to $\mathcal{O}(nd)$ and memory complexity $\mathcal{O}(n^2)$ to $\mathcal{O}(nd)$.

3.1.2 Summary of state of the art

The improvement of Transformer efficiency has been explored by many prior works [113]: Prior approaches include chunking methods [88, 24]; sliding windows, dilation and pooling [10]; intra-attention groupings [53]; and approximations to the self-attention [123, 126, 17]. Unlike the above-refenced works, our approach produces the exact same results for temporal sequences as the original Transformer Encoder [120] while providing computational speedups.

3.1.3 Description of work performed so far

At the basis of Transformer Encoders lies the Scaled Dot-product Attention. In order to accommodate step-wise updated attention outputs, we propose two SDA variants: The Continual Retro-active SDA, which updates prior attention outputs retroactively when a new token arrivies; and Continual Single-output SDA, which only computes an attention results for the latest token. For further details, we refer the reader to Section 3 in our paper found in Appendix

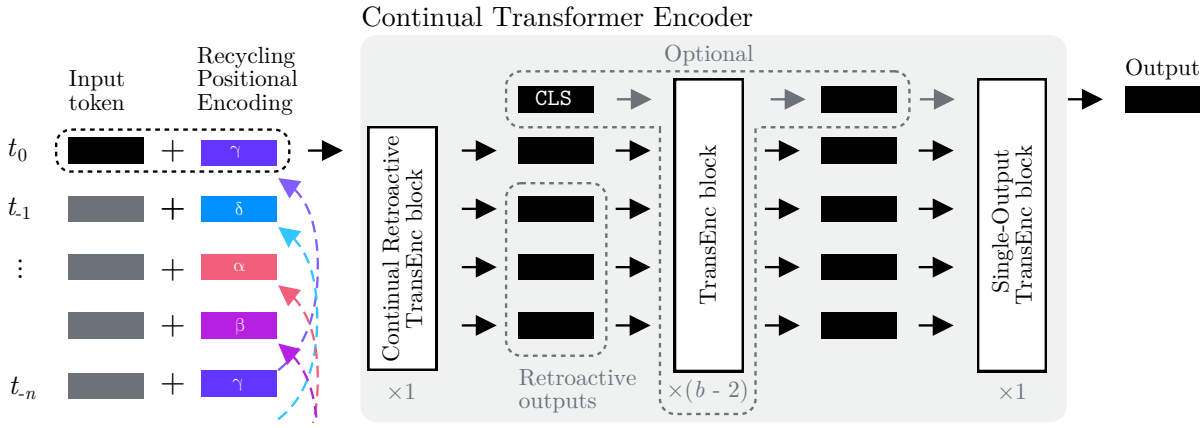


Figure 4: Multi-block Continual Transformer Encoder with Recycling Positional Encoding. For $b > 2$ blocks, regular Transformer Encoder blocks can be added between an initial Continual Retroactive block and a final Single-Output block. A class-token may be used after the initial block.

8.3. Corresponding to the regular SDA, our proposed Continual SDA's can be used in a Continual Multi-head Attention (MHA), as well as full Transformer Encoder blocks and sequences thereof. The multi-block Continual Transformer Encoder, however, is limited to use a Continual Retro-active MHA withing the first block, and Continual Single-output MHA in the last, with regular MHA utilised in the remaining blocks. This is illustrated in Fig. 4. Moreover, positional encodings must follow a token as time progresses. We propose a Recycling Positional Encoding to accommodate this.

The principles, properties and limitations of Continual Transformers are described in more detail in the associated paper, which can be found in Appendix 8.3:

- [39] L. Hedegaard, A. Bakhtiarnia and A. Iosifidis, "*Continual Transformers: Redundancy-Free Attention for Online Inference*", Advances in Neural Information Systems (NeurIPS) Workshop on Vision Transformers: Theory and Applications, 2022.

3.1.4 Performance evaluation

In the evaluation of the Continual Transformer Encoders, we perform experiments THUMOS14 [47] and TVSeries [22] datasets for Online Action Detection [22] as well as on the Music Genre Classification dataset GTZAN [117]. In our evaluation, we train Continual versions of prior state-of-the-art Transformer architectures (e.g. OadTR [124] for Online Action Detection) and evaluate their task performance alongside the Floating Point Operations (FLOPs) per prediction. The results of this are shown in Tables 4 an 5.

For THUMOS14 and TVSeries (Table 4), our proposed (*Co*)OadTR-b# architecture exhibit similar precision while achieving significantly reduced FLOPs per prediction. More specifically, *Co*OadTR-b1 and *Co*OadTR-b2 reduce FLOPs by 255× and 6.1×, respectively, compared to OadTR. On GTZAN, Continual Transformers achieve similar accuracy as their regular counterparts while using 1.76× less FLOPs with two blocks and 51.5× less FLOPs when using one Transformer Encoder block.

For further details on the experimental setup and ablation studies, we refer the reader to the full paper in Appendix 8.3.

Table 4: Online Action Detection results. FLOPs per prediction are noted for inference on THUMOS14. The **best** and ***next-best*** metrics are highlighted. †Using official source code or modifications there-off.

Model	Feat.	THUMOS14 mAP (%)	TVSeries mcAP (%)	FLOPs (M)
RED [29]	A.Net	45.3	79.2	-
TRN [127]		47.2	83.7	1387.5
FATS [51]		51.6	81.7	-
IDN [26]		50.0	84.7	-
TFN [27]		55.7	85.0	-
LSTR [128]		**65.3**	88.1	-
OadTR [124]		***58.3***	85.4	2445.6
OadTR†		57.0±0.5	**88.6**±0.1	2445.6
OadTR-b2†		56.6±0.3	***88.3***±0.2	1008.1
OadTR-b1†		56.3±0.2	88.1±0.1	605.5
*Co*OadTR-b2 (ours)		56.8±0.4	87.7±0.6	***410.9***
*Co*OadTR-b1 (ours)		56.1±0.7	87.6±0.7	**9.6**
TRN [127]	Kin.	62.1	86.2	1462.0
FATS [51]		59.0	84.6	-
IDN [26]		60.3	86.1	-
PKD [138]		64.5	86.4	-
LSTR [128]		**69.5**	**89.1**	-
OadTR [124]		***65.2***	87.2	2513.5
OadTR†		64.2±0.3	***88.6***±0.1	2513.5
OadTR-b2†		64.5±0.5	88.3±0.2	1075.7
OadTR-b1†		63.9±0.5	88.1±0.1	673.0
*Co*OadTR-b2 (ours)		64.4±0.1	87.6±0.7	***411.9***
*Co*OadTR-b1 (ours)		64.2±0.4	87.7±0.4	**10.6**

Table 5: Audio Classification results for GTZAN.

Method	Pos. Enc.	Acc. (%)	FLOPs (M)	Par. (K)
Maj. Voting	-	92.0	-	**0**
Trans-b2	learned	**95.0**±0.6	47.4	509
Trans-b1	learned	93.8±0.8	15.2	***286***
*Co*Trans-b2	fixed	***94.4***±1.0	27.0	509
*Co*Trans-b1	learned	93.2±1.1	***0.3***	***286***

3.2 Continual 3D Convolutional Neural Networks

3.2.1 Introduction and objectives

Continual 3D Convolutional Neural Networks (Co3D CNNs) are a novel computational formulation of spatio-temporal 3D CNNs, in which videos are processed frame-by-frame rather than by clip. In online tasks demanding frame-wise predictions, Co3D CNNs omit the computational redundancies of regular 3D CNNs, namely the repeated convolutions over frames, which appear in overlapping clips.

Continual 3D CNNs can reuse preexisting 3D-CNN weights to reduce the per-prediction floating point operations (FLOPs) in proportion to the temporal receptive field while retaining similar accuracy. This was validated with multiple models on Kinetics-400 and Charades with good results: CoX3D models attain state-of-the-art complexity/accuracy trade-offs on Kinetics-400 with 12.1-15.3x reductions of FLOPs and 2.3-3.8% improvements in accuracy compared to regular X3D models while reducing peak memory consumption by up to 48%. The full paper describing this work [38] was recently accepted at the European Conference on Computer Vision and can be found in Appendix 8.1.

Continual Inference is a Python library for implementing Continual Inference Networks (CINs), a class of Neural Networks designed for redundancy-free online inference. The library is drop-in replacement for PyTorch, which augments modules with the ability of efficient online inference. Moreover, it contains a functional-API, which let's it's users create complex neural networks that poses the CIN capabilities of efficient step-by-step processing, which can and has been used to reduce the floating point operations of prior networks by more than an order of magnitude.

The full paper describing this library in more detail can be found in Appendix 8.2:

- [39] L. Hedegaard and A. Iosifidis, "*Continual Inference: A Library for Efficient Online Inference with Deep Neural Networks in PyTorch*", International Workshop on Computational Aspects of Deep Learning (ECCV Workshop), 2022.

3.3 Continual Spatio-Temporal Graph Convolutional Networks for Online Skeleton-based Human Action Recognition

3.3.1 Introduction and objectives

Skeleton-based action recognition methods process a sequence of skeletons, representing a sequence of body poses encoding the action in a video, to recognize the performed action. Graph Convolutional Networks (GCNs) have shown significant progress in processing skeleton data for human action recognition. ST-GCN [130] was the first GCN-based method proposed for skeleton-based action recognition, which uses spatial and temporal graph convolutions to extract the per time-step features and also time-varying dynamics of skeletons. Recently, many methods have been extending ST-GCN to enhance its efficiency and performance. Unfortunately, the high computational complexity of these GCN-based methods makes them inapplicable in real-time scenarios and resource-constrained online inference settings, where the input is a continual stream of skeletons and step-by-step predictions are required. We proposed CoSTGCN method to reduce such redundant computations by reformulating the ST-GCN and its derived methods as a Continual Inference Network using Continual Convolutions in place of regular ones for aggregating temporal information. In this setting, the skeletons are processed one by one and

the model produces updated predictions for each time-step. Our continual models achieve up to 108× FLOPs reduction, 26× throughut increase, and 52% reduction in max allocated GPU memory compared to the corresponding non-continual models.

3.3.2 Summary of state of the art

GCN-NAS [91] and PST-GCN [40] are neural architecture search based methods which try to find an optimized ST-GCN architecture to increase the efficiency of the classification task; ShiftGCN [15] replaces graph and temporal convolutions with a zero-FLOPs shift graph operation and point-wise convolutions as an efficient alternative to the feature-propagation rule for GCNs [52]; ShiftGCN++[16] boost the efficiency of ShiftGCN further via progressive architecture search, knowledge-distillation, explicit spatial positional encodings, and a Dynamic Shift Graph Convolution. However, all the above-mentioned methods would need to rely on sliding window-based processing for online inference, and then performing their prediction on the whole sequence.

3.3.3 Description of work performed so far

With regular temporal convolutions, features produced by multiple blocks cannot be trivially disentangled and cached in time. Accordingly online operation with per-skeleton predictions can be attained by caching $T-1$ prior skeletons, concatenating these with the newest skeleton, and performing regular spatio-temporal inference. However, this comes with significant computational redundancy, where the complexity of online frame-wise inference is the same as for clip-based inference. To alleviate this issue, we propose to employ Continual Convolutions, Figure 5, in the temporal modeling of Spatio-temporal Graph Convolutional Networks. By restricting the graph convolution function to only operate locally within a time-step, we defined a *Continual* Spatio-Temporal block by replacing the original temporal 2D convolution with a continual one, To retain weight-compatibility with regular (non-continual) networks we moreover need to delay the residual to keep temporal alignment, Figure 6. For more details on the model architecture and training process, we refer the reader to the preprint appended in Appendix 8.4:

- [37] L. Hedegaard, N. Heidari and A. Iosifidis, "*Online Skeleton-based Action Recognition with Continual Spatio-Temporal Graph Convolutional Networks*", arXiv:2203.11009, 2022.

3.3.4 Performance evaluation

We conducted experiments on three benchmark datasets, NTU-RGBD-60 [101], NTU-RGBD-120 [66], Kinetics-Skeleton-400 [49], to evaluate our method. We compared the performance of our method with state-of-the-arts in Tables 6, 7, 8. The experimental results show up to 26× on-hardware speedups, 109× reduction in FLOPs per prediction, and 52% reduction in maximum memory allocated memory during online inference with similar accuracy to those of the original networks. Our proposed architectural modifications are generic in nature and can be used for many methods in skeleton-based action recognition.

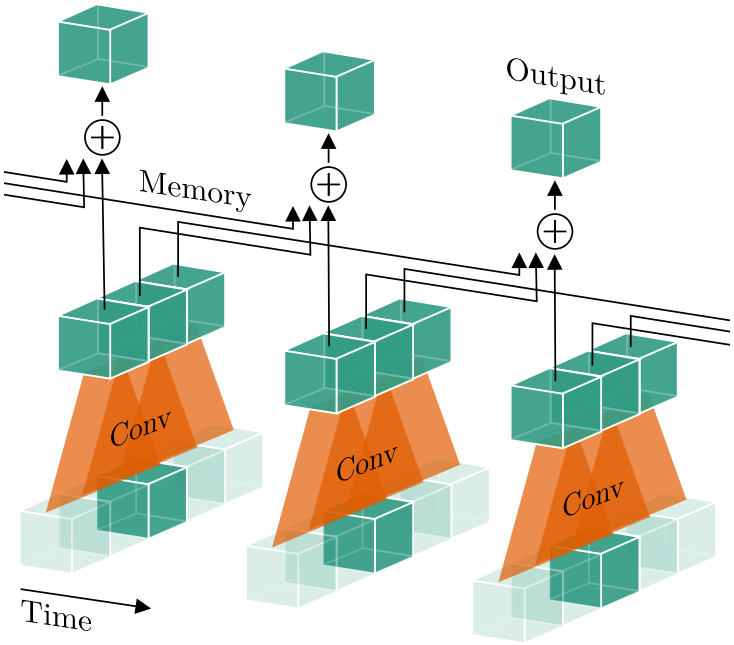


Figure 5: **Continual Convolutions** are performed in two stages: First, the input is zero-padded and convolved with the convolutional kernel ($K = 3$ in illustration) to produce intermediary results. Subsequently, these are cached and summed up to produce the final output.

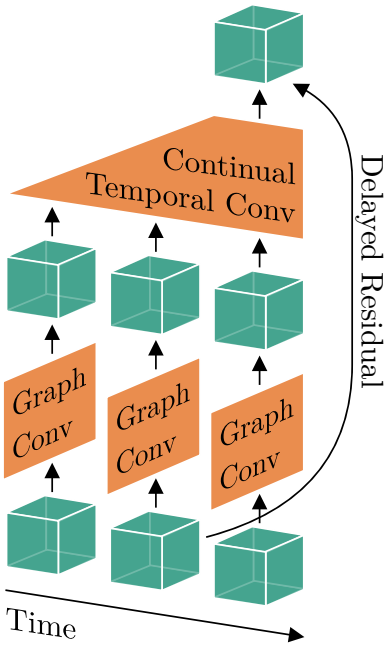


Figure 6: **Continual Spatio-temporal Graph Convolution Blocks** consist of an in-time Graph Convolution followed by an across-time Continual Convolution (here a kernel size of three is depicted). The residual connection is delayed to ensure temporal alignment with the continual temporal convolution that is weight-compatible with non-continual networks.

Table 6: **NTU RGB+D 60** comparison with recent methods, grouped by clip- and frame-based inference. Noted are the number of streams (S.), top-1 validation accuracy, and FLOPs per prediction. †Results for our implementation. Highlights indicate **best**, ***next-best*** results.

	Model	S.	Accuracy (%)		FLOPs
			X-Sub	X-View	(G)
Clip	SGN [137]	1	89.4	94.5	-
	MS-G3D [69]	1	89.4	95.0	-
		2	**91.5**	96.2	-
	ST-TR [93]	1	89.2	95.8	-
		2	90.3	***96.3***	-
	MS-AAGCN [106]	4	90.0	96.2	-
	Hyper-GNN [32]	3	89.5	95.7	-
	FGCN [131]	4	90.2	***96.3***	-
	DGNN [105]	4	89.9	96.1	126.80
	AS-GCN [60]	1	86.8	94.2	27.00
	AGC-LSTM [107]	2	89.2	95.0	54.40
	ShiftGCN [15]	1	87.8	95.1	2.50
		2	89.7	96.0	5.00
		4	***90.7***	**96.5**	10.00
	ShiftGCN++ [16]	1	87.9	94.8	**0.40**
		2	89.7	95.7	***0.80***
		4	90.5	***96.3***	1.70
	ST-GCN†	1	86.0	93.4	16.73
		2	88.1	94.9	33.46
	AGCN†	1	86.4	94.3	18.69
		2	88.3	95.3	37.38
	S-TR†	1	86.8	93.8	16.20
		2	89.1	95.3	32.40
Frame	Deep-LSTM [101]	1	60.7	67.3	-
	VA-LSTM [136]	1	79.2	87.7	-
	*Co*ST-GCN (ours)	1	86.0	93.4	0.27
		2	88.1	94.8	0.54
	*Co*ST-GCN* (ours)	1	86.3	93.8	***0.16***
		2	88.3	***95.0***	0.32
	*Co*AGCN (ours)	1	86.4	94.2	0.30
		2	88.2	**95.3**	0.60
	*Co*AGCN* (ours)	1	84.1	92.6	0.22
		2	86.0	93.1	0.44
	*Co*S-TR (ours)	1	86.5	93.5	0.17
		2	**88.8**	**95.3**	0.34
	*Co*S-TR* (ours)	1	86.3	92.4	**0.15**
		2	***88.9***	94.8	0.30

Table 7: **NTU RGB+D 120** comparison with recent methods, grouped by clip- and frame-based inference. Noted are the number of streams (S.), top-1 validation accuracy, and FLOPs per prediction. †Results for our implementation. Highlights indicate **best**, ***next-best*** results.

	Model	S.	Accuracy (%)		FLOPs
			X-Sub	X-Set	(G)
Clip	Part-Aware LSTM [67]	1	25.5	26.3	-
	ST-LSTM [68]	1	55.7	57.9	-
	TSRJI [11]	1	67.9	62.8	-
	SGN [137]	1	79.2	81.5	-
	MS-G3D [69]	2	**86.9**	**88.4**	-
	FGCN [131]	4	85.4	87.4	-
	ShiftGCN [15]	1	80.9	83.2	2.50
		2	85.3	86.6	5.00
		4	***85.9***	***87.6***	10.00
	ShiftGCN++ [16]	1	80.5	83.0	**0.40**
		2	84.9	86.2	***0.80***
		4	85.6	87.2	1.70
	ST-GCN†	1	79.0	80.7	16.73
		2	83.7	85.1	33.46
	AGCN†	1	79.7	80.7	18.69
		2	84.0	85.4	37.38
	S-TR†	1	80.2	81.8	16.20
		2	84.8	86.2	32.40
Frame	*Co*ST-GCN (ours)	1	78.9	80.7	0.27
		2	83.7	85.1	0.54
	*Co*ST-GCN* (ours)	1	79.4	81.6	***0.16***
		2	84.0	85.5	0.32
	*Co*AGCN (ours)	1	79.6	80.7	0.30
		2	84.0	85.3	0.60
	*Co*AGCN* (ours)	1	77.3	79.1	0.22
		2	80.4	82.0	0.44
	*Co*S-TR (ours)	1	80.1	81.7	0.17
		2	***84.8***	**86.1**	0.34
	*Co*S-TR* (ours)	1	79.7	81.7	**0.15**
		2	**84.8**	***86.1***	0.30

Table 8: **Kinetics Skeleton 400** comparison with recent methods, grouped by clip- and frame-based inference. Noted are the number of streams (S.), top-1 and top-5 validation accuracy, and FLOPs per prediction. †Results for our implementation. Highlights indicate **best**, ***next-best*** results.

	Model	**S.**	**Accuracy (%)**		**FLOPs**
			Top-1	Top-5	(G)
Clip	Feature Enc. [28, 130]	1	14.9	25.8	-
	Deep LSTM [102, 130]	1	16.4	35.3	-
	TCN [50, 130]	1	20.3	40.0	-
	AS-GCN [60]	1	34.8	56.5	-
	ST-GR [58]	1	33.6	56.1	-
	DGNN [105]	4	***36.9***	59.6	-
	MS-G3D [69]	2	**38.0**	***60.9***	-
	MS-AAGCN [106]	4	37.8	**61.0**	-
	Hyper-GNN [32]	3	37.1	60.0	-
	ST-GCN†	1	33.4	56.1	***12.04***
		2	34.4	57.5	24.09
	AGCN†	1	35.0	57.5	13.45
		2	***36.9***	59.6	26.91
	S-TR†	1	32.0	54.9	**11.62**
		2	34.7	57.9	23.24
Frame	*Co*ST-GCN (ours)	1	31.8	54.6	0.16
		2	33.1	56.1	0.32
	*Co*ST-GCN* (ours)	1	30.2	52.4	**0.11**
		2	32.2	54.5	0.22
	*Co*AGCN (ours)	1	33.0	55.5	0.18
		2	**35.0**	**57.3**	0.36
	*Co*AGCN* (ours)	1	23.3	44.3	***0.12***
		2	27.5	49.1	0.25
	*Co*S-TR (ours)	1	29.7	52.6	0.16
		2	***32.7***	***55.6***	0.31
	*Co*S-TR* (ours)	1	27.4	49.7	**0.11**
		2	29.9	52.7	0.22

3.4 Structured Pruning Adapters

3.4.1 Introduction and objectives

The reuse of pretrained source weights to improve performance on related target tasks has been utilized ubiquiously in Deep Learning for visual data analysis. Commonly, this is done by continuing training (fine-tuning) of source weights on the new data. Despite target tasks of simpler nature, the resulting models always have the same parameter count and computational complexity as the source model. Considering the (often simpler) nature of the target task, it is reasonable to assume, that a derived parameter set and network structure could be learned, which reduces both the computational complexity and parameter count while achieving similar predictive performance on the target task.

3.4.2 Summary of state of the art

Adapter-based methods [95, 96, 33, 142, 64, 44, 92, 74] propose an alternative means of adaptation compared to fine-tuning: instead of modifying the source weights, they define light-weight add-on modules, which are trained. Assuming that the source model weights are available at deployment, adapters can store the additional knowledge needed for a new task with an order of magnitude fewer parameters.

However, just like the fine-tuning approach, the computational complexity of the specialized models remains as high (or higher) than that of the source model. Pruning methods can be used to both the reduce the parameter count and computational complexity of models. While unstructured pruning approaches produce sparse weights, which may need specialized hardware to achieve accelerated inference in practice, structured pruning of either channels [59, 80, 110, 132] or blocks of weights [99, 56] accelerate networks on commodity hardware as well.

3.4.3 Description of work performed so far

We propose Structured Pruning Adapters (SPAs), the combination of tailor-made adapters with structured pruning. Our learned adapters can be fused with the frozen source weights to avoid run-time overhead and are affected by the same pruning masks as the source weights. This stands in contrast to prior adapter methods, where custom logic must be configured to prune adapters.

We propose a channel-based SPA and a block-based SPA. Our channel-SPA augments the Low-rank Adapter [45] with structured pruning of input- and output-channels. In channel-pruning, the weight mask $\mathbf{M} \in \{0,1\}^{n \times m}$ is decomposed as row and column masks $\mathbf{m}_{row} \in \{0,1\}^{n \times 1}$ and $\mathbf{m}_{col} \in \{0,1\}^{m \times 1}$, respectively. The adapter target weights $\mathbf{M}_t$ can then be adapter from the source weights $\mathbf{M}_s$ via the Structured Pruning Low-rank Adapter (SPLoRA):

$$\mathbf{W}_t = \mathbf{W}_s \odot \mathbf{m}_{row}\mathbf{m}_{col}^\top + (\mathbf{W}_{down} \odot \mathbf{m}_{row}\mathbf{1}^\top)(\mathbf{W}_{up} \odot \mathbf{1}\mathbf{m}_{col}^\top), \tag{1}$$

where $\mathbf{W}_{down} \in \mathbb{R}^{n \times r}$ and $\mathbf{W}_{up} \in \mathbb{R}^{r \times m}$ are adapter weights and r is the rank hyper-parameter. This is illustrated in Fig. 7

Similarly, we propose the Structured Pruning Low-rank PHM Adapter (SPLoPA) for block-based pruning with adaptation:

$$\mathbf{W}_t = (\mathbf{W}_s \odot [\mathbf{M}_b \otimes \mathbf{J}]) + \sum_{i=1}^{R} (\mathbf{M}_b \odot \mathbf{B}_i) \otimes \mathbf{p}_i \mathbf{q}_i^\top. \tag{2}$$

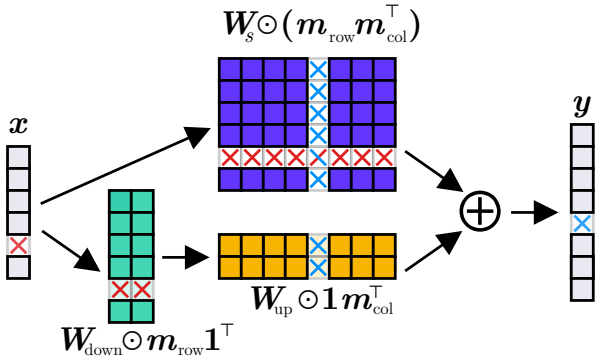


Figure 7: Structured Pruning Low-rank Adapter (SPLoRA). Pruning of in/out channels affects the adapter as well as source weights.

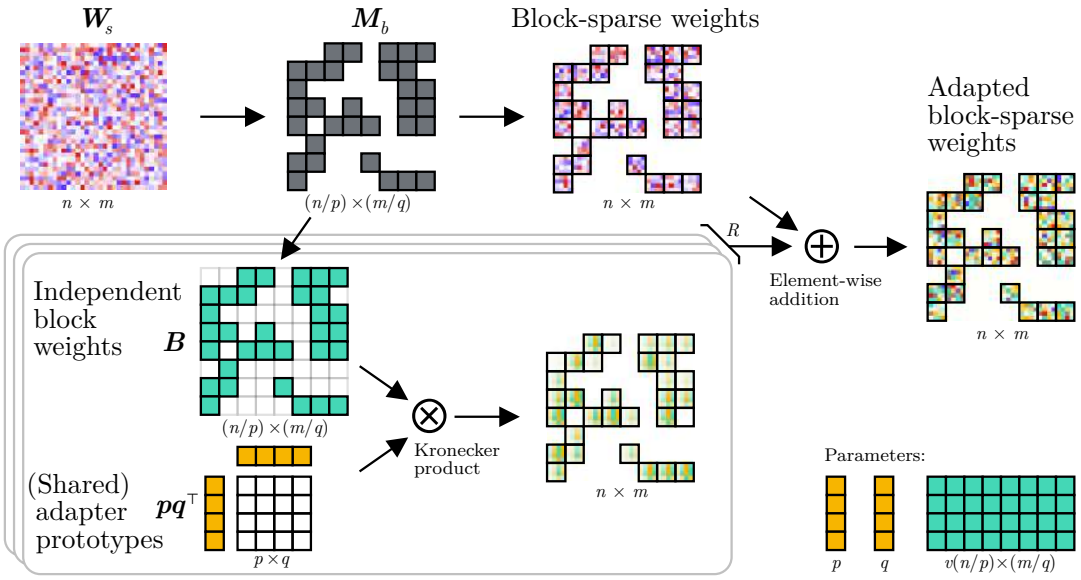


Figure 8: Structured Pruning Low-rank PHM Adapter (SPLoPA) visualized for $n = m = 32$, $p = q = 4$, and 50% density.

Here, we find inspiration in Parameterized Hypercomplex Multiplication (PHM) layers [135] and define the adaptation of each weight block with a weighted summation low-rank $p \times q$ block weights, which are shared across all blocks. For an $n \times m$ matrix, both this adaptation and the block masking with $\mathbf{M}_b \in \{0,1\}^{n/p \times m/q}$ are expressed with the Kronecker product ($\otimes$). In the above, we denote the low-rank weights as $\mathbf{p} \in \mathbb{R}^{p \times r}$ and $\mathbf{q} \in \mathbb{R}^{q \times r}$, the block position weights by $\mathbf{B} \in \mathbb{R}^{n/p \times m/q}$, and a matrix of ones as $\mathbf{J} \in \{1\}^{p \times q}$. Fig. 8 depicts the approach visually.

3.4.4 Performance evaluation

We evaluated our proposed methods, SPLoRA and SPLoPA, with a battery of channel- and block-based pruning methods on transfer-learning tasks for respectively image-classification (from ResNet-50 weights trained on ILSVRC 2012 [97] to CIFAR-10 [55], Oxford Flowers 102 [85], and Cats and Dogs [25]) and question-answering (BERT-base weights to the Stanford Question Answering Dataset 1.1 (SQuAD) [94]). We use fine-tuning with pruning ("fine-pruning") as a baseline in all cases. The channel-based pruning and adaptation results are

Table 9: Channel-based transfer-pruning from ResNet-50 pre-trained on ImageNet to Cats and Dogs, Oxford Flowers 102, and CIFAR-10. ΔParams and floating point operations (FLOPs) are shown for CIFAR-10. Mean ± standard deviation is shown for each metric. Changes specified relative to closest fine-pruning density with same pruning method. Best metric per pruning-method is highlighted with bold.

Pruning method	**Learning method**	**Density**	**ΔParams** (K)	**FLOPs** (M)	**CIFAR-10** Acc. (%)	**Oxford Flowers 102** Acc. (%)	**Cats & Dogs** Acc. (%)
Unpruned	Fine-tuning	100%	23,520.8±0.0	1,304.7±0.0	**97.10**±0.12	**92.20**±0.00	**99.30**±0.02
	LoRA-*r*32	100%	1,644.5±0.0 (↓ 14.3×)	1,304.7±0.0	95.32±0.13 (−1.8)	78.57±5.93 (−13.6)	98.60±0.43 (−0.5)
	LoRA-*r*8	100%	**466.3**±0.0 (↓ 50.4×)	1,304.7±0.0	95.35±0.10 (−1.8)	80.96±5.83 (−11.2)	98.84±0.52 (−0.46)
Weight [59]	Fine-pruning	30%	4,427.1±72.5	785.4±5.4	**96.38**±0.12	93.64±2.15	**98.64**±0.04
		10%	599.1±20.1	352.1±3.9	87.23±2.00	72.83±0.93	95.42±0.31
	SPLoRA-*r*32	30%	618.3±2.5 (↓ 7.2×)	778.6±2.9	95.59±0.22 (−0.8)	**94.57**±0.58 (+0.9)	98.43±0.13 (−0.2)
		10%	294.8±0.4 (↓ 2.0×)	**335.4**±7.3	93.04±0.37 (+5.8)	89.36±1.09 (+16.5)	96.25±0.99 (+0.8)
	SPLoRA-*r*8	30%	210.0±1.9 (↓ 21.1×)	773.4±2.1	94.91±0.24 (−1.5)	92.13±0.15 (−1.5)	98.40±0.12 (−0.2)
		10%	**128.9**±0.1 (↓ 4.6×)	338.9±7.0	91.24±0.51 (+4.0)	86.49±0.74 (+13.7)	96.07±0.69 (+0.6)
Gradient [110]	Fine-pruning	30%	3,719.8±59.2	571.9±6.5	**95.95**±0.09	**94.21**±0.74	**98.22**±0.00
		10%	615.7±4.3	244.7±3.3	91.83±1.17	73.06±0.70	95.84±0.20
	SPLoRA-*r*32	30%	601.0±0.4 (↓ 6.2×)	564.6±1.5	94.91±0.09 (−1.0)	93.58±0.43 (−0.6)	98.17±0.08 (−0.0)
		10%	293.1±0.4 (↓ 2.1×)	**244.0**±1.9	93.65±0.36 (+1.8)	91.35±0.47 (+18.3)	97.54±0.17 (+1.7)
	SPLoRA-*r*8	30%	205.2±0.2 (↓ 18.1×)	565.2±4.1	94.09±0.28 (−1.9)	91.60±0.30 (−2.6)	98.15±0.07 (−0.1)
		10%	**128.3**±0.0 (↓ 4.8×)	245.4±4.0	91.25±0.20 (−0.6)	87.46±0.71 (+14.4)	97.19±0.28 (+1.4)
Taylor [80]	Fine-pruning	30%	3,392.8±81.1	559.9±0.7	**95.71**±0.02	92.91±0.56	**98.22**±0.18
		10%	576.8±9.9	236.9±3.3	88.07±0.66	65.67±4.12	95.30±0.21
	SPLoRA-*r*32	30%	599.7±0.9 (↓ 5.7×)	555.5±6.7	94.88±0.21 (−0.8)	**93.41**±0.06 (+0.5)	97.84±0.48 (−0.4)
		10%	292.6±0.5 (↓ 2.0×)	**242.0**±1.5	93.27±0.12 (+5.2)	91.30±0.10 (+25.6)	97.21±0.10 (+1.9)
	SPLoRA-*r*8	30%	205.3±0.1 (↓ 16.5×)	566.2±10.9	93.98±0.24 (−1.7)	91.51±0.49 (−1.4)	97.90±0.13 (−0.3)
		10%	**128.4**±0.1 (↓ 4.5×)	243.2±9.8	91.22±0.32 (+3.2)	86.76±0.42 (+21.1)	96.83±0.30 (+1.5)
LRP [132]	Fine-pruning	30%	4,428.1±20.6	719.9±0.7	**96.54**±0.14	**95.37**±0.08	**98.65**±0.07
		10%	599.0±–	302.1±–	93.39±–	87.56±2.75	–
	SPLoRA-*r*32	30%	592.6±0.9 (↓ 7.5×)	585.5±6.7	94.85±0.13 (−1.7)	93.62±0.38 (−1.8)	98.09±0.11 (−0.6)
		10%	290.9±0.2 (↓ 2.1×)	**270.4**±6.4	93.47±0.36 (+0.1)	91.22±0.46 (+3.7)	97.01±0.27
	SPLoRA-*r*8	30%	203.3±0.5 (↓ 21.8×)	591.1±12.4	93.53±0.18 (−3.0)	91.26±0.19 (−4.1)	97.80±0.20 (−0.8)
		10%	**128.0**±0.1 (↓ 4.7×)	281.6±1.7	90.94±0.39 (−2.5)	85.69±1.39 (−1.9)	96.88±0.52

presented in Table 9 and the block-based results are found in 10.

For channel-based pruning, SPLoRA achieved competitive accuracies to fine-pruning. While fine-pruning performed better at higher model density (on average 0.6% and 1.6% higher than SPLoRA ranks 32 and 16 at 30% density), SPLoRA required substantially fewer learned parameters (on average 6.2× and 17.0×). At lower-model density (i.e., higher pruning rates) SPLoRA retained accuracy more successfully and both outperformed fine-pruning (achieving 6.9% and 4.7% higher average accuracy than fine-pruning for ranks 32 and 16), and reduced the learned parameter count (by 2.0× and 4.2× on average). For block-based pruning, SPLoPA was able to learn adaptations with significantly fewer parameters than fine-pruning (between 2.0× and 11.6× fewer) while achieving a slightly lower F1-score (on average 0.8% and 2.5% for $R = 128$ and $R = 64$).

Structured Pruning Adapters are described in more detail in the associated paper, which can be found in Appendix 8.5:

- [36] L. Hedegaard, A. Alok, J. Jose, and A. Iosifidis, "*Structured Pruning Adapters*", arXiv:2211.10155, 2022.

Table 10: Block-based transfer-pruning of BERT-base to the SQuAD dataset under block-based, L_0 Regularization pruning [71], Movement Pruning [99], and Soft Movement Pruning (SMvP) [99]. Changes specified relative to closest fine-pruning density with same pruning method. Best metric per pruning-method is highlighted with bold.

Pruning method	**Learning method**	**Density**	**ΔParams** (M)	**SQuAD F1** (%)
L_0 Reg. [71]	Fine-pruning	29.6%	25.21	**81.49**
		10.1%	9.28	74.92
	SPLoPA-$^{R128}_{r2}$	29.5%	4.30 (↓ 5.9×)	80.75 (−0.7%)
		10.7%	2.30 (↓ 4.0×)	76.22 (+1.3%)
	SPLoPA-$^{R64}_{r2}$	30.3%	**2.19** (↓ 11.5×)	80.56 (−0.9%)
MvP [99]	Fine-pruning	30.1%	25.56	**82.97**
		10.1%	8.57	73.77
	SPLoPA-$^{R128}_{r2}$	31.7%	6.49 (↓ 3.9×)	79.98 (−3.0%)
		14.1%	4.36 (↓ 2.0×)	74.29 (+0.5%)
	SPLoPA-$^{R64}_{r2}$	10.1%	**2.20** (↓ 3.9×)	71.00 (−2.8%)
SMvP [99]	Fine-pruning	28.8%	24.48	**84.18**
		6.8%	5.84	75.61
	SPLoPA-$^{R128}_{r2}$	26.8%	4.02 (↓ 6.1×)	82.90 (−1.3%)
		6.0%	1.80 (↓ 3.2×)	73.79 (−1.8%)
	SPLoPA-$^{R64}_{r2}$	28.6%	2.10 (↓ 11.6×)	82.51 (−1.7%)
		6.9%	**0.95** (↓ 6.2×)	70.75 (−4.9%)

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

4.1 Facial Expression Recognition with Learning Diversified Feature Representations

4.1.1 Introduction and objectives

Facial expression is a fundamental natural signal for human beings to communicate. Facial Expression Recognition (FER) has received a great research interest in recent years due to its applications in different artificial intelligence area such as Human Computer Interaction (HCI), and healthcare. FER methods aim to solve a visual perception problem by learning feature representations from facial images/videos to be classified as an emotional category, i.e. happiness, sadness, fear, anger, surprise, disgust, neutral, and contempt. Deep Convolutional Neural Networks (CNNs) have achieved remarkable progress on laboratory-controlled datasets, like CK+ [72], in which the facial images are in fixed frontal pose without any occlusion. However, FER methods still have challenges for recognizing emotions on in-the-wild datasets, like AffectNet [81], in which illumination, occlusion and pose variations make considerable

change in facial appearance. Many recent methods have been proposed to address this issue by extracting multi-scale facial features from global and local perspectives and employing region-based attention mechanisms [140, 122], and employing ensemble learning to independently train de-correlated models to achieve robust and accurate classification. Increasing the diversity of features learned by different network layers/neurons has been recognized as an effective way to improve model generalization [125]. We proposed a mechanism for learning diversified facial feature representations by encouraging the learner(s) to extract diverse spatial and channel-wise features and accordingly achieve better performance in real-world scenarios.

4.1.2 Summary of state of the art

MA-Net [140] is a state-of-the-art method which comprises of a backbone for preliminary feature extraction and a two-branch network with global multi-scale and local attention modules receiving the extracted features by the backbone as input and perform high-level feature extraction. The first branch applies several multi-scale convolutions on the input feature maps and the second branch divides the input feature maps into four local spatial regions without overlap, and applies several parallel local attention networks on them to extract key regional features. This model with 50.54 M parameters needs to be trained on a large dataset for another perception task, like MS-Celeb-1M [31], and then finetuned on in-the-wild facial datasets like AffectNet. ESR [108] is an efficient ensemble-based method for facial expression recognition and emotion estimation which reduces the residual generalization error on in-the-wild datasets. ESR consists of two building blocks: 1) the base network which is composed of a stack of convolutional layers and is responsible for extracting low/middle-level features, 2) the ensemble network composed of several network branches which are supposed to learn distinctive features. All branches in this ensemble module receive the same feature maps extracted by the base network as input and perform the classification independently.

4.1.3 Description of work performed so far

We proposed to increase spatial and channel-wise feature diversity in CNN architectures and ensemble-based models for facial expression recognition. The diversity between different feature maps obtained by different learners or different layers of a CNN model can be obtained in channel and spatial dimensions as illustrated in Figure 9 by first applying pooling on spatial and channel dimensions and then computing the average similarity between every two pooled feature maps l, k using radial basis function as follows:

$$S_{lk} = \frac{1}{N}\sum_{i=1}^{N} exp(-\gamma \|\phi_l(\mathbf{x}_i) - \phi_k(\mathbf{x}_i)\|^2), \tag{3}$$

where N denotes the number of samples, γ is a hyperparameter, $\phi_l(\cdot)$ and $\phi_k(\cdot)$ denote the pooled feature maps two learners/layers. Considering the pairwise similarities between feature maps, the model diversity is obtained by computing the determinant of the matrix $\mathbf{S}$ indicating pairwise similarities of learners as S_{lk}, i.e.,:

$$\mathscr{D} = det(\mathbf{S}). \tag{4}$$

The model can be optimized in an end-to-end manner by minimizing the combined loss function comprising of classification loss and diversity:

$$Loss = \mathscr{L} - (\mathscr{D}_{ch} + \mathscr{D}_{sp}), \tag{5}$$

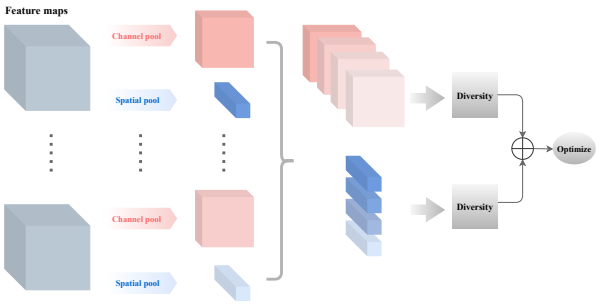


Figure 9: Illustration of diversity mechanism over the channel and spatial dimensions of feature maps.

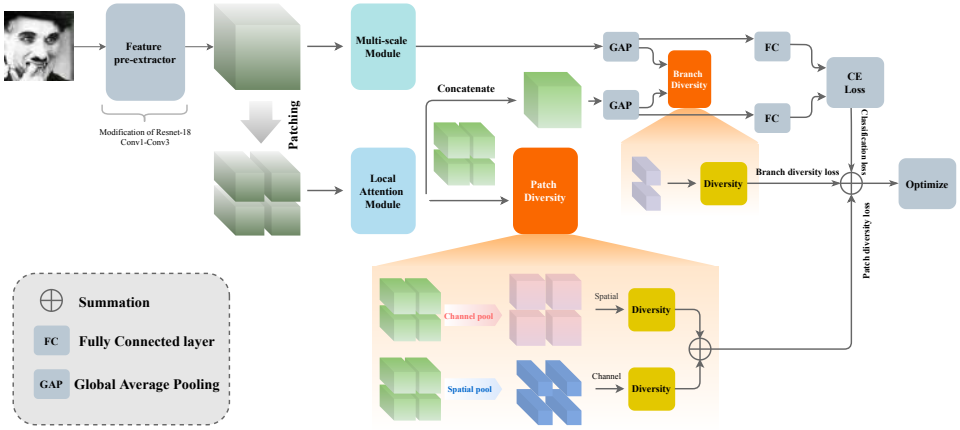


Figure 10: Illustration of the modified MA-Net structure by adding patch diversity and branch diversity blocks to diversify local region-based features in each feature map and also increase diversity of the extracted global and local features before passing them to the classification layers.

where $\mathscr{L}$ denotes the cross-entropy classification loss, and $\mathscr{D}_{ch}$, $\mathscr{D}_{sp}$ denote the feature diversity computed through channel and spatial dimensions, respectively.

We employed our proposed mechanism in two of the state-of-the-art methods, MA-Net and ESR, to increase diversity between regional feature maps and between the features extracted by ensemble branches, respectively. The modified structure of MA-Net and ESR are illustrated in Figure 10, 11, respectively. For more details about these new structures and their training mechanism, we refer the reader to the preprint appended in Appendix 8.6:

- [41] N. Heidari and A. Iosifidis, "*Learning Diversified Feature Representations for Facial Expression Recognition in the Wild*", arXiv:2210.09381, 2022.

4.1.4 Performance evaluation

We evaluated our proposed method by conducting experiments on three widely used in-the-wild datasets, AffectNet [81], FER+ [7], RAF-DB [63], and compared the classification accuracy of the modified version of MA-Net and ESR methods, indicated as ESR*, MA-Net*, with state-of-the-arts in Tables 11, 12, 13, respectively. ESR-9 and ESR-15 indicate ESR with 9 and 15 ensemble branches. Experimental results show that diversifying features extracted by different

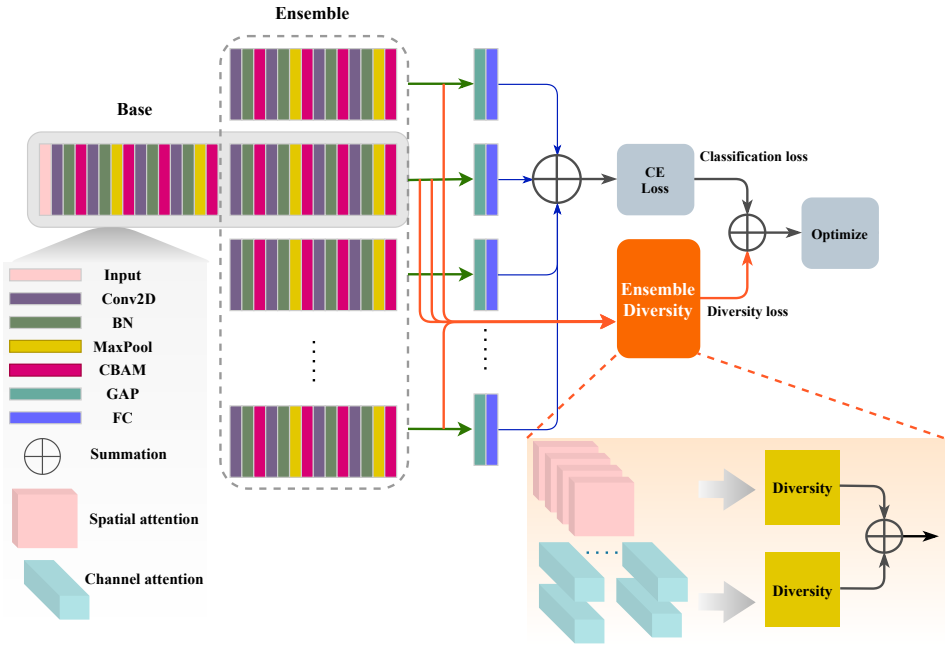


Figure 11: Illustration of the modified ESR structure with ensemble diversity block which diversifies the channel and spatial attention maps of the ensemble branches.

Table 11: Comparisons of the classification accuracy of the state-of-the-arts with our proposed version of ESR and MA-Net methods on AffectNet dataset with 8 classes. $*$ indicates our proposed version of the method.

Methods	Pretrained	Acc.(%)
MobileNet [42]	-	56.00
VGGNet [42]	-	58.00
AlexNet-WL [81]	-	58.00
RAN [122]	MS-Celeb-1M	59.50
SCN [121]	MS-Celeb-1M	60.23
ESR-9 [108]	AffectNet	59.30
ESR-9*	AffectNet	59.30
ESR-15*	AffectNet	**60.00**
MA-Net [140]	MS-Celeb-1M	**60.29**
MA-Net*	MS-Celeb-1M	60.02

ensemble learners can enhance the overall ensemble classification performance while increasing the model capacity to include more learners for feature extraction. Furthermore, diversifying local regional features extracted by a CNN learner improves the model performance in exploiting local features and classifying facial images.

Table 12: Comparisons of the classification accuracy of the state-of-the-arts with our proposed version of ESR and MA-Net methods on FER+ dataset with 8 classes. $*$ indicates our proposed version of the method.

Methods	Pretrained	Acc.(%)
TFE-JL [61]	-	84.30
PLD [7]	-	85.10
SHCNN [76]	-	86.54
SeNet50 [2]	VGG-Face2 [12]	88.80
RAN [122]	MS-Celeb-1M	88.55
	VGG-Face [87]	89.16
SCN [121]	MS-Celeb-1M	88.01
ESR-9 [108]	AffectNet	87.17
ESR-9*	AffectNet	**89.15**
ESR-15*	AffectNet	**89.34**
MAN-Net [140]	MS-Celeb-1M	-
MAN-Net*	MS-Celeb-1M	88.34

Table 13: Comparisons of the classification accuracy of the state-of-the-arts with our proposed version of ESR and MA-Net methods on RAF-DB dataset with 7 classes. $*$ indicates our proposed version of the method.

Methods	Pretrained	Acc.(%)
DLP-CNN [63]	-	84.22
IPA2LT [134]	AffectNet	86.77
gACNN [65]	AffectNet	85.07
LDL-ALSG [14]	AffectNet	85.53
RAN [122]	MS-Celeb-1M	86.90
SCN [121]	MS-Celeb-1M	87.03
ESR-9 [108]	AffectNet	-
ESR-9*	AffectNet	82.95
ESR-15*	AffectNet	83.00
MA-Net [140]	MS-Celeb-1M	88.40
MA-Net*	MS-Celeb-1M	**89.99**

5 Deep speech and biosignals analysis and recognition

5.1 Self-Attention Neural Bag-of-Features

5.1.1 Prior work and work performed so far

Detection of abnormal biosignals is important in many healthcare applications, with large focus being made on detection of heart abnormalities from ECG and PCG signals. During first years of OpenDR, a methodology to tackle this task by means of Neural Bag-of-Features has been proposed, introducing an attention mechanism referred to as 2DA [115]. The work extending this approach to a more generic self-attention based formulation has started in year 2, with

main approach mentioned in D3.2 along with some preliminary results, and has continued in year 3. To this end, we have proposed a self-attention based formulation of Neural Bag-of-Features framework that allows to learn attention over codebook, timestamps, and importantly joint attention maps over two dimensions. Additionally, the tool has been integrated in the toolkit and a corresponding publication can be found in [19] or Appendix 8.7.

Overall, we develop an attention mechanism for time-series classification and reformulate the existing Neural Bag-of-Features attention learning methodology by quantifying the relevance of feature/temporal dimensions through latent spaces based on self-attention rather than learning them directly. In addition, we propose a joint feature-temporal attention mechanism that learns a joint 2D attention mask highlighting relevant information without treating feature and temporal representations independently.

The full paper with more details can be found in Appendix 8.7:

- [19] K. Chumachenko, A. Iosifidis, and M. Gabbouj. "*Self-attention neural bag-of-features*", IEEE International Workshop on Machine Learning for Signal Processing, 2022.

5.1.2 Performance evaluation

For performance evaluation on biosignal datasets we consider two tasks: detection of atrial fibrilation from ECG (specifically, the task is formulated as a classification problem with 4 classes: normal sinus rhythm, atrial fibrillation, alternative rhythm, and noise) [20], and detection of abnormal phonocardiograms based on Physionet Computing in Cardiology Challenge dataset (PCG) [21]. In the latter scenario, we also consider the task of quality evaluation of phonocardiograms (PCG-2). We report the average results over 5 and 3 folds, respectively.

All the experiments are conducted with the logistic formulation of Neural Bag of Features [90] that uses hyperbolic kernel as a quantization layer and we use 256 codewords. We denote by 2DA-CA and 2DA-TA the conventional 2DA attention in its codebook and temporal formulations, respectively, and by 2DA-CTSA$_d$, 2DA-TSA$_d$, and 2DA-CSA$_d$ - the proposed variants of codebook-temporal self-attention, temporal self-attention, and codebook self-attention with the dimensionality of the latent space denoted by d. Note that d is a hyperparameter which can be tuned, but we instead report the results across multiple values.

The results of the experimental evaluation are shown in Table 14. The best result is highlighted in bold, and the result is underlined if it outperforms the corresponding 2DA formulation. As can be seen, in all three cases the best result is achieved by one of the proposed variants. In PCG dataset, codeword-temporal variant in high dimensions outperforms both codeword and temporal 2DA, and codeword self-attention significantly outperforms the codeword 2-DA. At the same time, in temporal representations the proposed approach outperform the 2DA approach in quality evaluation task on PCG dataset. Similar results are observed in AF dataset, where proposed self-attention approaches outperform codeword and temporal 2DA. We additionally perform experiments with multi-head formulation, outlined in Table 15, where the outcomes are similar. Additionally, to evaluate the generalization of the propsoed approach, we evaluate it in the task of acoustic scene classification on TUT-UAS2018 dataset [75]. The results are outlined in Table 16. More experimental evaluation can be found in Appendix 8.7.

Table 14: F1 scores on biosignal datasets

Attention models	**PCG**	**PCG-2**	**AF**
2DA-CA	86.93 + 0.35	73.44 + 1.23	77.33 + 2.44
2DA-TA	87.45 + 0.74	73.39 + 1.16	76.71 + 2.06
2DA-CTSA$_{512}$	87.75 + 0.78	73.75 + 1.81	77.56 + 1.75
2DA-CTSA$_{256}$	87.46 + 1.30	73.50 + 0.77	77.55 + 2.42
2DA-CTSA$_{128}$	87.74 + 0.65	73.62 + 1.80	76.96 + 1.24
2DA-CTSA$_{64}$	87.07 + 1.02	73.38 + 1.36	77.87 + 1.71
2DA-TSA$_{512}$	88.06 + 0.61	73.46 + 1.45	76.86 + 2.34
2DA-TSA$_{256}$	87.26 + 0.52	74.14 + 1.77	76.87 + 1.86
2DA-TSA$_{128}$	87.08 + 1.00	**74.47 + 1.03**	77.27 + 2.13
2DA-TSA$_{64}$	87.77 + 0.61	73.31 + 1.58	76.99 + 1.74
2DA-CSA$_{512}$	88.36 + 0.22	73.35 + 1.15	77.28 + 1.60
2DA-CSA$_{256}$	**88.38 + 0.55**	73.95 + 0.90	77.70 + 1.90
2DA-CSA$_{128}$	87.19 + 0.98	73.02 + 2.14	**78.70 + 1.50**
2DA-CSA$_{64}$	87.71 + 0.44	72.79 + 0.67	77.96 + 1.88

Table 15: F1 scores on biosignal datasets with 2 heads

Models	**PCG-1**	**PCG-2**	**AF**
2DA-CA	86.93 + 0.35	73.44 + 1.23	77.33 + 2.44
2DA-TA	87.45 + 0.74	73.39 + 1.16	76.71 + 2.06
2DA-CTSA$_{512}$	87.80 + 0.73	74.57 + 1.14	76.43 + 2.78
2DA-CTSA$_{256}$	87.84 + 0.12	73.14 + 0.70	76.60 + 1.70
2DA-CTSA$_{128}$	87.24 + 0.74	73.77 + 1.02	77.06 + 1.39
2DA-CTSA$_{64}$	88.04 + 0.52	73.73 + 1.16	76.98 + 1.92
2DA-CTSA$_{32}$	87.63 + 0.83	73.69 + 0.98	77.79 + 2.00
2DA-TSA$_{512}$	86.98 + 0.76	73.66 + 0.78	76.77 + 2.26
2DA-TSA$_{256}$	87.61 + 0.70	73.32 + 1.15	76.31 + 1.59
2DA-TSA$_{128}$	87.69 + 1.11	72.64 + 2.19	77.23 + 1.48
2DA-TSA$_{64}$	87.09 + 0.60	73.55 + 0.80	77.21 + 1.95
2DA-TSA$_{32}$	87.03 + 0.44	74.38 + 1.81	77.41 + 2.18
2DA-CSA$_{512}$	87.47 + 0.78	72.97 + 0.72	**77.88 + 1.43**
2DA-CSA$_{256}$	**88.31 + 0.60**	**74.94 + 1.77**	77.70 + 1.69
2DA-CSA$_{128}$	87.33 + 0.68	74.46 + 0.62	77.47 + 0.96
2DA-CSA$_{64}$	87.49 + 0.84	73.41 + 1.13	76.91 + 1.11
2DA-CSA$_{32}$	87.72 + 0.57	73.08 + 0.60	76.69 + 1.22

Table 16: Accuracies on TUT-UAS2018 dataset with 2 and 4 heads

Attention models	TUT-UAS, h=2	TUT-UAS, h=4
2DA-CA	56.15 + 0.21	56.15 + 0.21
2DA-TA	56.09 + 0.51	56.09 + 0.51
2DA-CTSA$_{512}$	57.23 + 1.00	57.04 + 0.80
2DA-CTSA$_{256}$	56.15 + 1.17	**58.52 + 0.70**
2DA-CTSA$_{128}$	**57.48 + 0.64**	58.02 + 0.45
2DA-CTSA$_{64}$	54.91 + 1.22	58.07 + 1.92
2DA-CTSA$_{32}$	57.21 + 0.29	56.31 + 0.74
2DA-TSA$_{512}$	56.61 + 0.91	55.82 + 1.18
2DA-TSA$_{256}$	55.80 + 0.98	56.07 + 0.48
2DA-TSA$_{128}$	55.84 + 0.51	57.62 + 1.71
2DA-TSA$_{64}$	57.83 + 0.16	56.71 + 0.81
2DA-TSA$_{32}$	56.31 + 1.10	57.83 + 0.23
2DA-CSA$_{512}$	57.40 + 0.23	57.13 + 1.20
2DA-CSA$_{256}$	56.37 + 1.24	55.45 + 0.71
2DA-CSA$_{128}$	55.62 + 1.18	56.91 + 1.31
2DA-CSA$_{64}$	56.99 + 1.21	56.54 + 1.45
2DA-CSA$_{32}$	56.04 + 1.06	56.26 + 1.53

6 Multi-modal human centric perception and cognition

6.1 Self-attention fusion for audiovisual emotion recognition

6.1.1 Introduction and objectives

An important problem in the field of machine learning is the recognition of human emotional states, which enables a better comprehension of social cues in a variety of applications from robotics to human-computer interaction. To this end, a number of strategies and emotion models have been put forth. These range from the problem of identifying discrete emotional states, such as ”happy,” ”angry,” or ”sad,” to the estimation of emotional properties, such as arousal and valence on a continuous scale. A variety of multi-modal techniques are emerging within this field, attempting to improve emotion recognition performance by utilizing multiple data sources simultaneously. These methods are ranging from simple decision-level fusion approaches to more sophisticated joint feature learning approaches. Nevertheless, the majority of multi-modal approaches created to date assume an idealistic environment with full presence of data of all modalities at all times, which generally does not hold for real-world unconstrained scenarios, where data of one modality can be absent, incomplete or noisy at times. Additionally, usage of pre-extracted features that are subsequently fused with a learned model, rather than development of end-to-end trainable models, still remains popular in the task of multi-modal emotion recognition. This limits the applicability of such methods in real-world scenarios, as necessary feature extraction is often challenging in unconstrained settings and introduces another point of uncertainty to the overall processing pipeline.

6.1.2 Summary of state of the art

In the field of multi-modal machine learning, three classes of multi-modal fusion approaches are typically recognized: early fusion, where the input data of various modalities are simply

combined via concatenation, addition, or any other operation and further processed together; late fusion, where modalities are treated independently and their features or softmax classification scores are only combined in the very last layers; and intermediate feature fusion, where different modalities are treated independently but their features or scores are combined throughout the entire process. Notable examples of multi-modal methods within the field of audiovisual emotion recognition include [116, 46, 54]. It can be noted that these models focus on building multi-modal fusion methods rather than end-to-end emotion recognition systems, and often employ features that require separate extraction, such as facial action units estimated from images or text transcribed from speech. Additionally, it should be noted that all three of the aforementioned approaches perform fusion in the beginning or middle of the pipeline, forcing the learning of joint representations. While joint feature learning has advantages, such fusion can also become a curse if the learned fused representations are overly dependent on one another and one of the modalities is noisy, lacking, or nonexistent at the time of inference. Indeed, the conventional approach has been to only assess the models' performance in the ideal circumstance of both modalities being present and complete at all times, despite the fact that real-world applications don't often correspond to such scenarios.

6.1.3 Description of work performed so far

We propose a model for emotion recognition from audio (speech) and visual data that is trainable into end-to-end manner [18]. On a general level, the model consists of two branches, that is, the audio and the vision branch, and three fusion approaches are proposed. We additionally propose a training approach aimed at improving robustness of the model to missing or noisy data. Here, we provide a brief description of each of the components of the architecture. More detailed description of the approach can be found in the following paper, attached in Appendix 8.8:

- [18]. K. Chumachenko, A. Iosifidis, and M. Gabbouj. "*Self-attention fusion for audiovisual emotion recognition with incomplete data*", arXiv preprint arXiv:2201.11095, 2022.

The vision branch consists of two branches, with the first one being the visual feature extraction from individual video frames, followed by learning of joint representation for the whole video sequence. To achieve an end-to-end trainable model capable of learning from raw video, we employ feature extraction as part of our pipeline and optimize it jointly with the multimodal fusion module. We choose one of the recently proposed facial expression recognition architectures, namely, EfficientFace [141] and incorporate it for feature extraction from individual frames prior to introducing them to subsequent 1D convolutional blocks. Specifically, the 1D convolutional blocks are added after average-pooled output of the last convolutional block of EfficientFace. Note that visual feature extraction can be decoupled from the model and any other features can be used instead as input to the second part of the vision branch.

Similarly to the vision branch, the audio branch operates on a feature representation, whether pre-computed or optimized jointly, and applies four blocks of 1D convolutional layers. For audio, we primarily use mel-frequency cepstral coefficients as features.

Further, one of the three fusion approaches are utilized. In the first approach, referred to as late transformer fusion, features learnt from two branches are fused with a transformer block. Specifically, we employ two transformers at the outputs of each branch, where fusion of one modality is performed into the other one via self-attention. This is achieved by learning queries

from data corresponding to one modality and attending them to keys and values learnt from the other modality. The fusion is followed by a standard transformer block processing and the outputs of these transformer blocks are further concatenated and passed to the final prediction layer. The second approach, referred to as intermediate transformer fusion is similar to the first one, while fusion is performed with a transformer block in each branch at an intermediate feature level, that is, after the first stage of feature extraction. The fused feature representation is added to the corresponding branch. Since data from complementary modality is introduced already at the earlier stage of the architecture, this allows to learn the features that are jointly meaningful for the task at hand between modalities during later convolutional layers.

Further, we propose a third fusion approach that is based merely on dot-product similarity that constitutes the attention in the transformer block. Formally, this is defined as follows. Given the two feature representations of different modalities Φ_a and Φ_b, we compute queries and keys with learnt weights, similarly to conventional attention. The scaled dot-product similarity is subsequently calculated as $A = softmax\left(\frac{\Phi_a \mathbf{W}_q \mathbf{W}_k^T \Phi_v^T}{\sqrt{d}}\right)$. Softmax activation promotes competition in the attention matrix, hence highlighting more important attributes/timestamps of each modality, and as a result providing the importance score of each key with respect to each query, i.e., each representation of modality a with respect to modality b. This allows to calculate the relative importance of each attribute of modality a by aggregating the scores corresponding to all the attributes of modality b for each attribute of modality a. As a result, we obtain an attention vector that can be used to highlight more relevant attributes of the modality a. Considering the dot-product attention between features of audio and vision modalities shown, attention vector of vision modality is given by $\mathbf{v}_v = \sum_{i=N_v} A[:,i]$.

In order to improve the robustness of the model in unconstrained environments, we further propose a simple approach to account for the potential cases of missing or noisy data. Namely, we propose the modality dropout, which randomly masks out or attenuates data of one of the modalities during training. During training, data of one modality in each sample is randomly replaced with zeros, while the representation of the other modality for a given sample is kept intact. This approach imitates missing data and can also act as a regularizer similarly to Dropout layer utilized in neural networks. Alternatively, for each pair of data samples we generate a random scaling factor α in the range [0,1] and scale one of the modalities by α, while the other with $1-\alpha$. The goal of this approach is to attenuate signals from different modalities at different training steps, and hence prevent the model from learning from strictly one modality. The third variant is aimed at the problem of noisy data, where the input signal of one modality is corrupted. Here, the data is randomly sampled from a normal distribution with zero mean and unit variance in one of the modalities for each pair.

6.1.4 Performance evaluation

For evaluating the performance of the proposed method, experiments are performed on two speech and vision based emotion recognition datasets, namely, RAVDESS [70] and CMU-MOSEI [133]. We compare our proposed approach with the methods proposed in [116, 46, 54]. RAVDESS dataset consists of video recordings of 24 people speaking with different emotions and poses a task of classification of emotional states into 7 classes: calm, happy, sad, angry, fearful, surprise, and disgust. 60 video sequences were recorded for each actor, and we crop or zero-pad them to 3.6 seconds, which is the average sequence length. CMU-MOSEI consists of 23,454 movie review video clips taken from YouTube and labeled by human annotators with a sentiment score in the range [-3..3].

We perform comparison in the standard setting as well as with one of the modalities missing or corrupted with noise. More details on experimental setup can be found in [18]. Table 17 shows the results of the proposed approaches on the RAVDESS and MOSEI datasets. Here, 'LT1' and 'LT4' denote late transformer fusion with one and four heads, respectively, and similarly 'IT' denotes intermediate transformer fusion, 'IA' denotes intermediate attention fusion, 'TAV' and 'TVA' refers to the fusion approaches described in [46], and 'MULT' refers to [116]. We report categorical accuracy on RAVDESS dataset, and binary accuracy (positive vs negative sentiment) on MOSEI dataset, as well as Mean Average Error between the true and predicted sentiment scores.

As can be seen, in the setting without any type of dropout, late transformer fusion achieves the best result on RAVDESS dataset, while intermediate attention fusion achieves the best result on MOSEI dataset on both the accuracy and MAE metrics. Note that intermediate attention fusion is also the most lightweight fusion approach compared to any of the methods using full transformer blocks. Further, it can be seen that utilization of modality dropout improves the performance drastically under incomplete data of one modality. This is the case for most fusion methods, while intermediate attention fusion benefits from it the most. Besides, the performance under the presence of both modalities is improved as well, with the best result on RAVDESS achieved by intermediate attention fusion. This is also the best result on this dataset among all methods and dropout settings.

Table 17: Performance of different fusion methods on RAVDESS and MOSEI.

	RAVDESS. ACC				MOSEI. ACC				MOSEI. MAE			
	AV	A	V	M	AV	A	V	M	AV	A	V	M
LT1	**79.33**	19.83	36.41	**45.19**	63.89	48.70	62.85	58.48	0.806	0.840	1.063	0.903
LT4	76.42	**27.92**	30.00	44.78	66.56	62.63	53.16	60.78	0.806	0.839	0.831	0.825
IT1	76.41	21.16	18.33	38.63	**67.72**	37.14	62.87	55.91	**0.792**	0.843	0.809	0.815
IT4	78.50	20.33	17.33	38.72	64.91	62.60	62.85	63.45	0.817	0.840	0.832	0.830
IA1	76.00	18.58	22.83	39.13	64.94	62.08	62.86	63.29	0.802	**0.837**	0.806	0.815
IA4	77.41	20.66	29.83	42.63	**67.72**	**63.07**	**65.77**	**65.52**	0.794	**0.837**	**0.803**	**0.811**
TAV	77.75	24.25	13.33	38.44	64.94	62.08	62.86	62.18	0.814	0.841	1.093	0.916
TVA	76.00	15.16	**42.67**	44.61	66.48	37.15	56.96	53.53	0.809	0.852	0.838	0.833
MLT	74.16	22.33	35.42	43.97	62.90	62.85	64.44	63.40	0.804	0.838	0.804	0.815
MODALITY DROPOUT												
LT1	79.08	**59.16**	72.66	70.30	67.11	63.62	62.90	64.54	0.802	0.829	0.801	0.811
LT4	79.25	53.00	70.92	67.72	64.47	53.71	64.91	61.03	0.814	0.837	0.819	0.824
IT1	77.33	48.41	73.75	66.50	62.80	62.85	63.09	62.91	0.804	0.831	0.803	0.813
IT4	78.91	44.33	**74.92**	66.05	67.01	64.30	63.12	64.81	0.796	0.826	**0.797**	0.806
IA1	**81.58**	58.08	72.83	**70.83**	67.19	**64.52**	**64.91**	**65.54**	0.795	**0.816**	0.798	**0.803**
IA4	79.58	57.16	71.83	69.52	63.48	62.74	63.18	63.13	0.807	0.820	0.808	0.812
TAV	76.58	54.83	13.33	48.24	65.32	63.84	62.85	64.01	0.811	0.832	0.839	0.828
TVA	74.42	44.91	69.58	62.97	**67.61**	63.98	60.95	64.18	**0.793**	0.819	0.798	0.803
MLT	78.50	53.58	70.66	67.58	63.87	62.85	63.37	63.36	0.806	0.836	0.835	0.826
MODALITY DROPOUT with NOISE												
LT1	77.08	53.16	68.50	66.246	65.57	64.03	64.94	64.94	0.809	0.826	0.806	0.813
LT4	**80.33**	54.33	**73.00**	69.22	64.08	63.31	62.85	62.85	0.813	0.827	0.813	0.818
IT1	76.75	53.75	71.58	67.36	**68.16**	**65.98**	63.53	63.53	**0.799**	0.821	0.804	**0.808**
IT4	76.08	54.50	71.00	67.19	67.83	63.56	64.22	64.22	0.801	0.826	**0.802**	0.809
IA1	78.25	**58.25**	71.66	**69.38**	62.76	63.89	63.18	63.27	0.804	**0.819**	0.805	0.809
IA4	78.41	55.75	68.58	67.58	63.51	64.08	62.54	63.37	0.805	0.820	0.808	0.811
TAV	75.83	56.25	12.83	48.30	66.81	65.68	**65.60**	**66.03**	0.810	0.820	0.811	0.813
TVA	73.66	41.25	71.41	62.10	66.23	63.18	64.58	64.66	0.804	0.831	0.806	0.813
MLT	77.41	54.16	66.33	65.96	64.52	62.74	63.51	63.59	0.805	0.830	0.805	0.811

7 Conclusions

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 3.3), which reduce the per-prediction floating point operations by an order of magnitude. AU formally described the class of Continual Inference Networks for neural architectures which remove the computational redundancies to efficient frame-wise predictions, and made available a public implementation (Section 3.2). AU also proposed an efficient methodology for image-based facial expression recognition (Section 4.1) which has improved the state-of-the-art performance on in-the-wild benchmark datasets by learning diversified feature representations to improve the model generalization. Moreover AU proposed Structured Pruning Adapters (Section 3.4), a weight-based transfer learning method, which can learn additional tasks with an order of magnitude fewer parameters than pruning with fine-tuning.

TAU worked towards objective O1 by developing an end-to-end multimodal emotion recognition model described in Section 6.1 based on speech and face video that is robust towards incomplete data, which improved the state-of-the-art performance on two emotion recognition datasets in both standard and unconstrained inference scenarios. Within the scope of the same

objective TAU has also improved upon the capabilities of its Neural Bag-of-Features framework (Section 5.1), further boosting the effectiveness of biosignal (i.e.) analysis.

AUTH worked towards objective O1 by developing two variants of its novel non-maximum suppression method for person detection (Section 2.5). While the first variant was demonstrated to be more robust to visual data distribution-shifts, the second was able to utilize existing feature maps of DL-based object detectors in an efficient manner, for computing appearance-based RoI representations, achieving top results and shorter inference times. Moreover, attempting to tackle the situation that active face recognition is largely understudied in recent literature, Furthermore, AUTH worked towards O1a by developing active perception models for face recognition (Sections 2.1 and 2.2, Section 2.3). Finally, AUTH developed a high resolution pose estimation methodology (Section 2.4), working towards O1b.

References

[1] Face recognition techniques: A survey. *CoRR*, abs/1803.07288, 2018. Withdrawn.

[2] S. Albanie, A. Nagrani, A. Vedaldi, and A. Zisserman. Emotion recognition in speech using cross-modal transfer in the wild. In *ACM International Conference on Multimedia*, 2018.

[3] A. Andreopoulos and J. K. Tsotsos. 50 years of object recognition: Directions forward. *Computer vision and image understanding*, 117(8):827–891, 2013.

[4] 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.

[5] R. Bajcsy. Active perception and exploratory robotics. 1989.

[6] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos. Revisiting active perception. *Autonomous Robots*, 42(2):177–196, 2018.

[7] E. Barsoum, C. Zhang, C. C. Ferrer, and Z. Zhang. Training deep networks for facial expression recognition with crowd-sourced label distribution. In *International Conference on Multimodal Interaction*, 2016.

[8] L. Bartolomei, L. Teixeira, and M. Chli. Semantic-aware active perception for uavs using deep reinforcement learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3101–3108. IEEE, 2021.

[9] 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.

[10] I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The long-document transformer. *arXiv:2004.05150*, 2020.

[11] C. Caetano, F. Brémond, and W. R. Schwartz. Skeleton image representation for 3d action recognition based on tree structure and reference joints. In *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 16–23, 2019.

[12] 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.

[13] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields, 2016.

[14] S. Chen, J. Wang, Y. Chen, Z. Shi, X. Geng, and Y. Rui. Label distribution learning on auxiliary label space graphs for facial expression recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[15] 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.

[16] K. Cheng, Y. Zhang, X. He, J. Cheng, and H. Lu. Extremely lightweight skeleton-based action recognition with shiftgcn++. *IEEE Transactions on Image Processing*, 30:7333–7348, 2021.

[17] K. M. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Q. Davis, A. Mohiuddin, L. Kaiser, D. B. Belanger, L. J. Colwell, and A. Weller. Rethinking attention with performers. In *International Conference on Learning Representations (ICLR)*, 2021.

[18] K. Chumachenko, A. Iosifidis, and M. Gabbouj. Self-attention fusion for audiovisual emotion recognition with incomplete data. *arXiv preprint arXiv:2201.11095*, 2022.

[19] K. Chumachenko, A. Iosifidis, and M. Gabbouj. Self-attention neural bag-of-features. *IEEE International Workshop on Machine Learning for Signal Processing*, 2022.

[20] G. D. Clifford, C. Liu, B. Moody, H. L. Li-wei, I. Silva, Q. Li, A. Johnson, and R. G. Mark. Af classification from a short single lead ecg recording: The physionet/computing in cardiology challenge 2017. In *2017 Computing in Cardiology (CinC)*, pages 1–4. IEEE, 2017.

[21] G. D. Clifford, C. Liu, B. Moody, D. Springer, I. Silva, Q. Li, and R. G. Mark. Classification of normal/abnormal heart sound recordings: The physionet/computing in cardiology challenge 2016. In *2016 Computing in cardiology conference (CinC)*, pages 609–612. IEEE, 2016.

[22] R. De Geest, E. Gavves, A. Ghodrati, Z. Li, C. Snoek, and T. Tuytelaars. Online action detection. In *European Conference on Computer Vision (ECCV)*, pages 269–284, 2016.

[23] 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.

[24] 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.

[25] J. Elson, J. J. Douceur, J. Howell, and J. Saul. Asirra: A captcha that exploits interest-aligned manual image categorization. In *Proceedings of 14th ACM Conference on Computer and Communications Security (CCS)*. Association for Computing Machinery, Inc., 2007.

[26] H. Eun, J. Moon, J. Park, C. Jung, and C. Kim. Learning to discriminate information for online action detection. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 806–815, 2020.

[27] H. Eun, J. Moon, J. Park, C. Jung, and C. Kim. Temporal filtering networks for online action detection. *Pattern Recognition*, 111:107695, 2021.

[28] B. Fernando, E. Gavves, M. José Oramas, A. Ghodrati, and T. Tuytelaars. Modeling video evolution for action recognition. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5378–5387, 2015.

[29] J. Gao, Z. Yang, and R. Nevatia. RED: reinforced encoder-decoder networks for action anticipation. In *British Machine Vision Conference (BMVC)*, 2017.

[30] 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.

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

[32] X. Hao, J. Li, Y. Guo, T. Jiang, and M. Yu. Hypergraph neural network for skeleton-based action recognition. *IEEE Transactions on Image Processing*, 30:2263–2275, 2021.

[33] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*, 2022.

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

[35] L. He, H. Li, Q. Zhang, and Z. Sun. Dynamic feature learning for partial face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7054–7063, 2018.

[36] L. Hedegaard, A. Alok, J. Jose, and A. Iosifidis. Structured pruning adapters. *preprint, arXiv:2211.10155*, 2022.

[37] L. Hedegaard, N. Heidari, and A. Iosifidis. Online skeleton-based action recognition with continual spatio-temporal graph convolutional networks. *preprint, arXiv:2203.11009*, 2022.

[38] L. Hedegaard and A. Iosifidis. Continual 3d convolutional neural networks for real-time processing of videos. In *European Conference on Computer Vision (ECCV)*, 2022.

[39] L. Hedegaard and A. Iosifidis. Continual transformers: Redundancy-free attention for online inference. *preprint, arXiv:2201.06268*, 2022.

[40] 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.

[41] N. Heidari and A. Iosifidis. Learning diversified feature representations for facial expression recognition in the wild. *preprint, arXiv:2210.09381*, 2022.

[42] C. Hewitt and H. Gunes. Cnn-based facial affect analysis on mobile devices. *arXiv preprint arXiv:1807.08775*, 2018.

[43] S. Hörmann, Z. Zhang, M. Knoche, T. Teepe, and G. Rigoll. Attention-based partial face recognition. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 2978–2982. IEEE, 2021.

[44] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799, 2019.

[45] E. J. Hu, yelong shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[46] J. Huang, J. Tao, B. Liu, Z. Lian, and M. Niu. Multimodal transformer fusion for continuous emotion recognition. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3507–3511. IEEE, 2020.

[47] H. Idrees, A. R. Zamir, Y.-G. Jiang, A. Gorban, I. Laptev, R. Sukthankar, and M. Shah. The thumos challenge on action recognition for videos "in the wild". *Computer Vision and Image Understanding*, 155:1–23, 2017.

[48] U. Iqbal and J. Gall. Multi-person pose estimation with local joint-to-person associations.

[49] 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.

[50] T. S. Kim and A. Reiter. Interpretable 3D human action analysis with temporal convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1623–1631, 2017.

[51] Y. H. Kim, S. Nam, and S. J. Kim. Temporally smooth online action detection using cycle-consistent future anticipation. *Pattern Recognition*, 116:107954, 2021.

[52] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*, 2017.

[53] N. Kitaev, L. Kaiser, and A. Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations (ICLR)*, 2020.

[54] D. Krishna and A. Patil. Multimodal emotion recognition using cross-modal attention and 1d convolutional neural networks. In *Interspeech*, pages 4243–4247, 2020.

[55] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[56] F. Lagunas, E. Charlaix, V. Sanh, and A. Rush. Block pruning for faster transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10619–10629, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics.

[57] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[58] B. Li, X. Li, Z. Zhang, and F. Wu. Spatio-temporal graph routing for skeleton-based action recognition. In *AAAI*, 2019.

[59] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations (ICLR)*, 2017.

[60] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, and Q. Tian. Actional-structural graph convolutional networks for skeleton-based action recognition. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3590–3598, 2019.

[61] M. Li, H. Xu, X. Huang, Z. Song, X. Liu, and X. Li. Facial expression recognition with identity and emotion joint learning. *IEEE Transactions on Affective Computing*, 12(2):544–550, 2018.

[62] M. Li, Z. Zhou, J. Li, and X. Liu. Bottom-up pose estimation of multiple person with bounding box constraint. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 115–120, 2018.

[63] S. Li, W. Deng, and J. Du. Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[64] X. L. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, 2021.

[65] Y. Li, J. Zeng, S. Shan, and X. Chen. Occlusion aware facial expression recognition using cnn with attention mechanism. *IEEE Transactions on Image Processing*, 28(5):2439–2450, 2018.

[66] J. Liu, A. Shahroudy, M. Perez, G. Wang, L.-Y. Duan, and A. C. Kot. Ntu rgb+d 120: A large-scale benchmark for 3d human activity understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.

[67] J. Liu, A. Shahroudy, M. Perez, G. Wang, L.-Y. Duan, and A. C. Kot. Ntu rgb+d 120: A large-scale benchmark for 3d human activity understanding. *IEEE transactions on pattern analysis and machine intelligence*, 42(10):2684–2701, 2019.

[68] J. Liu, A. Shahroudy, D. Xu, and G. Wang. Spatio-temporal LSTM with trust gates for 3D human action recognition. In *European Conference on Computer Vision*, pages 816–833. Springer, 2016.

[69] Z. Liu, H. Zhang, Z. Chen, Z. Wang, and W. Ouyang. Disentangling and unifying graph convolutions for skeleton-based action recognition. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 140–149, 2020.

[70] S. R. Livingstone and F. A. Russo. The ryerson audio-visual database of emotional speech and song (ravdess): A dynamic, multimodal set of facial and vocal expressions in north american english. *PloS one*, 13(5):e0196391, 2018.

[71] C. Louizos, M. Welling, and D. P. Kingma. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations (ICLR)*, 2018.

[72] 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.

[73] 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.

[74] R. K. Mahabadi, J. Henderson, and S. Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. In *Advances in Neural Information Processing Systems*, 2021.

[75] A. Mesaros, T. Heittola, and T. Virtanen. Tut acoustic scenes 2017, evaluation dataset, Nov. 2017.

[76] S. Miao, H. Xu, Z. Han, and Y. Zhu. Recognizing facial expressions using a shallow convolutional neural network. *IEEE Access*, 7:78000–78011, 2019.

[77] O. Michel. Cyberbotics ltd. webots™: professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1):5, 2004.

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

[79] 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.

[80] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations (ICLR)*, 2017.

[81] A. Mollahosseini, B. Hasani, and M. H. Mahoor. Affectnet: A database for facial expression, valence, and arousal computing in the wild. *IEEE Transactions on Affective Computing*, 10(1):18–31, 2017.

[82] H. K. Mousavi, G. Liu, W. Yuan, M. Takáč, H. Muñoz-Avila, and N. Motee. A layered architecture for active perception: Image classification using deep reinforcement learning. *arXiv preprint arXiv:1909.09705*, 2019.

[83] 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.

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

[85] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729, 2008.

[86] D. Osokin. Real-time 2d multi-person pose estimation on cpu: Lightweight openpose, 2018.

[87] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. 2015.

[88] 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.

[89] 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.

[90] 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. *Pattern Recognition Letters*, 136:183–189, 2020.

[91] 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.

[92] J. Pfeiffer, A. Kamath, A. Rücklé, K. Cho, and I. Gurevych. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, 2021.

[93] C. Plizzari, M. Cannici, and M. Matteucci. Skeleton-based action recognition via spatial and temporal transformer networks. *Computer Vision and Image Understanding*, 208:103219, 2021.

[94] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, Nov. 2016. Association for Computational Linguistics.

[95] S.-A. Rebuffi, H. Bilen, and A. Vedaldi. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[96] S.-A. Rebuffi, H. Bilen, and A. Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[97] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, Dec 2015.

[98] 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.

[99] V. Sanh, T. Wolf, and A. M. Rush. Movement pruning: Adaptive sparsity by fine-tuning. In *NeurIPS*, 2020.

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

[101] 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.

[102] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang. NTU RGB+D: A large scale dataset for 3D human activity analysis. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1010–1019, 2016.

[103] M. Shen and J. P. How. Active perception in adversarial scenarios using maximum entropy deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3384–3390. IEEE, 2019.

[104] J. Sherrah and S. Gong. Fusion of perceptual cues for robust tracking of head pose and position. *Pattern Recognition*, 34(8):1565–1572, 2001.

[105] L. Shi, Y. Zhang, J. Cheng, and H. Lu. Skeleton-based action recognition with directed graph neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7912–7921, 2019.

[106] L. Shi, Y. Zhang, J. Cheng, and H. Lu. Skeleton-based action recognition with multi-stream adaptive graph convolutional networks. *IEEE Transactions on Image Processing*, 29:9532–9545, 2020.

[107] C. Si, W. Chen, W. Wang, L. Wang, and T. Tan. An attention enhanced graph convolutional lstm network for skeleton-based action recognition. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1227–1236, 2019.

[108] H. Siqueira, S. Magg, and S. Wermter. Efficient facial feature learning with wide ensemble-based convolutional neural networks. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 5800–5809, 2020.

[109] K. Sun, B. Xiao, D. Liu, and J. Wang. Deep high-resolution representation learning for human pose estimation, 2019.

[110] X. Sun, X. Ren, S. Ma, and H. Wang. meProp: Sparsified back propagation for accelerated deep learning with reduced overfitting. In *Proceedings of the 34th International Conference on Machine Learning (ICLR)*, volume 70 of *Proceedings of Machine Learning Research*, pages 3299–3308, International Convention Centre, Sydney, Australia, 2017.

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

[112] C. Symeonidis, I. Mademlis, I. Pitas, and N. Nikolaidis. AUTH-persons: A dataset for detecting humans in crowds from aerial views. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 596–600, 2022.

[113] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler. Efficient transformers: A survey. *arXiv:2009.06732*, 2020.

[114] P. Tosidis, N. Passalis, and A. Tefas. Active vision control policies for face recognition using deep reinforcement learning. In *2022 30th European Signal Processing Conference (EUSIPCO)*, pages 1087–1091, 2022.

[115] D. T. Tran, N. Passalis, A. Tefas, M. Gabbouj, and A. Iosifidis. Attention-based neural bag-of-features learning for sequence data. *arXiv preprint arXiv:2005.12250*, 2020.

[116] Y.-H. H. Tsai, S. Bai, P. P. Liang, J. Z. Kolter, L.-P. Morency, and R. Salakhutdinov. Multimodal transformer for unaligned multimodal language sequences. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2019, page 6558. NIH Public Access, 2019.

[117] G. Tzanetakis, G. Essl, and P. Cook. Automatic musical genre classification of audio signals, 2001.

[118] M. Tzelepi and A. Tefas. Improving the performance of lightweight cnns for binary classification using quadratic mutual information regularization. *Pattern Recognition*, 106:107407, 2020.

[119] 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.

[120] 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.

[121] K. Wang, X. Peng, J. Yang, S. Lu, and Y. Qiao. Suppressing uncertainties for large-scale facial expression recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6897–6906, 2020.

[122] K. Wang, X. Peng, J. Yang, D. Meng, and Y. Qiao. Region attention networks for pose and occlusion robust facial expression recognition. *IEEE Transactions on Image Processing*, 29:4057–4069, 2020.

[123] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma. Linformer: Self-attention with linear complexity. *arXiv:2006.04768*, 2020.

[124] X. Wang, S. Zhang, Z. Qing, Y. Shao, Z. Zuo, C. Gao, and N. Sang. Oadtr: Online action detection with transformers. *International Conference on Computer Vision (ICCV)*, 2021.

[125] B. Xie, Y. Liang, and L. Song. Diverse neural network learns true target functions. In *Artificial Intelligence and Statistics*, pages 1216–1224, 2017.

[126] Y. Xiong, Z. Zeng, R. Chakraborty, M. Tan, G. Fung, Y. Li, and V. Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[127] M. Xu, M. Gao, Y.-T. Chen, L. Davis, and D. Crandall. Temporal recurrent networks for online action detection. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5531–5540, 2019.

[128] M. Xu, Y. Xiong, H. Chen, X. Li, W. Xia, Z. Tu, and S. Soatto. Long short-term transformer for online action detection. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[129] M. Yan, Q. Sun, I. Frosio, S. Tyree, and J. Kautz. How to close sim-real gap? transfer with segmentation! *arXiv preprint arXiv:2005.07695*, 2020.

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

[131] H. Yang, D. Yan, L. Zhang, Y. Sun, D. Li, and S. J. Maybank. Feedback graph convolutional network for skeleton-based action recognition. *IEEE Transactions on Image Processing*, 31:164–175, 2021.

[132] S.-K. Yeom, P. Seegerer, S. Lapuschkin, A. Binder, S. Wiedemann, K.-R. Müller, and W. Samek. Pruning by explaining: A novel criterion for deep neural network pruning. *Pattern Recognition*, 115, 2021.

[133] A. Zadeh, P. P. Liang, S. Poria, P. Vij, E. Cambria, and L.-P. Morency. Multi-attention recurrent network for human communication comprehension. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[134] J. Zeng, S. Shan, and X. Chen. Facial expression recognition with inconsistently annotated datasets. In *European Conference on Computer Vision*, 2018.

[135] A. Zhang, Y. Tay, S. Zhang, A. Chan, A. T. Luu, S. Hui, and J. Fu. Beyond fully-connected layers with quaternions: Parameterization of hypercomplex multiplications with $1/n$ parameters. In *International Conference on Learning Representations (ICLR)*, 2021.

[136] P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue, and N. Zheng. View adaptive recurrent neural networks for high performance human action recognition from skeleton data. In *IEEE International Conference on Computer Vision*, pages 2117–2126, 2017.

[137] 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.

[138] P. Zhao, J. Wang, L. Xie, Y. Zhang, Y. Wang, and Q. Tian. Privileged knowledge distillation for online action detection. *preprint, arXiv:2011.09158*, abs/2011.09158, 2020.

[139] W. Zhao, J. P. Queralta, and T. Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744. IEEE, 2020.

[140] Z. Zhao, Q. Liu, and S. Wang. Learning deep global multi-scale and local attention features for facial expression recognition in the wild. *IEEE Transactions on Image Processing*, 30:6544–6556, 2021.

[141] Z. Zhao, Q. Liu, and F. Zhou. Robust lightweight facial expression recognition network with label distribution training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3510–3519, 2021.

[142] Y. Zhu, J. Feng, C. Zhao, M. Wang, and L. Li. Counter-interference adapter for multilingual machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2812–2823, 2021.

8 Appendix

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

Publication place: European Conference on Computer Vision 2022.
The appended paper follows.

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

Lukas Hedegaard and Alexandros Iosifidis

Department of Electrical and Computer Engineering, Aarhus University, Denmark
{lhm,ai}@ece.au.dk

Abstract. We introduce *Continual* 3D Convolutional Neural Networks (*Co*3D CNNs), a new computational formulation of spatio-temporal 3D CNNs, in which videos are processed frame-by-frame rather than by clip. In online tasks demanding frame-wise predictions, *Co*3D CNNs dispense with the computational redundancies of regular 3D CNNs, namely the repeated convolutions over frames, which appear in overlapping clips. We show that *Continual* 3D CNNs can reuse preexisting 3D-CNN weights to reduce the per-prediction floating point operations (FLOPs) in proportion to the temporal receptive field while retaining similar memory requirements and accuracy. bgithuThis is validated with multiple models on Kinetics-400 and Charades with remarkable results: *Co*X3D models attain state-of-the-art complexity/accuracy trade-offs on Kinetics-400 with 12.1−15.3× reductions of FLOPs and 2.3−3.8% improvements in accuracy compared to regular X3D models while reducing peak memory consumption by up to 48%. Moreover, we investigate the transient response of *Co*3D CNNs at start-up and perform extensive benchmarks of on-hardware processing characteristics for publicly available 3D CNNs.

Keywords: 3D CNN, Human Activity Recognition, Efficient, Stream Processing, Online Inference, Continual Inference Network.

1 Introduction

Through the availability of large-scale open-source datasets such as ImageNet [37] and Kinetics [25], [4], 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 to multiple breakthroughs in the state-of-the-art [3], [43], [11], [10]. Despite their success in competitions and benchmarks where only prediction quality is evaluated, 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 [17], [48], [42] and video recognition [28], [10], [49], pruning of network weights [6], [13], [14], knowledge distillation [16], [47], [36], and network quantisation [19], [2], [12].

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 [26], [39] using specialised architectures. However, these are not weight-compatible with regular 3D CNNs. 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. Contrary to most deep learning papers, the work presented here needed no training; our goal was to validate the efficacy of converting regular 3D CNNs to Continual CNNs directly, and to explore their characteristics in the online recognition domain. Accordingly, we perform conversions from five 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 after 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. Full source code is available at `https://github.com/lukashedegaard/co3d`.

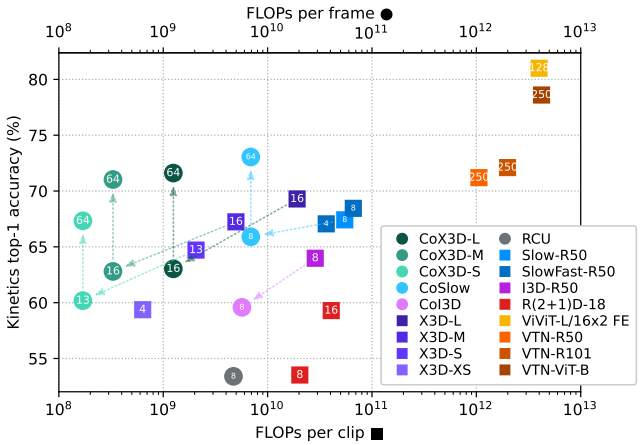


Fig. 1: **Accuracy/complexity trade-off** for *Continual* 3D CNNs and recent state-of-the-art methods on Kinetics-400 using 1-clip/frame testing. ■ FLOPs per *clip* are noted for regular networks, while ● FLOPs per *frame* are shown for the Continual 3D CNNs. Frames per clip / global average pool size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

2 Related Works

2.1 3D CNNs for video recognition

Convolutional Neural Networks with spatio-temporal 3D kernels may be considered the natural extension of 2D CNNs for image recognition to CNNs for video recognition. Although they did not surpass their 2D CNN + RNN competitors [7], [21] initially [20], [23], [44], arguably due to a high parameter count and insufficient dataset size, 3D CNNs have achieved state-of-the-art results on Human Action Recognition tasks [3], [43], [11] since the Kinetics dataset [25] was introduced. While recent large-scale Transformer-based methods [1], [32] have become leaders in terms of accuracy, 3D CNNs still achieve state-of-the-art accuracy/complexity trade-offs. Nevertheless, competitive accuracy comes with high computational cost, which is prohibitive to many real-life use cases.

In image recognition, efficient architectures such as MobileNet [17], ShuffleNet [48], and EfficientNet [42] attained improved accuracy-complexity trade-offs. These architectures were extended to the 3D-convolutional versions 3D-MobileNet [28], 3D-ShuffleNet [28] and X3D [10] (≈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 updates predictions for each frame; real-time processing rates can only be achieved with the smallest models at severely reduced accuracy. 3D CNNs suffer from 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

A well-explored approach to video-recognition [7], [21], [22], [40] is to let each frame pass through a 2D CNN trained on ImageNet in one stream alongside a second stream of Optical Flow [9] and integrate these using a recurrent network. Such architectures requires no network modification for deployment in online-processing scenarios, lends themselves to caching [46], and are free of the computational redundancies experienced in 3D CNNs. However, the overhead of Optical Flow and costly feature-extractors pose a substantial disadvantage.

Another approach is to utilise 3D CNNs for feature extraction. In [31], spatio-temporal features from non-overlaping clips are used to train a recurrent network for hand gesture recognition. In [27], a 3D CNN processes a sliding window of the input to perform spatio-temporal action detection. These 3D CNN-based methods have the disadvantage of either not producing predictions for each input frame [31] or suffering from redundant computations from overlapping clips [27].

Massively Parallel Video Networks [5] split a DNN into depth-parallel subnetworks across multiple computational devices to improve online multi-device

parallel processing performance. While their approach treats networks layers as atomic operations and doesn't tackle the fundamental redundancy of temporal convolutions, *Continual* 3D CNNs reformulate the network layers, remove redundancy, and accelerate inference on single devices as well.

Exploring modifications of the spatio-temporal 3D convolution operating frame by frame, the Recurrent Convolutional Unit (RCU) [39] replaces the 3D convolution by aggregating a spatial 2D convolution over the current input with a 1D convolution over the prior output. Dissected 3D CNNs [26] (D3D) cache the $1 \times n_H \times n_W$ frame-level features in network residual connections and aggregate them with the current frame features via $2 \times 3 \times 3$ convolutions. Like the our proposed *Continual* 3D CNNs, both RCU and D3D are causal and operate frame-by-frame. However, they are speciality architectures, which are incompatible with pre-trained 3D CNNs, and must be trained from scratch. We reformulate spatio-temporal convolutions in a one-to-one compatible manner, allowing us to reuse existing model weights.

3 Continual Convolutional Neural Networks

3.1 Regular 3D-convolutions lead to redundancy

Currently, the best performing architectures (e.g., X3D [10] and SlowFast [11]) 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:

$$\Theta([k_H \cdot k_W \cdot k_T + b] \cdot c_I \cdot c_O \cdot n_H \cdot n_W \cdot n_T), \tag{1}$$

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

$$\Theta(c_I \cdot m_H \cdot m_W \cdot [m_T - 1])), \tag{2}$$

and during inference the network has to transiently store feature-maps of size

$$\Theta(c_O \cdot n_H \cdot n_W \cdot n_T). \tag{3}$$

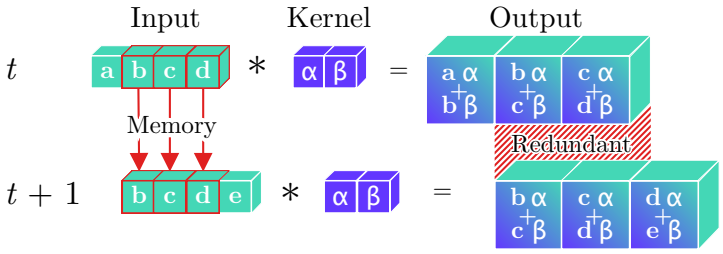


Fig. 2: **Redundant computations** for a temporal convolution during online processing, as illustrated by the repeated convolution of inputs (green $\mathbf{b}, \mathbf{c}, \mathbf{d}$) with a kernel (blue α, β) in the temporal dimension. Moreover, prior inputs ($\mathbf{b}, \mathbf{c}, \mathbf{d}$) must be stored between time-steps for online processing tasks.

3.2 Continual Convolutions

We can remedy the issue described in Sec. 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 [34] 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 1.1.

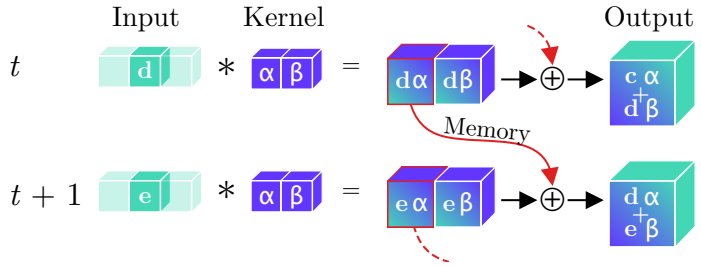


Fig. 3: **Continual Convolution**. An input (green $\mathbf{d}$ or $\mathbf{e}$) is convolved with a kernel (blue α, β). The intermediary feature-maps corresponding to all but the last temporal position are stored, while the last feature map and prior memory are summed to produce the resulting output. For a continual stream of inputs, Continual Convolutions produce identical outputs to regular convolutions.

```
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 1.1: **Pseudo-code** for Continual Convolution. Ready-to-use modules are available in the Continual Inference library [15].

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:

$$\Theta([k_H \cdot k_W \cdot k_T + b] \cdot c_I \cdot c_O \cdot n_H \cdot n_W). \tag{4}$$

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 overhead of storing the partially computed feature-maps for a frame is:

$$\Theta(d_T \cdot [k_T - 1] \cdot c_O \cdot n_H \cdot n_W). \tag{5}$$

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

$$\Theta(c_O \cdot n_H \cdot n_W). \tag{6}$$

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

$$\Theta(d_T \cdot [k_T - p_T - 1]). \tag{7}$$

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 operation with temporal kernel size k_T and spatial output size $n_H \cdot n_W$, the memory consumption is

$$\mathcal{O}([k_T - 1] \cdot n_H \cdot n_W), \tag{9}$$

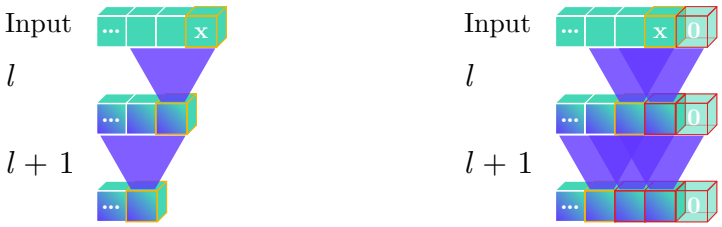
and the delay is

$$\mathcal{O}(k_T - p_T - 1). \tag{10}$$

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 [18] 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 feature-maps in consecutive CNN layers. 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. 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



(a) No padding (b) Zero padding

Fig. 4: **Issue with temporal padding**: The latest frame **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 **x** following a layer l increases with the number of padded zeros.

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" [24]. The use of zero-padding, however, has the downside that it alters the input-distribution along the boarders significantly [29], [33]. 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 a 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 a "model shift" in the conversion from regular 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 $r_T - p_T - 1$ frames where r_T and p_T are the aggregated temporal receptive field and padding of the network. For example, Continual X3D-{S, M, L} models have receptive fields of size $\{69, 72, 130\}$, aggregated padding $\{28, 28, 57\}$, and hence need to process $\{40, 43, 72\}$ frames prior to normal operation. The initial response depends on how internal state variables are initialised. In Sec. 4.2, we explore this further with two initialisation 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 Design considerations

Memory consumption is highly dependent on the clip size employed by the respective models. Disregarding the storage requirement of the model weights (which is identical between for regular and continual 3D CNNs), X3D-M has a worst-case total memory-consumption of 7,074,816 floats when prior frames

and the transient feature-maps are taken into account. Its continual counterpart, *Co*X3D-M, has a worst case memory only 5,072,688 floats. How can this be? Since Continual 3D CNNs do not store prior input frames and has smaller transient feature maps, the memory savings outweigh the cost of caching features in each continual layer. Had the clip size been four instead of sixteen, X3D-M_4 would have had a worst-case memory consumption of 1,655,808 floats and *Co*X3D-M_4 of 5,067,504 floats, and for clip size of 64, X3D-M_{64} consumes 28,449,792 floats and *Co*X3D-M_{64} uses 5,093,424 floats. The memory consumption of regular 3D CNNs is this thus highly dependent on the clip size, while *Co*3D CNNs are not. 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, 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 3D CNNs [3], [43], [11], [10]. 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 20.5% of the total model state memory (see Appendix A).

3.8 Training

*Co*3D CNNs are trained with back-propagation like other neural networks. However, special care must be taken in the estimation of data statistics in normalisation layers: 1) Momentum should be adjusted to $\text{mom}_{\text{step}} = 2/(1 + \text{timesteps} \cdot (2/\text{mom}_{\text{clip}} - 1))$ to match the exponential moving average dynamics of clip-based training, where T is the clip size; 2) statistics should not be tracked for the transient response. Alternatively, they can be trained offline in their "unrolled" regular 3D-CNN form with no temporal padding. This is similar to utilising pre-trained weights from a regular 3D CNN, as we do in our experiments.

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 publicly available pre-trained network weights of regular 3D CNNs [11], [10], [8] without further fine-tuning. Data pre-processing follows the respective procedures associated with the originating weights unless stated otherwise. The section is laid out as follows: First, we showcase the network performance following weight transfer from regular to Continual 3D on multiple datasets for Human Activity Recognition.

	Model	Acc. (%)	Par. (M)	Mem. (MB)	FLOPs (G)	Throughput (preds/s) CPU	TX2	Xavier	2080Ti
Clip	I3D-R50	63.98	28.04	191.59	28.61	0.93	2.54	9.20	77.15
	R(2+1)D-18$_8$	53.52	31.51	168.87	20.35	1.75	3.19	6.82	130.88
	R(2+1)D-18$_{16}$	59.29	31.51	215.44	40.71	0.83	1.82	3.77	75.81
	Slow-8×8-R50	67.42	32.45	266.04	54.87	0.38	1.34	4.31	61.92
	SlowFast-8×8-R50	68.45	66.25	344.84	66.25	0.34	0.87	2.72	30.72
	SlowFast-4×16-R50	67.06	34.48	260.51	36.46	0.55	1.33	3.43	41.28
	X3D-L	69.29	6.15	240.66	19.17	0.25	0.19	4.78	36.37
	X3D-M	67.24	3.79	126.29	4.97	0.83	1.47	17.47	116.07
	X3D-S	64.71	3.79	61.29	2.06	2.23	2.68	42.02	276.45
	X3D-XS	59.37	3.79	28.79	0.64	8.26	8.20	135.39	819.87
Frame	RCU$_8$ [39]†	53.40	12.80	-	4.71	-	-	-	-
	*Co*I3D$_8$	59.58	28.04	235.87	5.68	3.00	2.41	14.88	125.59
	*Co*I3D$_{64}$	56.86	28.04	236.08	5.68	3.15	2.41	14.89	126.32
	*Co*Slow$_8$	65.90	32.45	175.98	6.90	2.80	1.60	6.18	113.77
	***Co*Slow$_{64}$**	**73.05**	**32.45**	**176.41**	**6.90**	**2.92**	**1.60**	**6.19**	**102.00**
	*Co*X3D-L$_{16}$	63.03	6.15	184.29	1.25	2.30	0.99	25.17	206.65
	***Co*X3D-L$_{64}$**	**71.61**	**6.15**	**184.37**	**1.25**	**2.30**	**0.99**	**27.56**	**217.53**
	*Co*X3D-M$_{16}$	62.80	3.79	68.88	0.33	7.57	7.26	88.79	844.73
	***Co*X3D-M$_{64}$**	**71.03**	**3.79**	**68.96**	**0.33**	**7.51**	**7.04**	**86.42**	**796.32**
	*Co*X3D-S$_{13}$	60.18	3.79	41.91	0.17	13.16	11.06	219.64	939.72
	***Co*X3D-S$_{64}$**	**67.33**	**3.79**	**41.99**	**0.17**	**13.19**	**11.13**	**213.65**	**942.97**

Table 1: **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, predictions per second denote frames per second for the *Co*X3D models and clips per second for the remaining models. Throughput results are the mean of 100 measurements. Pareto-optimal models are marked with bold. Mem. is the maximum allocated memory during inference noted in megabytes. †Approximate FLOPs derived from paper (see Appendix C).

This is followed by a study on the transient response of *Co*3D CNNs at startup. Subsequently, 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 on-hardware benchmark of prior methods and Continual 3D CNNs, measuring the 1-clip/frame accuracy of publicly available models, as well as their inference throughput on various computational devices.

4.1 Transfer from regular to Continual CNNs

To gauge direct transferability of 3D CNN weights, we implement continual versions of various 3D CNNs and initialise them with their publicly available weights for Kinetics-400 [25] and Charades [38]. While it is common to use an ensemble prediction from multiple clips to boost video-level accuracy on these benchmarks, we abstain from this, as it doesn't apply to online-scenarios. Instead, we report the single-clip/frame model performance.

Kinetics-400. We evaluate the X3D network variants XS, S, M, and L on the test set using one temporally centred clip from each video. The XS network is omitted

	Model	FLOPs (G) × views	mAP (%)
Clip	Slow-8×8 [11]	54.9 × 30	39.0
	Slow-8×8 [11]†	54.9 × 1	21.4
	Slow-8×8 (ours)	54.9 × 1	24.1
Fr.	*Co*Slow$_8$	6.9 × 1	21.5
	*Co*Slow$_{64}$	6.9 × 1	25.2

Table 2: **Charades benchmark.** Noted are the FLOPs × views and video-level mean average precision (mAP) on the validation set using pre-trained model weights. †Results achieved using the publicly available SlowFast code [11].

in the transfer to *Co*X3D, given that it is architecturally equivalent to S, but with fewer frames per clip. In evaluation on Kinetics-400, we faced the challenge that videos were limited to 10 seconds. Due to the longer transient response of Continual CNNs (see Sec. 4.2) and low frame-rate used for training X3D models ($5.0, 6.0, 6.0$ FPS for S, M, and L), the video-length was insufficient to reach steady-state for some models. 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 X3D transfer are shown in Tab. 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 12.1×, 15.1×, and 15.3× respectively. The savings do not quite reach the clip sizes since the final pooling and prediction layers are active for each frame. As a side-effect of the transfer from zero-padded regular CNN to Continual CNN without zero-padding, we see a notable reduction in accuracy. This is easily improved by using an extended pooling size for the network (discussed in Sec. 3.7 and in Sec. 4.2). 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, Kinetics dataset did not have sufficient frames to fill the temporal receptive field of all models in these tests. We explore this further in Sections 4.2 and 4.2.

Charades. To showcase the generality of the approach, we repeat the above described procedure with another 3D CNN, the *Co*Slow network [11]. We report the video-level mean average precision (mAP) of the validation split alongside the FLOPs per prediction in Tab. 2. Note the accuracy discrepancy between 30 view (10 temporal positions with 3 spatial positions each) and 1 view (spatially and temporally centred) evaluation. As observed on Kinetics, the *Co*Slow network reduces the FLOPs per prediction proportionally with the original clip size (8 frames), and can recover accuracy by extending the global average pool size.

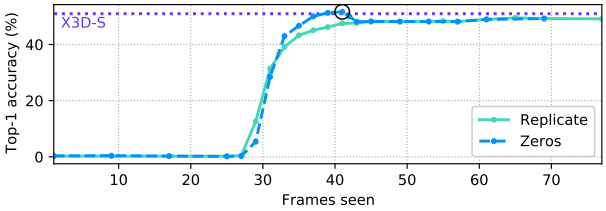
4.2 Ablation Experiments

As described in Sec. 3.6, Continual CNNs exhibit a transient response during their up-start. In order to gauge this response, we perform ablations on the

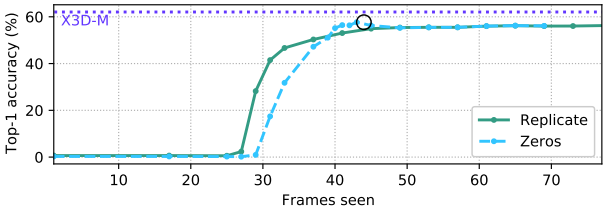
Kinetics-400 validation set, this time sampled at 15 FPS to have a sufficient number of frames available. This corresponds to a data domain shift [45] relative to the pre-trained weights, where time advances slower.

Transient response of Continual CNNs. Our expected upper bound is given by the baseline X3D network 1-clip accuracy at 15 FPS. The transient response is measured by varying the number of prior frames used for initialisation before evaluating a frame using the *Co*X3D model. 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 counters 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. We found results to vary up to 8% for a X3D-S network evaluated at different video positions. The experiment is repeated for two initialisation schemes, "zeros" (used in other experiments) and "replicate", and two model sizes, S and M. The transient responses are shown in Fig. 5.

For all responses, the first ≈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, topping with peaks of 51.6% and 57.6% at 41 and 44 frames seen as discussed in Sec. 3.6. This makes sense considering that the original network weights were trained with



(a) *Co*X3D-S



(b) *Co*X3D-M

Fig. 5: **Transient response** for Continual X3D-{S,M} on the Kinetics-400 val at 15 FPS. Dotted horizontal lines denote X3D validation accuracy for 1-clip predictions. Black circles highlight the theoretically required initialisation frames.

this exact amount of zero-padding. Adding more frames effectively replaces the padded zeros and causes a slight drop of accuracy in the steady state, where the accuracy settles at the same values as for the "replication" scheme.

Extended receptive field. Continual CNNs experience a negligible increase in computational cost when larger temporal receptive field are used (see Sec. 3.7). 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. The Continual CNNs are evaluated at frames corresponding to the steady state.

Tab. 3 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. As found in Sec. 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.

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	166,565	69
		16	50.1	166,567	72
		32	54.7	166,574	88
		64	59.8	166,587	120
		96	*61.8*	166,601	152
	M	16	56.3	325,456	72
		32	60.7	325,463	88
		64	64.9	325,477	120
		96	*67.3*	325,491	152
	L	16	53.0	1,245,549	130
		32	58.5	1,245,556	146
		64	*64.3*	1,245,570	178
		96	*66.3*	1,245,584	210

Table 3: **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 val. accuracy. *Italic numbers* denote measurement taken within the transient response due to a lack of frames in the video-clip.

4.3 Inference benchmarks

Despite their high status in activity recognition leader-boards [35], it is unclear how recent 3D CNNs methods perform in the online setting, where speed and accuracy constitute a necessary trade-off. To the best of our knowledge, there has not yet been a systematic evaluation of throughput for these video-recognition models on real-life hardware. In this set of experiments, we benchmark the FLOPs, parameter count, maximum allocated memory and 1-clip/frame accuracy of I3D [3], R(2+1)D [43], SlowFast[41], X3D [10], *Co*I3D, *Co*Slow, and *Co*X3D. To gauge achievable throughputs at different computational budgets, networks were tested on four hardware platforms as described in Appendix B.

As seen in the benchmark results found in Tab. 1, the limitation to one clip markedly lowers accuracy compared with the multi-clip evaluation published in the respective works [3], [43], [11], [10]. Nontheless, the Continual models with extended receptive fields attain the best accuracy/speed trade-off by a large margin. For example, *Co*X3D-L_{64} on the Nvidia Jetson Xavier achieves an accuracy of 71.3% at 27.6 predictions per second compared to 67.2% accuracy at 17.5 predictions per second for X3D-M while reducing maximum allocated memory by 48%! Confirming the observation in [30], we find that the relation between model FLOPs and throughput varies between models, with better ratios attained for simpler models (e.g., I3D) than for complicated ones (e.g., X3D). This relates to different memory access needs and their cost. Tailor-made hardware could plausibly reduce these differences. Supplementary visualisation of the results in Tab. 1 are found in Appendix C.

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, giving up to a 15.1× reduction of floating point operations, a 9.2× real-life inference speed-up on CPU, 48% peak memory reduction, and an accuracy improvement of 5.6% 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 these as important directions 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).

References

1. Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., Schmid, C.: Vivit: A video vision transformer. In: IEEE/CVF International Conference on Computer Vision (ICCV). pp. 6836–6846 (2021)
2. Cai, Z., He, X., Sun, J., Vasconcelos, N.: Deep learning with low precision by half-wave gaussian quantization. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5406–5414 (2017)
3. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4724–4733 (2017)
4. Carreira, J., Noland, E., Banki-Horvath, A., Hillier, C., Zisserman, A.: A short note about kinetics-600. preprint, arXiv:1808.01340 (2018)
5. Carreira, J., Pătrăucean, V., Mazare, L., Zisserman, A., Osindero, S.: Massively parallel video networks. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Proceedings of the European Conference on Computer Vision (ECCV). pp. 680–697 (2018)
6. Chen, W., Wilson, J.T., Tyree, S., Weinberger, K.Q., Chen, Y.: Compressing neural networks with the hashing trick. In: International Conference on International Conference on Machine Learning (ICML). p. 2285–2294 (2015)
7. Donahue, J., Hendricks, L.A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Darrell, T., Saenko, K.: Long-term recurrent convolutional networks for visual recognition and description. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2625–2634 (2015)
8. Fan, H., Murrell, T., Wang, H., Alwala, K.V., Li, Y., Li, Y., Xiong, B., Ravi, N., Li, M., Yang, H., Malik, J., Girshick, R., Feiszli, M., Adcock, A., Lo, W.Y., Feichtenhofer, C.: PyTorchVideo: A deep learning library for video understanding. In: ACM International Conference on Multimedia (2021), https://pytorchvideo.org/
9. Farnebäck, G.: Two-frame motion estimation based on polynomial expansion. In: Image Analysis. pp. 363–370. Springer Berlin Heidelberg (2003)
10. Feichtenhofer, C.: 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.
11. Feichtenhofer, C., Fan, H., Malik, J., He, K.: 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.
12. Floropoulos, N., Tefas, A.: Complete vector quantization of feedforward neural networks. Neurocomputing **367**, 55–63 (2019)
13. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In: International Conference on Learning Representations (ICLR) (2016)
14. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 1398–1406 (2017)
15. Hedegaard, L., Iosifidis, A.: Continual inference: A library for efficient online inference with deep neural networks in pytorch. preprint, arXiv:2204.03418 (2022)
16. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. In: NIPS Deep Learning and Representation Learning Workshop (2015)
17. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. preprint, arXiv:1704.04861 **abs/1704.04861** (2017)

18. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7132–7141 (2018)
19. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized neural networks. In: Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 29. Curran Associates, Inc. (2016)
20. Ji, S., Xu, W., Yang, M., Yu, K.: 3d convolutional neural networks for human action recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) **35**(1), 221–231 (2013)
21. Joe Yue-Hei Ng, Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., Toderici, G.: Beyond short snippets: Deep networks for video classification. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4694–4702 (2015)
22. Kalogeiton, V., Weinzaepfel, P., Ferrari, V., Schmid, C.: Action tubelet detector for spatio-temporal action localization. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 4415–4423 (2017)
23. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1725–1732 (2014)
24. Karpathy, A.: CS231n convolutional neural networks for visual recognition, https://cs231n.github.io/convolutional-networks/. Last visited on 2021/01/26
25. Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., Suleyman, M., Zisserman, A.: The kinetics human action video dataset. preprint, arXiv:1705.06950 (2017)
26. Köpüklü, O., Hörmann, S., Herzog, F., Cevikalp, H., Rigoll, G.: Dissected 3d cnns: Temporal skip connections for efficient online video processing. preprint, arXiv:2009.14639 (2020)
27. Köpüklü, O., Wei, X., Rigoll, G.: You only watch once: A unified cnn architecture for real-time spatiotemporal action localization. preprint, arXiv:1911.06644 (2019)
28. Köpüklü, O., Kose, N., Gunduz, A., Rigoll, G.: Resource efficient 3d convolutional neural networks. In: IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). pp. 1910–1919 (2019)
29. Liu, G., Shih, K.J., Wang, T.C., Reda, F.A., Sapra, K., Yu, Z., Tao, A., Catanzaro, B.: Partial convolution based padding. preprint, arXiv:1811.11718 pp. 1–11 (2018)
30. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: Proceedings of the European Conference on Computer Vision (ECCV) (September 2018)
31. Molchanov, P., Yang, X., Gupta, S., Kim, K., Tyree, S., Kautz, J.: 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). pp. 4207–4215 (2016)
32. Neimark, D., Bar, O., Zohar, M., Asselmann, D.: Video transformer network. In: 2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW). pp. 3156–3165 (2021)
33. Nguyen, A., Choi, S., Kim, W., Ahn, S., Kim, J., Lee, S.: Distribution padding in convolutional neural networks. In: International Conference on Image Processing (ICIP). pp. 4275–4279 (2019)
34. van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K.: Wavenet: A generative model for raw audio. preprint, arXiv:1609.03499 (2016)

35. Papers with Code: Kinetics-400 leaderboard, https://paperswithcode.com/sota/action-classification-on-kinetics-400 Last visited on 2021/02/03.
36. Passalis, N., Tefas, A.: Learning deep representations with probabilistic knowledge transfer. In: Proceedings of the European Conference on Computer Vision (ECCV) (2018)
37. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision (ICCV) **115**(3), 211–252 (2015)
38. Sigurdsson, G.A., Varol, G., Wang, X., Farhadi, A., Laptev, I., Gupta, A.: Hollywood in homes: Crowdsourcing data collection for activity understanding. In: Proceedings of the European Conference on Computer Vision (ECCV) (2016)
39. Singh, G., Cuzzolin, F.: Recurrent convolutions for causal 3d cnns. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). pp. 1456–1465 (2019)
40. Singh, G., Saha, S., Sapienza, M., Torr, P., Cuzzolin, F.: Online real-time multiple spatiotemporal action localisation and prediction. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 3657–3666 (2017)
41. Sovrasov, V.: Ptflops, https://github.com/sovrasov/flops-counter.pytorch. MIT License. Last visited on 2021/03/02
42. Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural networks. In: Proceedings of Machine Learning Research. vol. 97, pp. 6105–6114 (2019)
43. Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., Paluri, M.: A closer look at spatiotemporal convolutions for action recognition. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6450–6459 (2018)
44. Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3d convolutional networks. In: IEEE International Conference on Computer Vision (ICCV). pp. 4489–4497 (2015)
45. Wang, M., Deng, W.: Deep visual domain adaptation: A survey. Neurocomputing **312**, 135–153 (2018)
46. Xu, M., Zhu, M., Liu, Y., Lin, F., Liu, X.: Deepcache: Principled cache for mobile deep vision. International Conference on Mobile Computing and Networking (2018)
47. Yim, J., Joo, D., Bae, J., Kim, J.: A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7130–7138 (2017)
48. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6848–6856 (2018)
49. Zhu, L., Sevilla-Lara, L., Yang, Y., Feiszli, M., Wang, H.: 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 Tab. 5. An overview of the resulting computations can be found in Tab. 4. The total memory overhead for the network state is 4,771,632 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 **5,072,688** 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 5,067,504 floats (−0.1%). Increasing the clip size to 64 would yield an increased state memory of 20,736 floats giving a total worst-case memory of 5,093,424 floats (+0.4%).

Stage	**Layer**		**Mem.** (floats)
conv_1	conv_T	$(5-1)\times 24\times 112\times 112 =$	1,204,224
res_2	residual_1	$(3-1-1)\times 24\times 112\times 112 =$	301,056
	residual_{2-3}	$[(3-1-1)\times 24\times 56\times 56]\times 2 =$	150,528
	conv_{1-3}	$[(3-1-1)\times 54\times 56\times 56]\times 3 =$	508,032
res_3	residual_1	$(3-1-1)\times 24\times 56\times 56 =$	75,264
	residual_{2-5}	$[(3-1-1)\times 48\times 28\times 28]\times 4 =$	150,528
	conv_{1-5}	$[(3-1)\times 108\times 28\times 28]\times 5 =$	846,720
res_4	residual_1	$(3-1-1)\times 48\times 28\times 28 =$	37,632
	residual_{2-11}	$[(3-1-1)\times 96\times 14\times 14]\times 10 =$	188,160
	conv_{1-11}	$[(3-1)\times 216\times 14\times 14]\times 11 =$	931,392
res_5	residual_1	$(3-1-1)\times 96\times 14\times 14 =$	18,816
	residual_{2-3}	$[(3-1-1)\times 192\times 7\times 7]\times 6 =$	56,448
	conv_{1-3}	$[(3-1)\times 432\times 7\times 7]\times 7 =$	296,352
pool_5	-	$(16-1)\times 432 =$	6,480
Total			**4,771,632**

Table 4: ***CoX*3D-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	1×1^2, #classes	$1 \times 1 \times 1$

Table 5: **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 [10].

B Benchmarking details

This section should be read in conjunction with Sec. 4.3 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 Tab. 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 Tab. 6. 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**	**Batch size**			
	$(T \times S^2)$	**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
Slow-8×8-R50	8×256^2	1	8	8	8
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*I3D	1×224^2	1	8	8	8
*Co*Slow	1×224^2	1	8	8	8
*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 6: **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.

Model	FLOPs	Throughput (evaluations/s)			
		CPU	TX2	Xavier	RTX
(*Co*)I3D	5.04×	3.39×	0.95×	1.62×	1.64×
(*Co*)Slow	7.95×	7.68×	1.19×	1.44×	1.65×
(*Co*)X3D-L	15.34×	9.20×	5.21×	5.77×	5.98×
(*Co*)X3D-M	15.06×	9.05×	4.79×	4.95×	6.86×
(*Co*)X3D-S	12.11×	5.91×	4.15×	4.98×	3.41×

Table 7: **Relative improvements** in frame-by-frame inference in Continual 3D CNN relative to regular 3D CNN counterparts. The improvements (× lower for FLOPs and × higher for throughput) correspond to the results in Tab. 1 of the main paper.

C A note on RCU FLOPs

In Tab. 1 of the main paper, we have approximated the FLOPs for RCU [39] as follows: We use a different measure of FLOPs (the one from the `ptflops` [41]) than the RCU authors and therefore employ a translation factor of 28.6/41.0, which is our measured FLOPs for I3D (28.6) divided by theirs (41.0), multiplied with their reported 54.0 for RCU. Considering that their method used 8 frames and can be applied per frame, we also divide by 8. Note that the this approximation lacks the repeat classification layer and may thus be considered on the low side. The resulting computation becomes $28.6/41.0 \cdot 54.0/8 = 4.71$.

D Supplemental visualisations of benchmark

As a supplement to the results presented in the main paper, this appendix supplies additional views of the benchmarking results in Tab. 1. Accordingly, graphical representations of the accuracy versus speed trade-offs from Tab. 1 are shown in Figures 6-9. As in Fig. 1 of the main paper, the noted accuracies on Kinetics-400 were achieved using 1-clip/frame testing on publicly available pretrained models, the *Co*X3D models utilised X3D weights without further fine-tuning, and the numbers noted in each point represent the size of the global average pooling layer. Likewise, Tab. 7 shows the improvements in continual inference relative to the regular models. In general, the FLOPs improvements are higher than on-hardware speed evaluations, with relatively lower improvements on hardware platforms with GPUs. We attribute these differences to a memory operations overhead, which does not enjoy the same computational improvement as multiply-accumulate operations do on massively parallel hardware.

From Figures 6-9 we likewise observe, that the I3D, R(2+1)D and SlowFast models perform relatively better on hardware compared to the X3D and *Co*X3D models, which utilise computation-saving approaches such as 1D-convolutions and grouped 3D-convolutions at the price of increasing memory access cost.

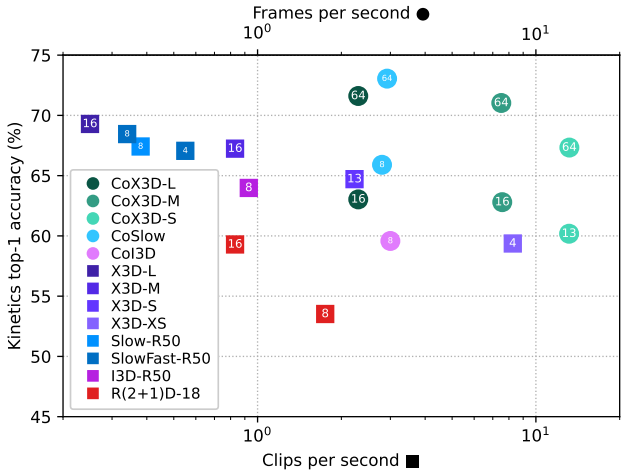


Fig. 6: **CPU** inference throughput versus top-1 accuracy on Kinetics-400.

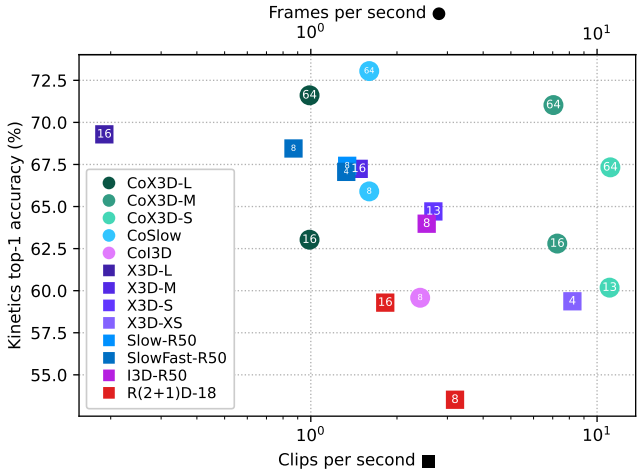


Fig. 7: **TX2** inference throughput versus top-1 accuracy on Kinetics-400.

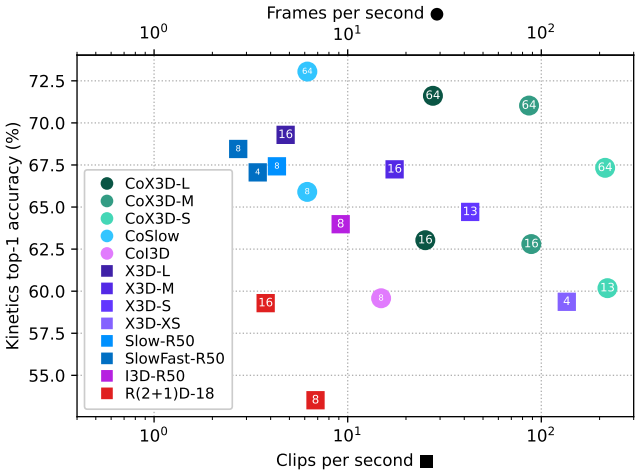


Fig. 8: **Xavier** inference throughput versus top-1 accuracy on Kinetics-400.

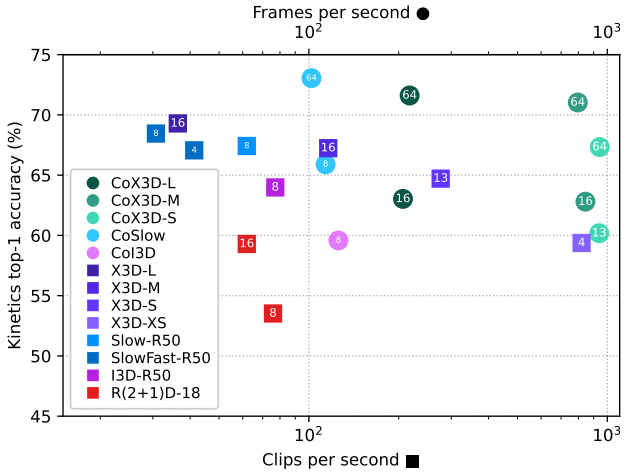


Fig. 9: **RTX2080Ti** inference throughput versus top-1 acc. on Kinetics-400.

8.2 Continual Inference: A Library for Efficient Online Inference with Deep Neural Networks in PyTorch

Publication place: European Conference on Computer Vision Workshop 2022 (Computational Aspects of Deep Learning). The appended paper follows.

Continual Inference: A Library for Efficient Online Inference with Deep Neural Networks in PyTorch

Lukas Hedegaard and Alexandros Iosifidis

Department of Electrical and Computer Engineering, Aarhus University, Denmark
{lhm,ai}@ece.au.dk

Abstract. We present *Continual Inference*, a Python library for implementing Continual Inference Networks (CINs), a class of Neural Networks designed for redundancy-free online inference. This paper offers a comprehensive introduction and guide to CINs and their implementation, as well as best-practices and code examples for composing basic modules into complex neural network architectures that perform online inference with an order of magnitude less floating-point operations than their non-CIN counterparts. Continual Inference provides drop-in replacements of PyTorch modules and is readily downloadable via the Python Package Index and at www.github.com/lukashedegaard/continual-inference.

Keywords: Online Inference, Continual Inference Network, Deep Neural Network, Python, PyTorch, Library.

1 Introduction

Designing and implementing Deep Neural Networks, which offer good performance in online inference scenarios, is an important but overlooked discipline in Deep Learning and Computer Vision. Research in areas such as Human Activity Recognition focuses heavily on improving accuracy on select benchmark datasets with limited focus on computational complexity and still less on efficient online inference capabilities. Yet, important real-life applications such as human monitoring [13], [18], driver assistance [3], and autonomous vehicles depend on performing predictions on a continual input stream with low latency and low energy consumption.

Continual Inference Networks (CINs) [9], [8], [7], are a recent family of Deep Neural Networks, which can accelerate a wide range of architectures for time-series processing (e.g., CNNs and Transformers) during online inference, even though source networks may have been trained exclusively for offline processing.

This paper provides comprehensive introduction to CINs (Sec. 2), the guiding principles of their design and implementation via the Continual Inference library (Sec. 4), and summarizes and compares achieved reductions in stepwise computational complexity and memory-usage using the library (Sec. 4).

2 Continual Inference Networks

Originally introduced in [9] and subsequently elaborated in [7], [8], Continual Inference Networks denote a variety of Neural Network, which can operate without redundancy during online inference on a continual input stream, as well as offline during batch inference. Specifically, CINs comply with Def. 1 [7]:

Definition 1 (Continual Inference Network). *A Continual Inference Network is a Deep Neural Network, which*

- *is capable of continual step inference without computational redundancy,*
- *is capable of batch inference corresponding to a non-continual Neural Network,*
- *produces identical outputs for batch inference and step inference given identical receptive fields,*
- *uses one set of trainable parameters for both batch and step inference.*

Many prior networks can be viewed as CINs. This includes networks, which perform their task within a single time-step (e.g., object detection and image recognition models), or which inherently process temporal data step-by-step (e.g., Recurrent Neural Networks such as LSTMs [10] and GRUs [2]). Some network types, however, are limited to batch inference exclusively. These include Convolutional Neural Networks (CNNs) with temporal convolutional components (e.g., 3D CNNs), as well as Transformers with tokens spanning the temporal dimension. While they can in principle be used for online inference, it is an inefficient process, where input steps are assembled to full (spatio-)temporal batches and fed to the network in a sliding window fashion, with many redundant intermediary computations as a result.

While some specialty architectures have been devised to let 3D convolutional network variants make predictions step by step [16], [11], and accordingly also qualify as CINs, these were not weight-compatible with regular 3D CNNs. Recently, *Continual* 3D Convolutions [9] changed this. Through a reformulation of the 3D convolution to compute outputs for each time-step individually rather than for the whole spatio-temporal input at once, well-performing 3D CNNs such as X3D [4], Slow [5], and I3D [1] trained for Trimmed Activity Recognition were re-implemented to execute step by step without any re-training. Likewise, Spatio-temporal Graph Convolutional Networks for Skeleton-based Action Recognition [20], [15], [14], which originally operated only on complete sequences of skeleton graphs, were transformed to perform stepwise inference as well though a continual formulation of their Spatio-temporal Graph Convolution blocks [8]. Temporal Transformer networks had also been restricted to operate on batches until a *Continual* Multi-head Attention (*Co*MHA) [7] was introduced, which is weight-compatible with the original MHA [19], while being able to compute updated outputs for each time step.

With these innovations, many existing DNNs can be converted to operate efficiently during online inference. In general, non-continual networks, which are transformed to continual ones attain reductions in per-step computational complexity in proportion to the temporal receptive field of the network. In some

cases, these savings can amount to multiple orders of magnitude [8]. Still, the implementation of Continual Inference Networks with temporal convolutions and Multi-head Attention in frameworks such as PyTorch [12] requires deep knowledge and practical experience with CINs. With the Continual Inference library described in the next section, we hope to change this.

3 Library Design

3.1 Principles

The fundamental feature of CINs, that networks are flexible and perform well on both online inference and batch inference, is a guiding principle in the design of the Continual Inference library as well: Refactoring of existing implementations in pure PyTorch should be straightforward. In the following, we will adopt the Python import abbreviations `import continual as co` and `from torch import nn`. The library follows Principle 1 to ensure that `co` modules can be used as drop-in replacements for `nn` modules without behavior change:

Principle 1 (Compatibility with PyTorch) `co` *modules with identical names to* `nn` *modules also have:*

1. *identical* `forward`*,*
2. *identical model weights,*
3. *identical or extended constructors,*
4. *identical or extended supporting functions.*

Before proceeding to the enhanced functionality of `co` modules, let us state our assumption to the input format:

Assumption 1 (Order of input dimensions) *Inputs to* `co` *modules use the order* $(B, C, T, S_1, S_2, ...)$ *for multi-step inputs and* $(B, C, S_1, S_2, ...)$ *for single-step inputs, where* B *is the batch size,* C *is the input channel size,* T *is the temporal size, and* S_n *are additional optional dimensions.*

The core difference between Continual Inference Networks and regular networks is their ability to efficiently compute results for each time-step. Besides the regular `forward` function found in `nn` modules, `co` modules add multiple call modes that allow for continual inference with a simple interface:

Principle 2 (Call modes) `co` *modules provide three forward operations:*

1. `forward`*: takes a (spatio-) temporal input and operates identically to the* `forward` *of an* `nn` *module,*
2. `forward_step`*: takes a single time-step as input without a time-dimension and produces an output corresponding to* `forward`*, had it's input been shifted by one time-step, given identical prior inputs.*
3. `forward_steps`*: takes multiple time-steps as input and produces outputs identical to applying* `forward_step` *the number of times corresponding to the temporal size of the input.*

Furthermore, the `__call__` *method of* `co` *modules can be changed to use any of the three by either setting the* `call_mode` *attribute of the module or applying the* `co.call_mode()` *context with a string spelling out the wanted forward type.*

Let us exemplify Principle 2 in practice. Example 1.1 shows how the different forward functions introduced in Principle 2.1 can be used. Principle 2.2 is illustrated in Example 1.2.

```python
import torch
import continual as co

con = co.Conv3d(in_channels=4,
                out_channels=8,
                kernel_size=3)
assert con.delay == 2
assert con.receptive_field == 3

reg = torch.nn.Conv3d(in_channels=4,
                      out_channels=8,
                      kernel_size=3)
# Reuse weights
con.load_state_dict(reg.state_dict())

x = torch.randn((2, 3, 5, 6, 7))  # B,C,T,H,W
y = con.forward(x)
assert torch.equal(y, reg.forward(x))

# Multiple steps
firsts = con.forward_steps(x[:, :, :4])
assert torch.allclose(firsts, y[:, :, : con.delay])

# Single step
last = con.forward_step(x[:, :, 4])
assert torch.allclose(last, y[:, :, con.delay])
```

Example 1.1: Definition and usage of `co.Conv3d` and its forward modes.

```python
net(x)            # Invokes ‘forward‘ by default

net.call_mode = "forward_step"
net(x[:, :, 0])   # Invokes ‘forward_step‘

with co.call_mode("forward_steps"):
    net(x)        # Invokes ‘forward_steps‘

net(x[:, :, 0])   # Invokes ‘forward_step‘ again
```

Example 1.2: Changing the `call_mode` for a continual module `net`.

Continual modules, which use information from multiple time-steps, are inherently stateful. Whenever `forward_step` or `forward_steps` is invoked, intermediary results needed for future step results are optimistically computed and stored. Principle 3 states the rules for state-manipulation and updates.

Principle 3 (State) *Module state is updated according to the following rules:*
- `forward_step` *and* `forward_steps` *use and update state by default.*
- *Step results may be computed without updating internal state by passing* `update_state=False` *to either* `forward_step` *or* `forward_steps`*.*
- `forward` *neither uses nor updates state.*
- *Module state can be wiped by invoking the* `clean_state()` *method.*
- *A module produces non-empty outputs after its has conducted a number of stateful forwards steps corresponding to its* `delay`*.*

Regular `nn` modules predominantly operate on input batches in an offline setting and do not have a built-in concept of delay. `co` modules on the other hand are designed to operate on time-series. Since `co` modules often integrate information over multiple time-steps and online operation is causal by nature, some modules produce the output corresponding to a given input only after observing additional steps. For instance, a `co.Conv1d` module with `kernel_size = 3` produces an output from the third input step as illustrated in Fig. 1. The delay of a module is calculated according to Principle 4:

Principle 4 (Delay) `co` *modules produce step outputs that are delayed by*

$$d = f - p - 1 \tag{1}$$

steps relative to the earliest input step used in the computation, where f is the receptive field and p is the temporal padding.

While padding is used in regular networks to retain the size of feature-maps in consecutive layers, this interpretation of temporal padding does not make sense in the context of an infinite, continual input, as handled by CINs. Instead, we may interpret padding as a reduction in delay. For instance, a `co.Conv1d` module with `kernel_size = 3` and `padding = 2` has a delay of zero, because the padded zeros already "saturated" the state before-hand. This is illustrated in Fig. 2. Considering, that `co` modules expect an infinite and continual input stream, end-padding padding is omitted by default. If an end-padding is required, the library supports its use by either passing manually defined zeros as steps or by setting `pad_end = True` for an invocation of the `forward_steps` function.

Similar to padding, the stride of a `co` module impacts the timing of the outputs. Specifically, stride results in empty outputs every $(s-1)/s$ outputs, as well as larger delays for downstream network modules through increased receptive fields. This is stated in Principles 5 and 6.

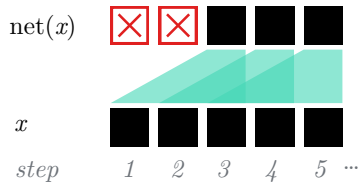


Fig. 1: Sketch of delay and receptive field. Here, the stepwise operation of a `co` module `net` with `receptive_field = 3` is illustrated. ■ are non-zero step-features and ⊠ are empty outputs.

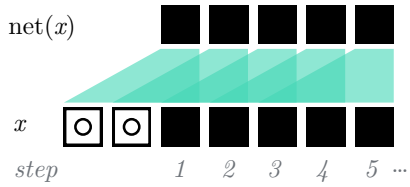


Fig. 2: Sketch of how padding reduces delay. Here, the stepwise operation of a `co` module `net` with `receptive_field = 3`, `padding = 2` is illustrated. ⊡ are padded zeros and ■ are non-zero step-features.

Principle 5 (Stride and prediction rate) *For neural network of N modules with strides $s^{(i)}, i \in \{1..N\}$, the accumulated stride at any given layer is*

$$s_{acc}^{(i)} = s^{(i)} \cdot s_{acc}^{(i-1)} \quad i \in 1..N \tag{2}$$

$$s_{acc}^{(0)} = s^{(0)}. \tag{3}$$

Equivalently, the resulting network stride is

$$s_{NN} = \prod_{i=1}^{N} s^{(i)}, \tag{4}$$

and the network prediction rate is

$$r_{NN} = 1/s_{NN}. \tag{5}$$

Accordingly, the outputs of a `co` *network are empty every $(s_{NN} - 1)/s_{NN}$ steps.*

Principle 6 (Accumulated delay) *The accumulated receptive field of a downstream module i in a network of N modules is given by:*

$$f_{acc}^{(i)} = f^{(i)} + (f_{acc}^{(i-1)} - 1)s^{(i)}, \quad i \in 1..N \tag{6}$$

$$f_{acc}^{(0)} = f^{(0)}. \tag{7}$$

The accumulated delay of layer i in a network is

$$d^{(i)} = f_{acc}^{(i)} - p_{acc}^{(i)} - 1, \tag{8}$$

where the accumulated padding p_{acc} is given by

$$p_{acc}^{(i)} = p^{(i)} \cdot s_{acc}^{(i-1)}, \quad i \in 1..N, \tag{9}$$

$$p_{acc}^{(0)} = p^{(0)}. \tag{10}$$

Fig. 3 illustrates a mixed example, where the first layer of a two-layer network has `padding = 2` and `stride = 2`. Noting layer attributes in consecutive order, and using Equations 2 to 10, the example has the following network attributes:

$$\begin{aligned}
s &= \{2, \quad 1\} \\
p &= \{2, \quad 0\} \\
s_{acc} &= \{2, \quad 2 \cdot 1 = 2\} \\
p_{acc} &= \{2, \quad 2 + 2 \cdot 0 = 2\} \\
f_{acc} &= \{3, \quad 3 + (3-1) \cdot 2 = 7\} \\
d_{acc} &= \{3 - 2 - 1 = 0, \quad 7 - 2 - 1 = 4\} \\
s_{NN} &= s_{acc}^{(1)} = 2 \\
r_{NN} &= 1/s_{NN} = 1/2 \\
d_{NN} &= d_{acc}^{(1)} = 4.
\end{aligned}$$

Before continuing onto the specific modules, we have to discuss a final principle of CINs, namely that of parallel modules.

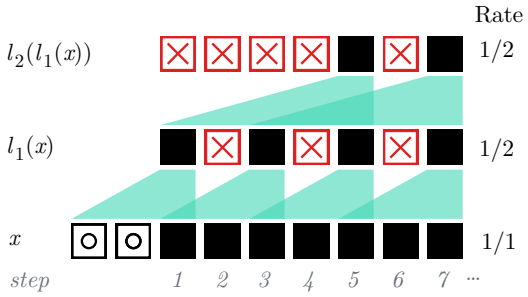


Fig. 3: A mixed example of delay and outputs under padding and stride. Here, we illustrate the stepwise operation of two `co` module layers, l_1 with with `receptive_field = 3`, `padding = 2`, and `stride = 2` and l_2 with `receptive_field = 3`, no padding and `stride = 1`. ⊡ denotes a padded zero, ■ is a non-zero step-feature, and ☒ is an empty output.

Principle 7 (Parallel modules) *Modules can be arranged in parallel to execute on each their separate stream of data under the following rules:*
- *Parallel modules follow the same global clock.*
- *The delay of a collection of parallel modules is the maximum delay of any module in the collection.*
- *If the merger of parallel step values includes an empty value, then the resulting step output of the merger is also empty.*

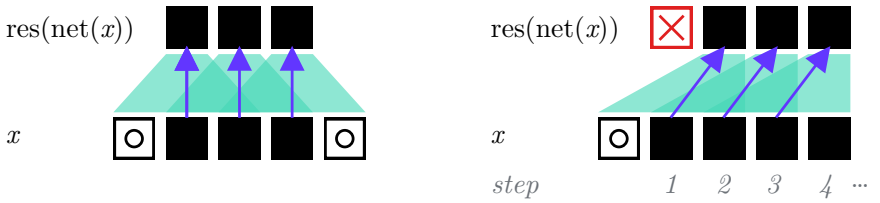
A discussion of residual connections provides a practical example for Principle 7.

Residual connections The residual connection is a simple but crucial tool for avoiding vanishing and exploding gradients; by adding the input of a module to its output, gradients can flow freely through models with hundreds of layers. Without exaggeration, we can state that almost all recent deep architectures at the time of writing use some form of residual connection [6], [19], [20], [5]. Yet, their implementation in Continual Inference Networks may not follow common intuition in all cases. Let us first consider the residual connection during regular `forward` operation as found in a non-continual residual shown in Fig. 4a. Here, the wrapped module will almost always use padding to ensure equal input and output shapes (known as "equal padding"). For a module with receptive field three, we would thus have a padding of one. In this case, the `forward` computation of the residual amounts to adding the input to the output of the convolution. However, the implementation of `forward_step` illustrated in Fig. 4b is different. Since the first output uses information from the second step, the module has a delay of one. Accordingly, the residual connection requires a delay of one as well.

Now consider the same scenario but without padding. This will be quite foreign to many Deep Learning practitioners, and it is not clear how exactly to align residuals. We will use a separate module to shrink the residual by an equivalent amount as the wrapped module. Of the possible alignment choices, a sensible approach is to discard the border values to align the feature maps on *center*. Contrary to other alignment forms, this has the benefit of weight-compatibility between the no-padding case and the case with equal padding described in the former paragraph. The outputs of step 3 in Figures 4b and 5b are equal given the same weights and inputs. However, two issues arise:

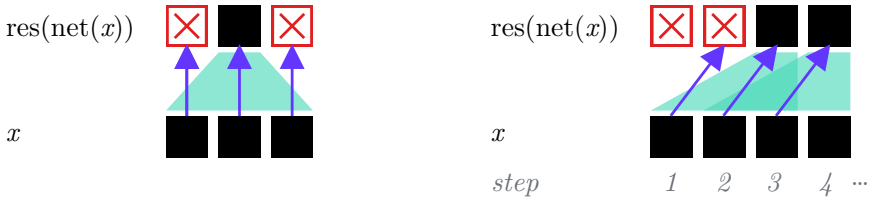
1. Delay mismatch: While the residual connection has a delay of one, the wrapped module has a delay of two.
2. Mix of empty and non-empty results: C.f. the differences in delay, the residual will start producing non-empty outputs before the wrapped module.

Principle 7 helps us navigate this. Despite the internal delay mismatch, the delay of the whole residual module corresponds to the largest delay, in this case two. Consequently, the whole residual module produces outputs from the third step, despite the fact that the delayed input already has non-empty outputs from the second step. Both of these issues can also be avoided if we force residuals to employ the same delay as the wrapped module. This corresponds to a *lagging* alignment. However, using such a strategy breaks weight compatibility between the same residual modules with and without padding.



(a) `forward` (b) `forward_step`

Fig. 4: Residual connections ↑ over a module with receptive field of size ▲ and padding one ("equal padding") ⊡. ☒ are empty outputs.



(a) `forward` (b) `forward_step`

Fig. 5: Centered residual connections ↑ over a module with receptive field of size ▲ and no padding. ☒ are empty outputs.

3.2 Core modules

Designed as an augmentation of PyTorch, the Continual Inference library provides a collection of basic building blocks for composing neural networks. Following Principle 1, we use the same public interfaces as PyTorch , i.e. class constructor, function names and arguments, and attribute names, to ensure that `co` modules can be used as drop-in replacements for `nn` modules. The basic modules can be categorized as follows:

- Convolutions [9]: `co.Conv1d`, `co.Conv2d`, ...
- Pooling: `co.AvgPool1d`, `co.MaxPool1d`, ...
- Linear: `co.Linear`.
- Transformer [7]: `co.TransformerEncoder`, ...
- Shape: `co.Delay`, `co.Reshape`.
- Arithmetic: `co.Lambda`, `co.Add`, ...

Here, the `MultiheadAttention` implementation is a special case, which features two distinct versions of continual operation: 1) "single-output", where only the attention output corresponding to the latest input is produced, and 2) "retrospective", where updates to prior outputs are also produced retrospectively. The details of this are explained in greater detail in [7]. Linear `co` modules follow the `nn` modules closely, but ensure compatibility of dimension c.f. Assumption 1. `co.Delay` adds a specified delay to the input stream. This is handy

Module	Description
`Sequential`	Arrange modules sequentially.
`Broadcast`	Broadcast one stream to multiple parallel streams.
`Parallel`	Apply modules in parallel, each on a separate stream.
`Reduce`	Reduce multiple input streams into one.
`Residual`	Add a residual connection for a wrapped module.
`Conditional`	Conditionally invoke a module (or another) at runtime.

Table 1: Composition modules.

for aligning the delay of multiple streams as required by residual connections (see Sec. 3.1). `co.Lambda` allows a user to pass in functions and functors that are applied stepwise to the inputs. Besides the above list of tailor-made modules, the Continual Inference library has interoperability with most activation functions (`nn.ReLU`, `nn.Softmax`, etc.), normalisation layers (`nn.BatchNorm1d`, `nn.LayerNorm`, etc.), and `nn.DropOut` when used within the composition modules as presented in Sec. 3.3. The full list of compatible modules can be found at `www.github.com/lukashedegaard/continual-inference`.

3.3 Composition modules

In PyTorch, modules are composed by either by using the `nn.Sequential` container or by creating a new class which inherits from `nn.Module` and manually controls data flow within the `forward` function. While the latter is commonly used to handle complex modules in a simple and easily debuggable manner, it is not necessarily the simplest approach for implementing complex Continual Inference Networks. In addition to defining the basic forward flow, a CIN implementation also needs to handle stepwise computations, which require meticulous alignment of delays if Principle 2 is to be kept.

Instead, we expand the container interface of PyTorch to include modules for parallel and conditional processing. While each module is simple in nature, they can be used to compose complex neural network architectures, which retain all the principles in Sec. 3.1 without explicitly needing to consider them. A brief overview and description of each `co` container module is given in Sec. 3.3. To get a practical understanding of these, we will give implementation examples of two common architecture blocks, the residual connection as discussed in Sec. 3.1 and an Inception module [17].

Example 1.3 shows three equivalent implementations of a residual 3D convolution block. `res1` is the verbose version, in which `co.Broadcast` is used to split a single input into two parallel stream, `co.Parallel` specifies that `conv` handles the first stream, while a delay is used on the second. `co.Reduce` merges the streams via an add reduce operation. Due to the commonality of broadcast-apply-reduce operations, the library features a `co.BroadcastReduce` shorthand

```
conv = co.Conv3d(1, 1, kernel_size=3, padding=1)

res1 = co.Sequential(co.Broadcast(2),
                     co.Parallel(conv, co.Delay(1)),
                     co.Reduce("sum"))

res2 = co.BroadcastReduce(conv, co.Delay(1))

res3 = co.Residual(conv)
```

Example 1.3: Equivalent implementations of a residual block.

to specify such composition more succinctly. Even shorter, `co.Residual` can automatically infer the needed delay from the module it wraps. Other reduction functions can be specified in `co.BroadcastReduce` and `co.Residual` using the `reduce` argument, which is `"sum"` by default. The code in Example 1.3 correspond to Fig. 4. The centered residual module in Fig. 5 is easily specified as `co.Residual(conv, residual_shrink=True)` where `conv` has `padding = 0`.

```
def norm_relu(conv):
    return co.Sequential(conv,
                         nn.BatchNorm3d(conv.out_channels),
                         nn.ReLU())

inception_module = co.BroadcastReduce(
    co.Conv3d(192, 64, 1),
    co.Sequential(
        norm_relu(co.Conv3d(192, 96, 1)),
        norm_relu(co.Conv3d(96, 128, 3, padding=1)),
    ),
    co.Sequential(
        norm_relu(co.Conv3d(192, 16, 1)),
        norm_relu(co.Conv3d(16, 32, 5, padding=2))
    ),
    co.Sequential(
        co.MaxPool3d(kernel_size=(1, 3, 3),
                     padding=(0, 1, 1),
                     stride=1),
        norm_relu(co.Conv3d(192, 32, 1)),
    ),
    reduce="concat",
)
```

Example 1.4: *Continual* Inception module using a mix of `co` and `nn` modules.

We can showcase a more advanced application of parallel streams by considering an Inception module [17]. An Inception module broadcasts the input into four streams and applies convolution of varying kernel sizes in parallel before concatenating the channels to produce one output. Without the `co` container modules, it would be complicated to keep track of and align delays of the different branches to create valid `forward`, `forward_step`, and `forward_steps` methods. Using `co.Sequential`, which automatically sums up delays, and `co.BroadcastReduce`, which automatically adds delays to match the branch with highest inherent delay, the implementation becomes simple as shown in Example 1.4.

Model	**Dataset performace** (%)		**Params** (M)	**Max mem.** (MB)	**FLOPs** (G)
	Kinetics-400 (Acc.)				
X3D-L	69.3		6.2	240.7	19.17
*Co*X3D-L$_{64}$	71.6 (+2.3)		6.2	184.4 (75%)	1.25 (↓ 15.34×)
X3D-M	67.2		3.8	126.3	4.97
*Co*X3D-M$_{64}$	71.0 (+3.8)		3.8	69.0 (55%)	0.33 (↓ 15.06×)
X3D-S	64.7		3.8	61.3	2.06
*Co*X3D-S$_{64}$	67.3 (+2.6)		3.8	42.0 (69%)	0.17 (↓ 12.12×)
Slow-8×8	67.4		32.5	266.0	54.87
*Co*Slow$_{64}$	73.1 (+5.7)		32.5	176.4 (66%)	6.90 (↓ 7.95×)
I3D	64.0		28.0	191.6	28.61
*Co*I3D$_8$	59.6 (−4.4)		28.0	235.9 (123%)	5.68 (↓ 5.04×)
	THUMOS14 (mAP)	**TVSeries** (mcAP)			
OadTR-b2	64.2	89.0	15.9	67.6	1.08
*Co*OadTR-b2	64.4 (+0.2)	88.2 (−0.8)	15.9	71.7 (106%)	0.41 (↓ 2.61×)
OadTR-b1	64.4	89.1	9.6	43.3	0.67
*Co*OadTR-b1	64.5 (+0.1)	88.0 (−1.1)	9.6	45.1 (104%)	0.01 (↓ 63.49×)
	NTU RGB+D 60 (Acc.)				
	X-Sub	*X-View*			
ST-GCN	86.0	93.4	3.1	45.3	16.73
*Co*ST-GCN*	86.3 (+0.3)	93.8 (+0.4)	3.1	36.1 (80%)	0.16 (↓ 107.7×)
AGCN	86.4	94.3	3.5	48.4	18.69
*Co*AGCN*	84.1 (−2.3)	92.6 (−1.7)	3.5	37.4 (77%)	0.17 (↓ 108.8×)
S-TR	86.8	93.8	3.1	74.2	16.14
*Co*S-TR*	86.3 (−0.3)	92.4 (−1.4)	3.1	36.1 (49%)	0.15 (↓ 107.6×)

Table 2: Dataset performance, parameter count, maximum allocated memory (Max mem.), and floating-point operations (FLOPs) of continual and non-continual models on video and spatio-temporal graph classification datasets. Subscript$_{xx}$ denotes expanded temporal average pooling, b1 and b2 denote one and two block transformer decoders, and superscript* indicates architectures where network stride was reduced to one. Parentheses show the improvement / deterioration of the continual model relative to the corresponding non-continual model. The noted metrics were originally presented in [9], [7], [8].

4 Performance comparisons

Using the basic `co` modules and composition building blocks, continual versions of advanced neural networks have been implemented in multiple recent works with manyfold speedups and significant reductions in memory consumption during online inference [9], [7], [8]. Specifically, the 3D-CNNs *Co*X3D, *Co*I3D, and *Co*Slow for video-based Human Activity Recognition were proposed in [9]; the Transformer *Co*OadTR for Online Action Detection in [7]; and Spatio-temporal Graph Convolutional Networks *Co*ST-GCN, *Co*AGCN, and *Co*S-TR for Skeleton-based Action Recognition in [8]. While direct conversion from regular to continual versions of the above noted architectures works well in accelerating inference in itself, further improvements can be achieved by exploiting some core characteristics of CINs: in [9], accuracy was improved by increasing model receptive fields through expansions of temporal global average pooling to 64 steps, and in [8], the stride of temporal convolutions was reduced to one to increase prediction rates. Tab. 2 presents a summary of benchmark performance, computational complexity, and maximum allocated memory on GPU for each of these networks alongside with their non-continual counterparts [9], [7], [8].

5 Conclusion

We presented Continual Inference, an easy-to-use library for implementing Continual Inference Networks in Python. Following interfaces closely, the components provided in the library are backwards-compatible drop-in replacements for PyTorch modules, which add the capability of redundancy-free online inference without the need for intimate knowledge of CINs nor their meticulous low-level implementation. Having shown the vast computational advantages of CINs over regular neural networks in multiple settings of video and spatio-temporal graph classification, we hope that this library will contribute to the adoption of CINs and the advancement of use-cases requiring low-latency online inference under recourse constraints in general.

Acknowledgement

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

References

1. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4724–4733 (2017)
2. Cho, K., van Merriënboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: Encoder–decoder approaches. In: Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation. pp. 103–111 (2014)
3. Enkelmann, W.: Video-based driver assistance–from basic functions to applications. International Journal of Computer Vision (IJCV) **45**(3), 201–221 (2001)
4. Feichtenhofer, C.: X3D: Expanding architectures for efficient video recognition. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
5. Feichtenhofer, C., Fan, H., Malik, J., He, K.: Slowfast networks for video recognition. In: IEEE/CVF International Conference on Computer Vision (ICCV). pp. 6201–6210 (2019)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016)
7. Hedegaard, L., Bakhtiarnia, A., Iosifidis, A.: Continual Transformers: Redundancy-Free Attention for Online Inference. preprint, arXiv:2201.06268 (2022)
8. Hedegaard, L., Heidari, N., Iosifidis, A.: Online skeleton-based action recognition with continual spatio-temporal graph convolutional networks. preprint, arXiv: 2203.11009 (2022)
9. Hedegaard, L., Iosifidis, A.: Continual 3d convolutional neural networks for real-time processing of videos. In: European Conference on Computer Vision (ECCV). pp. 1–12 (2022)
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**, 1735–80 (12 1997)
11. Köpüklü, O., Hörmann, S., Herzog, F., Cevikalp, H., Rigoll, G.: Dissected 3D CNNs: Temporal skip connections for efficient online video processing. preprint, arXiv:2009.14639 (2020)
12. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019)
13. Pigou, L., van den Oord, A., Dieleman, S., Van Herreweghe, M., Dambre, J.: Beyond temporal pooling: Recurrence and temporal convolutions for gesture recognition in video. International Journal of Computer Vision (IJCV) **126**(2), 430–439 (2018)
14. Plizzari, C., Cannici, M., Matteucci, M.: Skeleton-based action recognition via spatial and temporal transformer networks. Computer Vision and Image Understanding **208**, 103219 (2021)
15. Shi, L., Zhang, Y., Cheng, J., Lu, H.: Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 12026–12035 (2019)

16. Singh, G., Cuzzolin, F.: Recurrent convolutions for causal 3d cnns. In: IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). pp. 1456–1465 (2019)
17. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1–9 (2015)
18. Tavakolian, M., Hadid, A.: A spatiotemporal convolutional neural network for automatic pain intensity estimation from facial dynamics. International Journal of Computer Vision (IJCV) **127**(10), 1413–1425 (2019)
19. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 30, pp. 5998–6008 (2017)
20. Yan, S., Xiong, Y., Lin, D.: Spatial temporal graph convolutional networks for skeleton-based action recognition. In: AAAI Conference on Artificial Intelligence. pp. 7444–7452 (2018)

8.3 Continual Transformers: Redundancy-Free Attention for Online Inference

The appended paper follows.

Continual Transformers: Redundancy-Free Attention for Online Inference

Lukas Hedegaard **Arian Bakhtiarnia** **Alexandros Iosifidis**
DIGIT, Department of Electrical and Computer Engineering, Aarhus University, Denmark
{lhm, arianbakh, ai}@ece.au.dk

Abstract

Transformers in their common form are inherently limited to operate on whole token sequences rather than on one token at a time. Consequently, their use during online inference on time-series data entails considerable redundancy due to the overlap in successive token sequences. In this work, we propose novel formulations of the Scaled Dot-Product Attention, which enable Transformers to perform efficient online token-by-token inference on a continual input stream. Importantly, our modifications are purely to the order of computations, while the outputs and learned weights are identical to those of the original Transformer Encoder. We validate our *Continual* Transformer Encoder with experiments on the THUMOS14, TVSeries and GTZAN datasets with remarkable results: Our *Continual* one- and two-block architectures reduce the floating point operations per prediction by up to 63× and 2.6×, respectively, while retaining predictive performance.

1 Introduction

Many real-life usage scenarios such as the perception in self-driving cars and live monitoring of critical resources process a continual stream of inputs and require near-instantaneous predictions per time-step. This stands in contrast to what many common benchmarks for deep learning evaluate, namely the operation on distinct batches of data with no inter-batch relationships. Consequently, a plethora of methods have been developed [1, 2, 3, 4, 5, 6, 7, 8], which focus on batch-wise processing, but fail to optimise for online operation, where new information (e.g., a video frame / token) arrives at each step from a continual input stream, and future information is not available at the current time-step. We need a class of networks, which operate efficiently on *both batches of data and on continual streams*.

Accordingly, we propose a reformulation of the Transformer Encoder as a Continual Inference Network (CIN, Section 2.1) which accelerates the stream processing on time-series data, while retaining weight-compatibility. Specifically, we derive two variants of Continual Scaled Dot-Product Attention (SDA) for the cases where prior output tokes *should* and *should not* be updated after observing a new input token. Notably, our attention formulations reduce the per-step cost of SDA [6] from time complexity $\mathcal{O}(n^2d)$ to $\mathcal{O}(nd)$ and memory complexity $\mathcal{O}(n^2)$ to $\mathcal{O}(nd)$ and are readily embedded into Continual Multi-Head Attention (MHA) and Continual Transformer Encoder blocks. Finally, we propose the use of Recycling Positional Encoding to accommodate progressive caching of partial attention results for continual data streams.

Due to the interdependence of SDA outputs, Continual Transformers are most efficient for shallow architectures. Shallow Transformers have many applications such as augmentations of CNNs [9], light-weight Natural Language Processing [10], fusion operations in multi-modal (e.g. audio-visual)

Preprint. Under review.

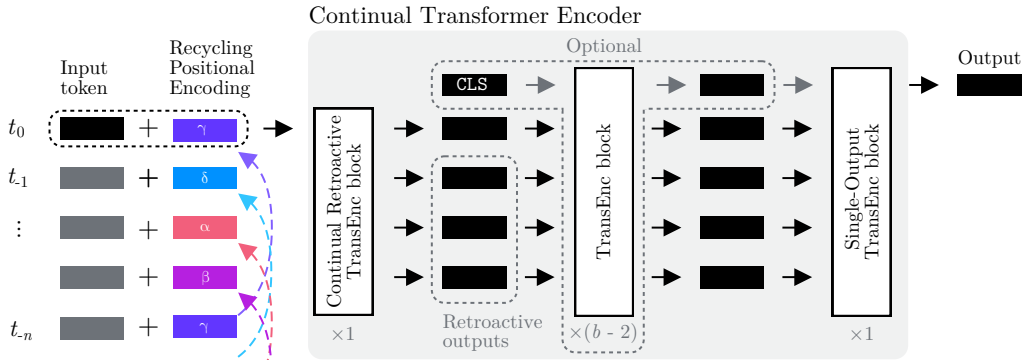


Figure 1: **Multi-block Continual Transformer Encoder with Recycling Positional Encoding.** For $b > 2$ blocks, regular Transformer Encoder blocks can be added between an initial Continual Retroactive block and a final Single-Output block. A class-token may be used after the initial block.

settings [11] and early exit branches in multi-exit architectures [12, 8]. In our experiments[1], we validate their exceptional efficiency improvements on common benchmarks in Online Action Detection [13] and Online Audio Classification [14].

2 Related Work

2.1 Continual Inference Networks

Definition (Continual Inference Network). *A Deep Neural Network, which*

- *is capable of continual step inference without computational redundancy,*
- *is capable of batch inference corresponding to a non-continual Neural Network,*
- *produces identical outputs for batch- and step inference given identical receptive fields,*
- *uses one set of trainable parameters for both batch and step inference.*

These requirements ensure that a Neural Network has broad applicability for both (offline) batch-wise inference (i.e., most research benchmarks) and online stream processing. While non-CINs can operate on streams of data by caching prior steps in a first-in first-out (FIFO) queue and aggregating them to a full (spatio-)temporal input, which is processed similarly to an offline batch, this entails computational redundancy in proportion with the sequence length. CINs perform step-wise inference without such caching and repeat computation. Uni-directional Recurrent Neural Networks are an example of Continual Inference Networks. Their default mode of operation is by time-step and they are easily applied to spatio-temporal batches of data by concatenation of the step-wise outputs. Recently, a modification to the spatio-temporal 3D convolution was proposed [15], which enables existing 3D CNNs to operate efficiently during continual inference. A similar principle was used to enhance Spatio-temporal Graph Convolutions as well [16]. In Section 3, we derive a CIN formulation for Transformer Encoders.

2.2 Transformer architectures

Initially proposed for sequence-to-sequence modelling in Natural Language Processing, the Transformer [6] has become a canonical building block in many applications of Deep Learning, including Computer Vision [17, 7, 18, 19] and Audio Classification [20]. Their success can be partly attributed to reduced inductive bias compared with CNNs and RNNs, which allows better adaptations when sufficiently large datasets are available; the Scaled Dot-Product Attention (SDA) maps a set of input tokens to a set of outputs without inherent preconceptions. However, this many-to-many attention exhibits quadratic growth in time and space complexity with the token count n in the set.

A great deal of research has sought to improve the efficiency of Transformers [21]. Block-wise or Chunking methods such as Image Transformer [22] and Vision Transformer [17] group up entities of a

[1] Source is code provided in supplementary material. Link will be made available upon acceptance.

local receptive field into a single block, reducing the $\mathcal{O}(n^2)$ complexity to $\mathcal{O}(n_b^2)$, where $n_b < n$ is the number of blocks. Techniques such as sliding windows, dilation and pooling can be used to achieve a similar effect [23]. The Reformer [24] reduces the complexity to $\mathcal{O}(n \log n)$ by learning groupings in a data-driven manner via Locality-Sensitive Hashing (LSH). A different paradigm aims to derive approximations of the self-attention matrix. Methods such as Linformer [25], Nyströmformer [26] and Performer [27] reduce the complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$. Unlike these efforts, our approach produces the *exact* same computational outputs for temporal sequences as the original Multi-Head Attention.

3 Continual Transformers

The Scaled Dot-Product Attention (SDA) is central to the Transformer. Consider the case, where the query, key and value inputs to the SDA constitute a continual stream of d-dimensional tokens and we wish to compute the outputs for each step immediately considering $n-1$ prior tokens. Let us examine three SDA implementations and derive the complexity of each.

3.1 Regular Scaled Dot-Product Attention

Denoting query, key, and value sequence matrices by $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n \times d}$, the regular Scaled Dot-Product Attention first defined by Vaswani et al. [6] can be written as:

$$\text{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{D}^{-1}\mathbf{A}\mathbf{V} \qquad \mathbf{A} = \exp\left(\mathbf{Q}\mathbf{K}^\top / \sqrt{d}\right) \qquad \mathbf{D} = \text{diag}\left(\mathbf{A}\mathbb{1}_n^\top\right), \tag{1}$$

where $\mathbf{A}, \mathbf{D} \in \mathbb{R}^{n \times n}$ and $\mathbb{1}_n$ is a row-vector of n ones. In each time-step, we can update $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ by discarding their oldest token and prepending a new one in a FIFO manner. This is a common implementation for step-wise inference, e.g. found in the FAIRSEQ library [28].

Each time-step results in $2n^2d + 2nd$ multiplications, $2n^2d - nd - n$ additions, and n^2 exponentiations as accounted for in Appendix A.1, which amounts to a time complexity of $\mathcal{O}(n^2d)$ and a $\mathcal{O}(n^2)$ memory complexity originating from the transient feature-map $\mathbf{A}$. Furthermore, a constant-sized cache of size $3(n-1)d$ is needed to store the $n-1$ latest tokens in $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$. We could avoid considerable redundancy by caching $\mathbf{Q}\mathbf{K}^\top$ directly. However, this comes with a memory penalty of $(n-1)^2$. Fortunately, another computational scheme can be devised.

3.2 Continual Retroactive Scaled Dot-Product Attention

We can compute $\mathbf{D}^{-1}\mathbf{A}\mathbf{V}$ in a step-wise manner using the latest query, key, and value steps, $\mathbf{q}_{\text{new}}, \mathbf{k}_{\text{new}}, \mathbf{v}_{\text{new}} \in \mathbb{R}^{1 \times d}$, alongside appropriately cached partial results. The softmax normalisation with $\mathbf{D}^{-1}$ can be efficiently implemented via column-aligned element-wise multiplications (denoted by $\odot$ hereafter) of a column-vector $\mathbf{d} = \mathbf{A}\mathbb{1}_n^\top$. If we cache the $n-1$ values for the prior step tokens, i.e. $\mathbf{d}_{\text{mem}} = \mathbf{A}_{\text{prev}}^{(-n+1:-1)}\mathbb{1}_{n-1}^\top$, alongside $\mathbf{Q}$ and $\mathbf{K}$, we can define the step update as:

$$\mathbf{d}^{(-n+1:-1)} = \mathbf{d}_{\text{mem}}^{(-n+2:0)} - \exp\left(\mathbf{Q}_{\text{mem}}\mathbf{k}_{\text{old}}^\top\right) + \exp\left(\mathbf{Q}_{\text{mem}}\mathbf{k}_{\text{new}}^\top\right) \tag{2}$$

$$\mathbf{d}^{(0)} = \exp\left(\frac{\mathbf{q}_{\text{new}}}{\sqrt{d}}\left(\mathbf{K}_{\text{mem}} \parallel \mathbf{k}_{\text{new}}\right)^\top\right)\mathbb{1}_n^\top, \tag{3}$$

where $\mathbf{Q}_{\text{mem}}$ ($\mathbf{K}_{\text{mem}}$) are the $n-1$ prior query (key) tokens, $\mathbf{k}_{\text{old}}$ is the key from n steps ago, and $\parallel$ denotes concatenation of matrices along the first dimension. Negative indices indicate prior time-steps. An update for $\mathbf{A}\mathbf{V}$ can likewise be defined as a function of the $n-1$ prior values $\mathbf{A}\mathbf{V}_{\text{mem}}$:

$$\mathbf{A}\mathbf{V}^{(-n+1:-1)} = \mathbf{A}\mathbf{V}_{\text{mem}}^{(-n+2:0)} - \exp\left(\mathbf{Q}_{\text{mem}}\mathbf{k}_{\text{old}}^\top\right)\mathbf{v}_{\text{old}} + \exp\left(\mathbf{Q}_{\text{mem}}\mathbf{k}_{\text{new}}^\top\right)\mathbf{v}_{\text{new}} \tag{4}$$

$$\mathbf{A}\mathbf{V}^{(0)} = \exp\left(\frac{\mathbf{q}_{\text{new}}}{\sqrt{d}}\left(\mathbf{K}_{\text{mem}} \parallel \mathbf{k}_{\text{new}}\right)^\top\right)\left(\mathbf{V}_{\text{mem}} \parallel \mathbf{v}_{\text{new}}\right). \tag{5}$$

Finally, we compute the Continual Retroactive Attention output in the usual manner:

$$CoRe\text{Att}(\mathbf{q}_{\text{new}}, \mathbf{k}_{\text{new}}, \mathbf{v}_{\text{new}}) = \mathbf{d}^{-1} \odot \mathbf{A}\mathbf{V}. \tag{6}$$

An visual depiction of these update steps is provided in Appendix A.2. A time-step can now be computed with $7nd + 2n - 3d$ multiplications, $6nd + 3n - 6d - 3$ additions, and $3n - 2$ exponentials. This time complexity of $\mathcal{O}(nd)$ per step and a $\mathcal{O}(nd)$ memory complexity is a significant improvement over the prior $\mathcal{O}(n^2d)$ and $\mathcal{O}(n^2)$ complexities in Section 3.1.

3.3 Continual Single-Output Scaled Dot-Product Attention

Both the Regular and Continual Retroactive Dot-Product Attentions produce attention outputs for the current step, as well as $n-1$ retroactively updated steps. In cases where retroactive updates are not needed, we can simplify the computation greatly via a Continual Single-Output Dot-Product Attention ($CoSi$Att). In essence, the regular SDA is reused, but prior values of $\mathbf{k}$ and $\mathbf{v}$ are cached between steps (as in [28]), and only the attention corresponding to a single query token $\mathbf{q}$ is computed:

$$CoSi\text{Att}(\mathbf{q}, \mathbf{k}_{\text{new}}, \mathbf{v}_{\text{new}}) = \mathbf{a}\left(\mathbf{V}_{\text{mem}} \parallel \mathbf{v}_{\text{new}}\right) / \mathbf{a}\mathbb{1}_n^\top, \qquad \mathbf{a} = \exp\left(\frac{\mathbf{q}}{\sqrt{d}}\left(\mathbf{K}_{\text{mem}} \parallel \mathbf{k}_{\text{new}}\right)^\top\right). \tag{7}$$

A step output is computed with $2nd + 2d$ multiplications, $2nd - d - 1$ additions, and n exponentials. The time- and memory complexities remain $\mathcal{O}(nd)$ per step. Using the (leading) query $\mathbf{q}_{\text{new}}$ as input, the attention is purely causal. Alternatively, prior (lagging) query vectors could be cached and used as query input, though this would introduce a network delay.

3.4 Comparison of Scaled Dot-Product Attentions

Assuming $n-1$ prior $\mathbf{q}$, $\mathbf{k}$ and $\mathbf{v}$ steps have been calculated by the Continual SDA modules, and that $\mathbf{Q} = (\mathbf{Q}_{\text{mem}} \parallel \mathbf{q}_{\text{new}})$, $\mathbf{K} = (\mathbf{K}_{\text{mem}} \parallel \mathbf{k}_{\text{new}})$, and $\mathbf{V} = (\mathbf{V}_{\text{mem}} \parallel \mathbf{v}_{\text{new}})$, we have the correspondence:

$$\text{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V})^{(t)} = CoRe\text{Att}(\mathbf{q}_{\text{new}}, \mathbf{k}_{\text{new}}, \mathbf{v}_{\text{new}})^{(t)} = CoSi\text{Att}(\mathbf{q}_t, \mathbf{k}_{\text{new}}, \mathbf{v}_{\text{new}}) \tag{8}$$

Here, $\mathbf{q}_t$ is the t^{th} row of $\mathbf{Q}$, i.e. $\mathbf{Q}^{(t)}$. During stream processing, the complexity of the Continual Retroactive SDA scales significantly more favourably that the regular SDA. For example, the floating point operations (FLOPs) are reduced by $31\times$ when $n = d = 100$ and $308\times$ when $n = d = 1000$. If retroactive output updates are not needed, the Continual Single-Output SDA reduces FLOPs by respectively $100\times$ and $1000\times$. The scaling properties are detailed in Appendix A.1.

3.5 Continual Multi-Head Attention

Continual Scaled Dot-Product Attentions can replace regular SDA's directly in a Multi-Head Attention (MHA). Given a new query, key, and value, $\mathbf{q}, \mathbf{k}, \mathbf{v}$, the Continual MHA is defined as

$$Co\text{MHA}(\mathbf{q}, \mathbf{k}, \mathbf{v}) = \left(\Big\|_{i=0}^{h-1} Co\text{Att}(\mathbf{q}\mathbf{W}_Q^i, \mathbf{k}\mathbf{W}_K^i, \mathbf{v}\mathbf{W}_V^i)\right)\mathbf{W}_O, \tag{9}$$

where $\parallel$ denotes concatenation of h heads and $\mathbf{W}_Q^i, \mathbf{W}_K^i \in \mathbb{R}^{d \times d_K/h}$, $\mathbf{W}_V^i \in \mathbb{R}^{d \times d_V/h}$, and $\mathbf{W}_O \in \mathbb{R}^{d_V \times d_O}$ are projection matrices of head i. CoAtt can be either $CoRe$Att or $CoSi$Att.

3.6 Continual Transformer Encoder

A Continual MHA block can be integrated in a Continual Transformer Encoder block as follows:

$$\mathbf{z} = \text{LayerNorm}\left(\mathbf{y} + \text{FF}(\mathbf{y})\right), \qquad \mathbf{y} = \text{LayerNorm}\left(\text{Sel}(\mathbf{x}) + Co\text{MHA}(\mathbf{x}, \mathbf{x}, \mathbf{x})\right), \tag{10}$$

where $\mathbf{x}$ corresponds to the newest step input and $\text{Sel}(\cdot)$ selects a single (last) token of $\mathbf{x}$ if *CoSi*MHA is used, or selects all tokens otherwise. $\text{FF}(\cdot)$ is a two-layer feed-forward network with weights $\mathbf{W}_1, \mathbf{W}_2$, biases w_1, w_2, and a activation function $\sigma(\cdot)$, i.e. $\text{FF}(\mathbf{x}) = \sigma(\mathbf{x}\mathbf{W}_1 + w_1)\mathbf{W}_2 + w_2$. Aside from the residual selection, this is identical to common Transformer Encoder implementations [6, 17].

3.7 Recycling Positional Encoding

Since a Transformer Encoder does not provide positional bias, it is common to augment a token $\mathbf{x}_i$ with a positional encoding $\mathbf{p}$, i.e. $\tilde{\mathbf{x}}_i = \mathbf{x}_i \circ \mathbf{p}_i$, where $\circ$ could be addition or concatenation. In regular Transformers, the index i denotes a position in a sequence rather than a position in time. However, this static positional assignment is problematic in the context of continual inference; the last token at time $t = 0$ will be the next-to-last token at time $t = 1$, and thus in need of a different positional encoding than in the prior time-step. Instead, CINs require dynamic positions. There have been multiple prior

works [29, 30, 31] which create relative encodings by augmenting the SDA with positional offsets between query and keys. While such a modification to the continual attentions is possible, it hinders compatibility with the regular SDA. Instead, we use a *Recycling Positional Encoding* (RPE), which lets the positional encoding follow each token in time and recycles old encodings:

$$\tilde{\mathbf{x}_t} = \mathbf{x}_t + \mathbf{p}_{\tau_t}, \qquad \tau_t = (\tau_{t-1} + 1) \bmod T, \tag{11}$$

where T is the number of encodings. While RPE does not specify relative encodings explicitly, the absolute positional interpretation of each token changes dynamically when a new token arrives. In practice, the network learns relative, shift-invariant positional information by training with random τ_0 for each batch. Random shifts during training were recently explored in [32, 33, 34] as well. RPE can use either learned or predefined encodings. In the latter case, Cyclic Positional Encoding [35], a sinusoidal encoding inspired by Gray code, is a good fit. If we reuse the encoding immediately after an old token has "slided out", i.e. $T = n$, a token will have the same positional encoding relative to another whether it was m steps older or $n - m$ steps newer. The positional ambiguity can be avoided by extending the number of positional tokens to $T = 2n - 1$. We explore both options in Section 4.1.2.

3.8 Architectural considerations

Block count In Section 3.4, we observed an exact correspondence between the results of the continual and regular SDA layers. However, the correspondence does not necessarily hold for stacked layers. Consider the result of stacking two Continual Single-Output Transformer Encoder blocks. While the first block outputs a step t that is identical to that in a corresponding regular block, the second block would have been initialised with prior step-wise inputs, which were the result of prior input windows instead of the current one; the correspondence would not hold. Though it is not convertible to/from a regular Transformer Encoder, the stacked Single-Output Transformer Encoder architecture has the merit of efficiency. This was exploited in Transformer-XL [31]. Given a single step input, the Continual Retroactive Transformer Encoder block produces output tokens corresponding to the entire observed sequence inside the window. Due to this one-to-many input-output mapping, it is not possible to stack multiple such layers. Nevertheless, it can be used in conjunction with a Continual Single-Output Transformer Encoder with optional regular Transformer Encoder blocks in between as illustrated in Fig. 1. The Regular Transformer Encoder blocks in the resulting architecture have a significantly larger computational complexity than the Continual Retroactive and Single-Output blocks. Consequently, we recommend that Continual Transformer Encoders be used primarily in lightweight architectures with one or two blocks unless compatibility with non-continual Transformers is not required and only Single-output blocks are used.

Class token It is common to add a class token as input to transformers [36, 17], which accumulates information from other tokens prior to classification. However, it cannot be used naïvely with CINs, as this would effectively double the number of input steps. In practice, it can be employed in Continual multi-block Transformer Encoders as input to the second block (see Fig. 1), but this placement limits class token interaction with downstream layers. It can also be used for one-block Transformer Encoders if the value token is omitted as input.

Peak memory reduction trick The FLOPs for $\text{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ are exactly n times those of $CoSi\text{Att}(\mathbf{q}, \mathbf{k}_{\text{new}}, \mathbf{v}_{\text{new}})$. Comparing their memory complexity, the regular SDA is $\mathcal{O}(n^2)$, while the Single-output SDA is $\mathcal{O}(nd)$. In practical applications where system memory is limited, we may thus reduce the maximum memory requirement of the computational device at inference by up to d/n (assuming $n \gg d$) by computing each row of the attention individually. However, this may reduce throughput due to reduced parallelism.

4 Experiments

We provide case studies within two perception disciplines, namely Online Action Detection (Section 4.1) and Audio Classification (Section 4.2). In each case, we will start with a brief overview of the field, followed by experiments and results.

4.1 Online Action Detection

Online Action Detection (OAD) [37] entails the per-frame classification of human actions in a video stream as they happen without the ability to change prior predictions nor to use future information. This is fundamentally more restrictive than Temporal Action Localisation, where the whole video clip is processed before start and end frames of an action are determined [38, 39, 40, 41].

The dominant design in OAD works at the time of writing is to employ a two-stream Convolutional Neural Network as backbone for frame-wise feature extraction with RGB images as inputs in one stream and Optical Flow fields in the other [42, 43, 44, 18, 45][2]. On top of these, OAD methods encode temporal information and perform predictions per time-step, e.g. by means of RNNs [42, 43, 44] or Transformers [18, 45]. Alongside the action detection for the current frame, an action anticipation task may be learned in parallel by means of decoder structures, as this has been found to improve the primary OAD task.

Unlike RNNs, an output update for the regular SDA in a Transformer block cannot be naïvely computed for a single step by feeding successive video frames. Instead, prior step features must be cached, re-loaded and re-processed by the Transformer in each step in correspondence with a predefined window-size of prior steps. As laid out in Section 3.8, Continual Transformers are especially efficient when either one or two Continual Transformer Encoder blocks are used. Accordingly, we start our experiments with a set ablation studies to simplify a recent transformer-based architecture, the OadTR [18]. We further investigate the impact of ablating class token position and the use of Recycling Positional Encoding and compare different RPE schemes for Continual Transformers. Finally, we evaluate our configurations on two widely used OAD datasets, THUMOS14 [13] and TVSeries [37].

4.1.1 Experimental setup

The THUMOS14 dataset [13] for OAD has 200 and 213 validation and test videos, respectively, with frame-level class annotations across 20 classes. As in prior OAD works, the model is trained on the validation set and evaluated on the test set. Similar to [18] we use pre-extracted features from a two-stream Temporal Segment Network (TSN) [46] trained on ActivityNet v1.3 [47] or Kinetics-400 [2].

For TVSeries [37], the network learns on the train and validations sets (20 videos) and evaluates on the test set (7 videos) as in [18]. RGB and Optical Flow features were extraced using an MMAction2 [48] pipeline with ActivityNet v1.3 [47] and Kinetics-400 [2] pretrained TSN ResNet-50 [49] backbones. This is similar to the feature extraction process used by LSTR [45].

Following Wang et al. [18], we use a batch size of 128, sequence length 64, initial learning rate 10^{-4} with a factor ten reduction each epoch, alongside weight decay 10^{-4}, and dropout with probability 0.1. We report results using two epochs of training on a Nvidia RTX2080 Ti GPU. We track mean Average Precision (mAP) for THUMOS14 and calibrated mean Average Precision (cmAP) [37] for TVSeries, alongside FLOPs per prediction and parameters of the OAD module (feature extraction excluded). We report the mean $\pm$ standard deviation over five runs.

4.1.2 Ablation studies

Removing the Decoder As a first step to make an efficient Continual OadTR, we remove the decoder blocks used for action anticipation, which has a large impact on computational efficiency and the ease of transformation to a Continual Inference Network. The first two lines of Table 1a present the results of the removal. Contrary to the observations of Wang et al., we did not find any drop in accuracy when excluding the decoder. We do, however, gain a large reduction in FLOPs and model size; they were reduced to 58% and 30%, respectively. Given these computational improvements, we exclude the decoder in subsequent experiments.

(Re)moving the Class token Class tokens should not be input naively to the first Transformer Encoder layer of a CIN (see Section 3.8). Accordingly, we ablate its use and position. In cases where

[2]The feature extraction commonly used in Online Action Detection (OAD) works is in itself quite computationally costly. We consider the optimisation of the backbone as orthogonal future work and will follow the same feature extraction procedure as other OAD works at this time.

Table 1: **Ablation experiments** on THUMOS14 with TSN-Anet features. **Best** metrics are highlighted. '-' indicates that a particular feature was not used.

(a) **Class token** variations with OadTR. `CLS` pos. is the encoder block into which `CLS` is input.

Enc. blocks	Dec.	CLS pos.	mAP (%)	FLOPs (M)	Params (M)
3	✓	1	57.0±0.5	2445.6	74.7
3	-	1	57.0±0.4	1430.6	22.2
3	-	2	**57.3±0.7**	1423.5	22.2
3	-	3	56.7±0.6	1417.2	22.2
3	-	-	56.8±0.3	**1410.9**	22.2
2	-	1	56.5±0.5	1020.7	15.9
2	-	2	**56.7±0.3**	1014.5	15.9
2	-	-	56.6±0.3	**1008.1**	15.9
1	-	1	**57.1±0.6**	611.7	9.6
1	-	-	56.3±0.2	**605.5**	9.6

(b) **Positional encodings** variations for *Co*OadTr.

Enc. blocks	Re-cycling	Learn	Pos. tokens	mAP (%)	FLOPs (M)	Params (K)
2	-	✓	n	45.3±0.9	410.9	15832
2	✓	✓	n	56.4±0.3	410.9	15832
2	✓	✓	$2n-1$	56.0±0.5	410.9	15897
2	✓	-	n	55.8±1.0	410.9	**15767**
2	✓	-	$2n-1$	**56.8±0.4**	410.9	**15767**
1	-	✓	n	44.0±0.8	9.6	9535
1	✓	✓	n	55.6±0.3	9.6	9535
1	✓	✓	$2n-1$	55.6±0.3	9.6	9599
1	✓	-	n	54.4±1.8	9.6	**9469**
1	✓	-	$2n-1$	**56.1±0.7**	9.6	**9469**

it is removed, we predict on the token corresponding to the last input token. The results of varying `CLS` pos are noted in Table 1a. For the one-block architecture, the removal came with noticeable drop in mAP, while the two-block architecture saw small improvements when removing or introducing the class token later. For the three block model, the use of class tokens in block two achieved the highest mAP. Though it is commonly accepted, that class tokens should be introduced alongside other inputs in the first block, our results indicate that they can accumulate sufficient information with only one or two blocks, and that later stage introduction may work better in some applications. In general, the achieved mAP when varying `CLS` pos. and number of blocks are very similar to one another, while (re)moving the class token and reducing the block size both reduce computational complexity. This encourages the use of shallow Transformer Encoders over deeper ones as well as the removal of class tokens, as we do in the following experiments.

Positional Encodings We can transfer parameters from the simplified one- and two block OadTR to the corresponding Continual architecture, *Co*OadTR. Here, the one block version (*Co*OadTR-b1) uses *CoSi*MHA, and the two block model (*Co*OadTR-b2) uses *CoRe*MHA in the first block and Single-output MHA in the second. However, a regular positional encoding is not suited for continual inference (see Section 3.7). We evaluate the performance of using non-continual encodings for continual inference, as well as of our proposed Recycling Positional Encodings with fixed or learned parameters. In addition, we explore the impact of extending the number of tokens from n to $2n-1$ to avoid positional ambiguity. As seen in Table 1b), non-continual encoding used in the continual setting result in severe mAP drop. Recycling Positional Encodings alleviate this. Comparing learned and fixed encodings, we find the learned encodings to work better when the number of encoding tokens corresponds to the sequence length n and the fixed encoding to work best when positional ambiguity is alleviated by extending the number of tokens to $2n-1$. Fixed encoding with $2n-1$ tokens works best overall and is employed in subsequent experiments unless stated otherwise. There is no difference in FLOPs for either strategy, and the difference in parameter count is negligible.

4.1.3 Comparison with prior works

We evaluate the (*Co*)OadTR architectures on THUMOS14 and TVSeries with two sets of features as described in Section 4.1.1. Since no prior OAD works have reported complexity metrics, we measured the FLOPs for TRN [43] based on the publicly available source code to serve as a point of reference. The results of this benchmark are presented in Table 2 and Fig. 2. OadTR and our simplified (continual) one-block (b1) and two-block (b2) versions without decoder and class tokens generally achieve competitive precision in comparison with prior works, surpassing all but OadTR and LSTR. On THUMOS14, our reproduced OadTR results are slightly lower than originally reported [18][3], whereas achieved TVSeries results are higher[4]. The (*Co*)OadTR-b# architecture largely retain precision and allow significantly reduced FLOPs per prediction. Our proposed continual variants *Co*OadTR-b1 and *Co*OadTR-b2 reduce FLOPs by 255× and 6.1×, respectively, compared

[3]The reported 58.3% on THUMOS14 could not be reproduced using their publicly available code.

[4]We attribute our higher mcAP to differences in the feature extraction pipeline.

Table 2: **Online Action Detection** results. FLOPs per prediction are noted for inference on THUMOS14. The **best** and ***next-best*** metrics are highlighted.

Model	Feat.	THUMOS14 mAP (%)	TVSeries mcAP (%)	FLOPs (M)
RED [42]	A.Net	45.3	79.2	-
TRN [43]		47.2	83.7	1387.5
FATS [50]		51.6	81.7	-
IDN [44]		50.0	84.7	-
TFN [51]		55.7	85.0	-
LSTR [45]		**65.3**	88.1	-
OadTR [18]		***58.3***	85.4	2445.6
OadTR†		57.0±0.5	**88.6**±0.1	2445.6
OadTR-b2†		56.6±0.3	***88.3***±0.2	1008.1
OadTR-b1†		56.3±0.2	88.1±0.1	605.5
*Co*OadTR-b2 (ours)		56.8±0.4	87.7±0.6	***410.9***
*Co*OadTR-b1 (ours)		56.1±0.7	87.6±0.7	**9.6**
TRN [43]	Kin.	62.1	86.2	1462.0
FATS [50]		59.0	84.6	-
IDN [44]		60.3	86.1	-
PKD [52]		64.5	86.4	-
LSTR [45]		**69.5**	**89.1**	-
OadTR [18]		***65.2***	87.2	2513.5
OadTR†		64.2±0.3	***88.6***±0.1	2513.5
OadTR-b2†		64.5±0.5	88.3±0.2	1075.7
OadTR-b1†		63.9±0.5	88.1±0.1	673.0
*Co*OadTR-b2 (ours)		64.4±0.1	87.6±0.7	***411.9***
*Co*OadTR-b1 (ours)		64.2±0.4	87.7±0.4	**10.6**

†Using official source code or modifications there-off.

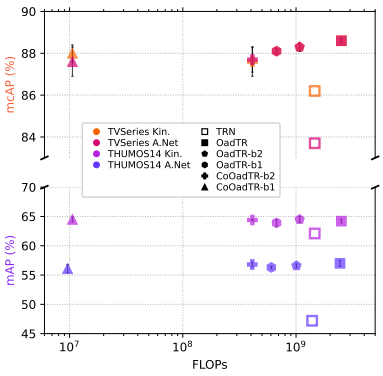


Figure 2: **Visual comparison** of OAD methods on THUMOS14 and TVSeries for backbones trained on ActivityNet 1.3 and Kinetics-400.

to OadTR. On average, continual and non-continual (*Co*)OadTR-b# models achieve similar mAP on THUMOS14, while OadTR-b# have slightly higher mcAP on TVSeries. We attribute these discrepancies to differences in positional encoding.

4.1.4 Audio-Visual Online Action Detection

To showcase the validity of our method in audio-visual settings as well, we explore the addition of audio-features to the Online Action Detection task on THUMOS14. As described in Section 4.2, audio-features are extracted using Mel spectrograms and an AudioSet pre-trained VGGish network [53] (output of the penultimate layer) on 1.0 second windows with a step size of 0.2 seconds to match the 5.0 FPS sampling rate of the video features.

The audio-features by themselves do not provide enough signal to reach good Online Action Detection performance (yielding only 6.7% mAP with an OadTR network). When concatenated with RGB and Flow they do provide a modest improvement as seen in Table 3. On average, this amounts to $+0.6\%$ mAP when combined with ActivityNet features and $+0.5\%$ mAP when used with Kinetics-400 features with shallower models enjoying the largest improvements.

4.2 Audio Classification

4.2.1 Background

Audio Classification is the categorisation of audio waveforms. Though waveform sequences can be used directly [54], it is common to first convert them to spectrograms. Mel spectrograms are obtained by a nonlinear transformation of a frequency scale [55], which is designed based on empirical knowledge about the human auditory system [56]. By employing spectrograms, audio classification can be approached in the same way as image classification [57].

4.2.2 Experiments

We conduct experiments on the Music Genre Classification dataset GTZAN [58]. It consists of 100 30-second clips for each of ten music genres. Each audio clip is sampled at 22,050 Hz. Since there

Table 3: **Audio-Visual** result, THUMOS14.

Model	Feat.	mAP (%)	FLOPs (M)
OadTR		**57.6**$_{\pm 0.6}$	2714.9
OadTR-b2	A.Net	***57.5***$_{\pm 0.5}$	1277.0
OadTR-b1	+	57.4$_{\pm 0.4}$	874.1
*Co*OadTR-b2	AudioSet	56.5$_{\pm 1.1}$	***415.0***
*Co*OadTR-b1		56.8$_{\pm 0.5}$	**13.8**
OadTR		64.4$_{\pm 0.4}$	2781.9
OadTR-b2	Kin.	**65.0**$_{\pm 0.4}$	1344.1
OadTR-b1	+	64.5$_{\pm 0.4}$	941.2
*Co*OadTR-b2	AudioSet	64.7$_{\pm 0.8}$	***416.0***
*Co*OadTR-b1		***64.8***$_{\pm 0.3}$	**14.8**

Table 4: **Audio Classification** results for GTZAN.

Method	Pos. Enc.	Acc. (%)	FLOPs (M)	Par. (K)
Maj. Voting	-	92.0	-	**0**
Trans-b2	learned	**95.0**$_{\pm 0.6}$	47.4	509
Trans-b1	learned	93.8$_{\pm 0.8}$	15.2	***286***
*Co*Trans-b2	fixed	***94.4***$_{\pm 1.0}$	27.0	509
*Co*Trans-b1	learned	93.2$_{\pm 1.1}$	***0.3***	***286***

are no predefined splits for GTZAN, we randomly select 10% of the data for validation and 10% for testing. The input is transformed to a temporal sequence by sliding a one-second window over each 30-second clip with a slide step size of 250ms, leading to 120 one-second clips. These are subsequently converted to Mel spectrograms. We then fine-tune a VGGish network, pre-trained on AudioSet [53] and use the penultimate layer for feature extraction. A batch size of 64 and the Adam optimizer [59] are used with an initial learning rate of 10^{-4}. The learning rate is reduced by a factor of 0.6 on plateau with a tolerance of two epochs, and an early stopping mechanism, where a maximum of 100 epochs are allowed. The VGGIsh base-network attains an accuracy of 86.1% on the dataset of one-second clips with 72.1M parameters and 864.7M FLOPs. Subsequently, the audio features are passed to a (Continual) Transformer Encoder which has 16 attention heads, an embedding dimension of 192 and an MLP dimension of 384. The Transformer Encoder is trained on the whole temporal sequence using a batch size of 32 and the AdamW optimizer [60] with a learning rate of 10^{-5} and a weight decay of 10^{-4} for 50 epochs. Since the Transformer Encoder is trained on entire 30-second clips, there are less data points available for this training. Accordingly, the size of the validation set is increased to 18%. All audio classification training procedures were carried out on a single Nvidia RTX 2080 Ti GPU. Table 4 presents the accuracy and efficiency of regular and Continual Transformers during online inference. As a baseline, we also include the result of majority voting among the clips to classify the entire sequence. The Continual Transformers obtain similar accuracy as regular a Transformers while consuming $1.76\times$ less FLOPS when using two blocks and $51.5\times$ less FLOPS when using one Transformer Encoder block.

5 Conclusion

In this work, we presented Continual Transformers, a redundancy-free reformulation of Transformers tailored for online inference. Central to the Continual Transformer are the Continual Retroactive and Single-Output Attention operations, which produce outputs identical to the original Scaled Dot-Product Attention for continual input sequences, while greatly reducing the time and memory complexity per prediction. The applicability of Continual Transformer architectures was experimentally validated in Online Action Detection and Online Audio Classification settings, observing upwards of multiple orders of magnitude reduction in time complexity for lightweight architectures at modest accuracy concessions. Continual Transformers constitute an algorithmic innovation, which could make possible hitherto unseen precision, speed, and power efficiency in online inference use-cases. With applications spanning enhanced perception and reactivity of robots and autonomous vehicles, weather forecasting, price prediction and surveillance, we hope it will be used for the common good.

Acknowledgments and Disclosure of Funding

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

References

[1] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. "3D Convolutional Neural Networks for Human Action Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.1 (2013), pp. 221–231. DOI: 10.1109/TPAMI.2012.59.

[2] 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)*. 2017, pp. 4724–4733.

[3] Gül Varol, Ivan Laptev, and Cordelia Schmid. "Long-Term Temporal Convolutions for Action Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.6 (2018), pp. 1510–1517. DOI: 10.1109/TPAMI.2017.2712608.

[4] Sijie Yan, Yuanjun Xiong, and Dahua Lin. "Spatial temporal graph convolutional networks for skeleton-based action recognition". In: *AAAI Conference on Artificial Intelligence*. 2018, pp. 7444–7452.

[5] 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.

[6] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 30. 2017, pp. 5998–6008.

[7] Anurag Arnab et al. "ViViT: A Video Vision Transformer". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[8] Arian Bakhtiarnia, Qi Zhang, and Alexandros Iosifidis. "Multi-Exit Vision Transformer for Dynamic Inference". In: *British Machine Vision Conference (BMVC)* (2021).

[9] Hugo Touvron et al. "Augmenting Convolutional networks with attention-based aggregation". In: *preprint, arXiv:2112.13692* abs/2112.13692 (2021).

[10] Marcella Cornia, Lorenzo Baraldi, and Rita Cucchiara. "SMArT: Training Shallow Memory-aware Transformers for Robotic Explainability". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 1128–1134. DOI: 10.1109/ICRA40945.2020.9196653.

[11] Kateryna Chumachenko, Alexandros Iosifidis, and Moncef Gabbouj. "Self-attention fusion for audiovisual emotion recognition with incomplete data". In: *arXiv preprint arXiv:2201.11095* (2022).

[12] Arian Bakhtiarnia, Qi Zhang, and Alexandros Iosifidis. "Single-Layer Vision Transformers for More Accurate Early Exits with Less Overhead". In: *preprint, arXiv:2105.09121* (2021).

[13] Haroon Idrees et al. "The THUMOS challenge on action recognition for videos "in the wild"". In: *Computer Vision and Image Understanding* 155 (2017), pp. 1–23. ISSN: 1077-3142.

[14] George Tzanetakis, Georg Essl, and Perry Cook. *Automatic Musical Genre Classification Of Audio Signals*. 2001. URL: http://ismir2001.ismir.net/pdf/tzanetakis.pdf.

[15] Lukas Hedegaard and Alexandros Iosifidis. "Continual 3D Convolutional Neural Networks for Real-time Processing of Videos". In: *preprint, arXiv:2106.00050* (2021). Apache 2.0 Licence.

[16] Lukas Hedegaard, Negar Heidari, and Alexandros Iosifidis. "Online Skeleton-based Action Recognition with Continual Spatio-Temporal Graph Convolutional Networks". In: *preprint, arXiv: 2203.11009* (2022). Apache 2.0 Licence.

[17] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations (ICLR)*. 2021.

[18] Xiang Wang et al. "OadTR: Online Action Detection with Transformers". In: *International Journal of Computer Vision (ICCV)* (2021). MIT License. URL: https://github.com/wangxiang1230/OadTR.

[19] Nicolas Carion et al. "End-to-End Object Detection with Transformers". In: *European Conference on Computer Vision (ECCV)*. Ed. by Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm. 2020, pp. 213–229.

[20] Yuan Gong, Yu-An Chung, and James Glass. "AST: Audio Spectrogram Transformer". In: *Proc. Interspeech 2021*. 2021, pp. 571–575. DOI: 10.21437/Interspeech.2021-698.

[21] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. "Efficient Transformers: A Survey". In: *arXiv:2009.06732* (2020).

[22] Niki Parmar et al. “Image Transformer”. In: *International Conference on Machine Learning (ICML)*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, July 2018, pp. 4055–4064.

[23] Iz Beltagy, Matthew E. Peters, and Arman Cohan. “Longformer: The Long-Document Transformer”. In: *arXiv:2004.05150* (2020).

[24] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. “Reformer: The Efficient Transformer”. In: *International Conference on Learning Representations (ICLR)*. 2020.

[25] Sinong Wang et al. “Linformer: Self-Attention with Linear Complexity”. In: *arXiv:2006.04768* (2020).

[26] Yunyang Xiong et al. “Nyströmformer: A Nyström-based Algorithm for Approximating Self-Attention”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* (2021).

[27] Krzysztof Marcin Choromanski et al. “Rethinking Attention with Performers”. In: *International Conference on Learning Representations (ICLR)*. 2021.

[28] Myle Ott et al. “fairseq: A Fast, Extensible Toolkit for Sequence Modeling”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 48–53. DOI: 10.18653/v1/N19-4009.

[29] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. “Self-Attention with Relative Position Representations”. In: *North American Chapter of the Association for Computational Linguistics (NAACL)*. 2018.

[30] Cheng-Zhi Anna Huang et al. “Music Transformer: Generating Music with Long-Term Structure”. In: *International Conference on Learning Representations (ICLR)*. 2019.

[31] Zihang Dai et al. “Transformer-XL: Attentive Language Models beyond a Fixed-Length Context”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, July 2019, pp. 2978–2988. DOI: 10.18653/v1/P19-1285.

[32] Shun Kiyono, Sosuke Kobayashi, Jun Suzuki, and Kentaro Inui. “SHAPE: Shifted Absolute Position Embedding for Transformers”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 3309–3321. DOI: 10.18653/v1/2021.emnlp-main.266.

[33] Tatiana Likhomanenko et al. “CAPE: Encoding Relative Positions with Continuous Augmented Positional Embeddings”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 16079–16092.

[34] Mostafa Dehghani et al. “Universal Transformers”. In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=HyzdRiR9Y7.

[35] Yining Ma et al. “Learning to Iteratively Solve Routing Problems with Dual-Aspect Collaborative Transformer”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan. 2021.

[36] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

[37] Roeland De Geest et al. “Online Action Detection”. In: *European Conference on Computer Vision (ECCV)*. 2016, pp. 269–284.

[38] Zheng Shou, Dongang Wang, and Shih-Fu Chang. “Temporal Action Localization in Untrimmed Videos via Multi-stage CNNs”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1049–1058. DOI: 10.1109/CVPR.2016.119.

[39] Huijuan Xu, Abir Das, and Kate Saenko. “R-C3D: Region Convolutional 3D Network for Temporal Activity Detection”. In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 5794–5803. DOI: 10.1109/ICCV.2017.617.

[40] Zheng Shou et al. “CDC: Convolutional-De-Convolutional Networks for Precise Temporal Action Localization in Untrimmed Videos”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1417–1426. DOI: 10.1109/CVPR.2017.155.

[41] Chao-Yuan Wu et al. “Long-Term Feature Banks for Detailed Video Understanding”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 284–293. DOI: 10.1109/CVPR.2019.00037.

[42] Jiyang Gao, Zhenheng Yang, and Ram Nevatia. “RED: Reinforced Encoder-Decoder Networks for Action Anticipation”. In: *British Machine Vision Conference (BMVC)*. 2017.

[43] Mingze Xu et al. “Temporal Recurrent Networks for Online Action Detection”. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. MIT Licence. 2019, pp. 5531–5540. DOI: 10.1109/ICCV.2019.00563. URL: https://github.com/xumingze0308/TRN.pytorch.

[44] H. Eun et al. “Learning to Discriminate Information for Online Action Detection”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 806–815.

[45] Mingze Xu et al. “Long Short-Term Transformer for Online Action Detection”. In: *Conference on Neural Information Processing Systems (NeurIPS)*. 2021.

[46] Limin Wang et al. “Temporal Segment Networks for Action Recognition in Videos”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.11 (2019), pp. 2740–2755. DOI: 10.1109/TPAMI.2018.2868668.

[47] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. “ActivityNet: A large-scale video benchmark for human activity understanding”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 961–970. DOI: 10.1109/CVPR.2015.7298698.

[48] MMAction2 Contributors. *OpenMMLab’s Next Generation Video Understanding Toolbox and Benchmark*. https://github.com/open-mmlab/mmaction2. Apache 2.0 License. 2020.

[49] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2016. DOI: 10.1109/cvpr.2016.90.

[50] Young Hwi Kim, Seonghyeon Nam, and Seon Joo Kim. “Temporally smooth online action detection using cycle-consistent future anticipation”. In: *Pattern Recognition* 116 (2021), p. 107954. ISSN: 0031-3203.

[51] Hyunjun Eun et al. “Temporal filtering networks for online action detection”. In: *Pattern Recognition* 111 (2021), p. 107695.

[52] Peisen Zhao et al. “Privileged Knowledge Distillation for Online Action Detection”. In: *preprint, arXiv:2011.09158* abs/2011.09158 (2020).

[53] Shawn Hershey et al. “CNN architectures for large-scale audio classification”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Mar. 2017. DOI: 10.1109/icassp.2017.7952132.

[54] Jongpil Lee, Taejun Kim, Jiyoung Park, and Juhan Nam. “Raw Waveform-based Audio Classification Using Sample-level CNN Architectures”. In: *NIPS, Machine Learning for Audio Signal Processing Workshop (ML4Audio)* (2017).

[55] S. S. Stevens, J. Volkmann, and E. B. Newman. “A Scale for the Measurement of the Psychological Magnitude Pitch”. In: *The Journal of the Acoustical Society of America* 8.3 (1937), pp. 185–190. DOI: 10.1121/1.1915893.

[56] Keunwoo Choi, George Fazekas, and Mark Sandler. “Automatic tagging using deep convolutional neural networks”. In: *International Society of Music Information Retrieval Conference (ISMIR)* (2016).

[57] Kamalesh Palanisamy, Dipika Singhania, and Angela Yao. “Rethinking CNN Models for Audio Classification”. In: *arXiv:2007.11154* (2020).

[58] G. Tzanetakis and P. Cook. “Musical genre classification of audio signals”. In: *IEEE Transactions on Speech and Audio Processing* 10.5 (July 2002), pp. 293–302. DOI: 10.1109/tsa.2002.800560.

[59] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations (ICLR)*. Ed. by Yoshua Bengio and Yann LeCun. 2015.

[60] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization”. In: *International Conference on Learning Representations (ICLR)*. 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Limitations are described in the problem setting, intro, and when relevant throughout the paper.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Some use cases with societal risks noted in conclusion.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Details supplied in appendix.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] Licences were added in appropriate reference for code or data used.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] A URL to the experiment code will be provided in the final version. During review, the code was submitted as supplemental material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] Usage of 3rd party code and data follows relevant licence guidelines.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] Not relevant.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] Not relevant.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] Not relevant.

A Appendix

A.1 Scaling properties of Continual and regular Multi-head Attention

A detailed account for the floating point operations involved in computing Regular-, Continual Retroactive-, and Single-output Scaled Product Attentions is given in Tables 5, 6, and 7.

Table 5: **Floating Point Operations** for the Scaled Dot-Product Attention in Eq. (1). $\mathbf{D}^{-1}(\cdot)$ can be efficiently computed as element-wise multiplication with $\mathbf{AV}$.

	Mul.	Add	Exp
Eq. (1.1)	$n^2d + nd$	$nd(n-1)$	0
Eq. (1.2)	$n^2d + nd$	$n^2(d-1)$	n^2
Eq. (1.3)	0	$n(n-1)$	0

Table 6: **Floating Point Operations** for the Continual Retroactive Dot-Product Attention in Eqs. (2) to (6). The outputs of the exponentials in Eq. (2) and Eq. (3) can be reused in Eq. (4) and Eq. (5) respectively, and are omitted in the count.

	Mul.	Add	Exp
Eq. (2)	$2(n-1)d$	$2(n-2)d + 2(n-1)$	$2(n-1)$
Eq. (3)	$nd + n + d$	$nd + (n-1) + d$	n
Eq. (4)	$2(n-1)d$	$2(n-1)d$	0
Eq. (5)	nd	$(n-1)d$	0
Eq. (6)	$nd + n$	0	0

Table 7: **Floating Point Operations** for the Continual Single-Output SDA in Eq. (7).

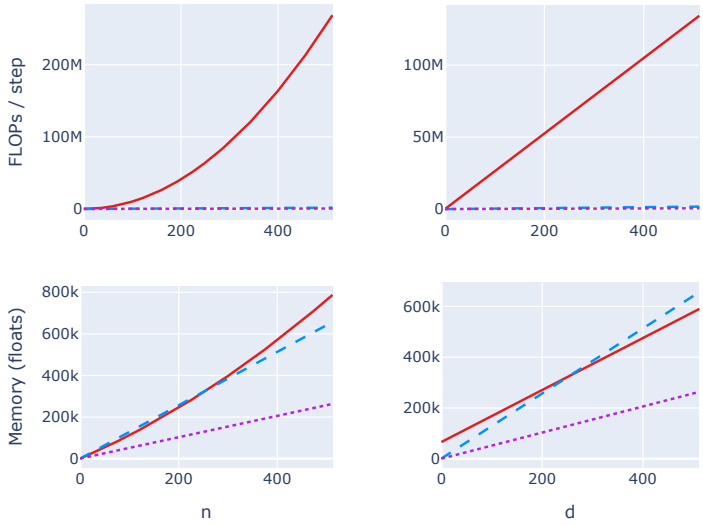
	Mul.	Add	Exp
Eq. (7.1)	$nd + d$	$(n-1)d + n - 1$	0
Eq. (7.2)	$nd + d$	$n(d-1)$	n

Fig. 3 illustrates the scaling of FLOPs and memory footprint with increasing sequence length n and embedding dimension d. Here, the Continual Retroactive and Single-Output SDAs spend significantly less FLOPs than the Regular SDA, which scales $\mathcal{O}(n^2)$ as opposed to $\mathcal{O}(nd)$ the continual variants. The Continual Single-Output SDA reduces memory footprint for all value combinations, and the Continual Retroactive SDA does so when $n \gtrapprox d$.

A.2 Supplemental visualisations

For the visually inclined, we supply a complementary graphical depictions of the Continual Retroactive SDA corresponding to Eqs. (2) to (6) in Fig. 4 and the Single-Output SDA in Eq. (7) in Fig. 5.

A schematic illustration of the Audio Classification experiments architecture is depicted in Fig. 6.



(a) $d = 256$ (b) $n = 256$

Figure 3: **FLOPs/step and memory footprint** for Regular, Continual Retroactive, and Continual Single-Output Scaled Dot-Product Attention at varying sequence length n and embedding dimension d. Column (a) has d fixed to 256; Column (b) has n fixed to 256.

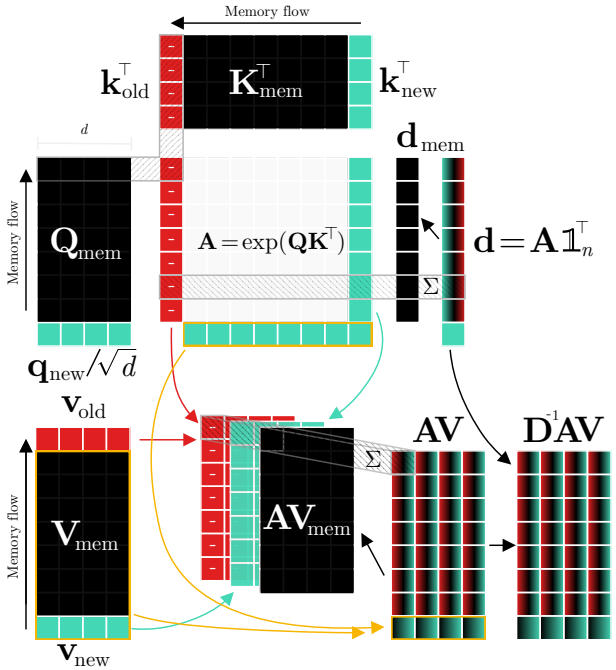


Figure 4: **Continual Retroactive Dot-Product Attention**. The query ($\mathbf{Q}$), key ($\mathbf{K}$), and value ($\mathbf{V}$) matrices are aggregated over time by caching the step vectors $\mathbf{q}_{\text{new}}$, $\mathbf{k}_{\text{new}}$, and $\mathbf{v}_{\text{new}}$ in each their FIFO queue (denoted by $\square_{\text{mem}}$). During each step, only the entries of $\mathbf{A}$ associated with $\mathbf{q}_{\text{new}}$, $\mathbf{k}_{\text{new}}$ and the oldest $\mathbf{K}$ step, $\mathbf{k}_{\text{old}}$ are computed. The diagonal entries of the row-normalisation matrix $\mathbf{D}$ as well as the $\mathbf{AV}$ can be updated retroactively by subtracting features corresponding to $\mathbf{k}_{\text{old}}$ and adding features related to $\mathbf{k}_{\text{new}}$ to the cached outputs of the previous step, $\mathbf{D}_{\text{mem}}$ and $\mathbf{AV}_{\text{mem}}$, respectively.

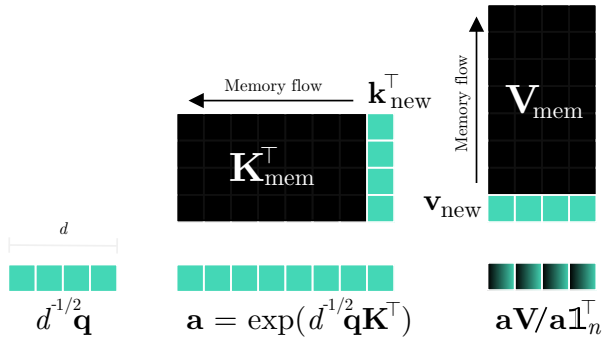


Figure 5: **Continual Single-Output Dot-Product Attention**. The key ($\mathbf{K}$) and value ($\mathbf{V}$) matrices are aggregated over time by caching the step vectors $\mathbf{k}_{\text{new}}$ and $\mathbf{v}_{\text{new}}$ in a FIFO queue. During each step, only the attention output associated with $\mathbf{q}$ is computed.

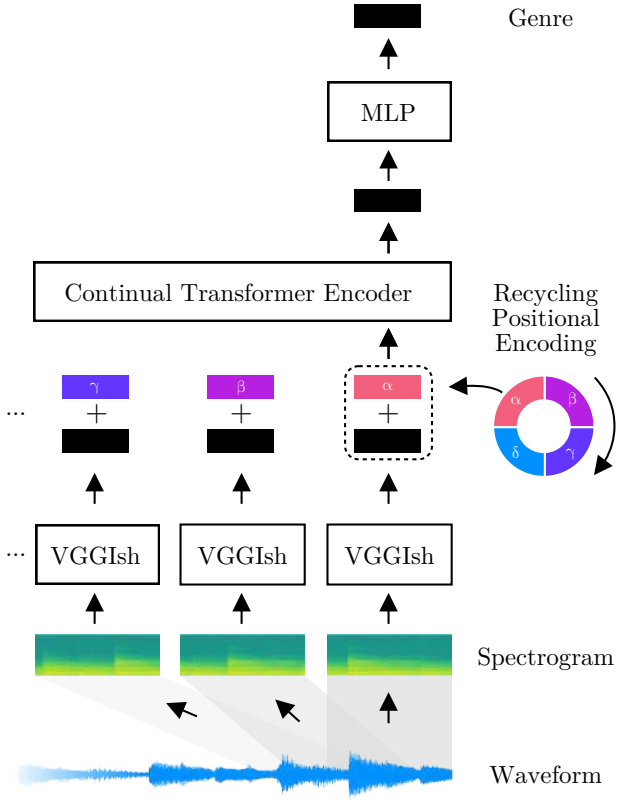


Figure 6: **Audio Classification Architecture**.

8.4 Continual Spatio-Temporal Graph Convolutional Networks for Online Skeleton-based Human Action Recognition

The appended paper follows.

Online Skeleton-based Action Recognition with Continual Spatio-Temporal Graph Convolutional Networks

Lukas Hedegaard, Negar Heidari, and Alexandros Iosifidis
Department of Electrical and Computer Engineering, Aarhus University, Denmark
{lhm, negar.heidari, ai}@ece.au.dk

***Abstract*—Graph-based reasoning over skeleton data has emerged as a promising approach for human action recognition. However, the application of prior graph-based methods, which predominantly employ whole temporal sequences as their input, to the setting of online inference entails considerable computational redundancy. In this paper, we tackle this issue by reformulating the Spatio-Temporal Graph Convolutional Neural Network as a Continual Inference Network, which can perform step-by-step predictions in time without repeat frame processing. To evaluate our method, we create a continual version of ST-GCN, *Co*ST-GCN, alongside two derived methods with different self-attention mechanisms, *Co*AGCN and *Co*S-TR. We investigate weight transfer strategies and architectural modifications for inference acceleration, and perform experiments on the NTU RGB+D 60, NTU RGB+D 120, and Kinetics Skeleton 400 datasets. Retaining similar predictive accuracy, we observe up to 109× reduction in time complexity, on-hardware accelerations of 26×, and reductions in maximum allocated memory of 52% during online inference.**

***Index Terms*—Continual Inference Networks, Graph-Convolution, Attention, Convolutional Neural Network, Skeleton-based Action Recognition, Human Activity Recognition, Online Inference**

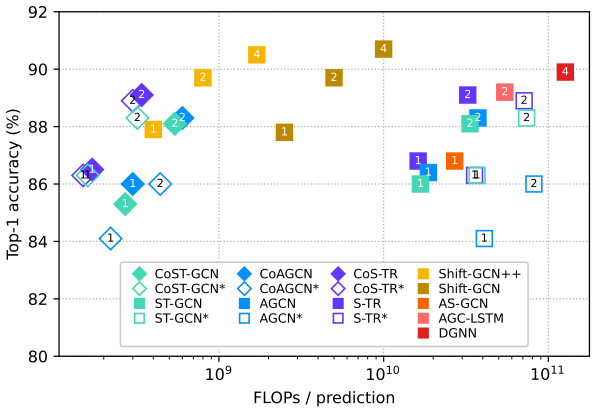


Fig. 1: **Accuracy/complexity trade-off** on NTU RGB+D 60 X-Sub for ◆ *Continual* and ■ prior methods during online inference. Numbers denote streams for each method. *Architecture modification with stride one and no padding.

I. Introduction

A human action can be described by a temporal sequence of human body poses, each of which is represented by a set of spatial joint coordinates forming a body skeleton. Accordingly, skeleton-based action recognition methods process a sequence of skeletons (instead of an image sequence) to recognize the performed action. Compared with predicting actions from videos, a sequence of skeleton data not only gives the spatial and temporal features of the body poses, but also provides robustness against different background variations and context noise [1]. The estimation of such skeletal data has become a staple in the human action recognition toolkit thanks to publicly available toolboxes such as OpenPose [2].

Early deep learning methods for skeleton-based action recognition either rearrange the body joint coordinates of each skeleton to make a pseudo-image which is used to train a CNN model [3, 4, 5, 6, 7, 8], or concatenate the human body joints as a sequence of feature vectors and train a RNN model [9, 10, 11, 12, 13, 14]. However, these methods cannot take advantage of the non-Euclidean structure of the skeletons. Recently, Graph Convolutional Networks (GCNs) have shown prowess in the modeling of skeleton data. ST-GCN [15] was the first GCN-based method proposed for skeleton-based action recognition. It uses spatial graph convolutions to extract the per time-step features of each skeleton and employs temporal convolutions to capture time-varying dynamics throughout the skeleton sequence. Since its publication, several methods have sprung from ST-GCN, which enhance feature extraction or optimize the structure of the model.

2s-AGCN [16] proposed to learn the graph structure in each GCN layer adaptively based on input graph node similarity and also utilized an attention method which highlights both the existing spatial connections in the graph (bones) and new potential connections between them. MS-AAGCN [17] extended 2s-AGCN by proposing a multi-stream framework which uses four different data streams for training the model. Moreover it enhanced the adaptive graph convolution in 2s-AGCN with a spatio-temporal channel attention module to highlight the most important skeletons, nodes in each skeleton, and features of each node. DGNN [18] modeled the spatial connections between the graph nodes with a directed graph and utilized both node features and edge features simultaneously. HyperGNN [19] captured the non-physical connections between the nodes by constructing hyperedges which help to extract both

local and global features in each graph. FGCN [20] proposed to extract coarse to fine spatio-temporal features by a multi-stage temporal sampling strategy and introduced a feedback mechanism in graph convolution to transfer the high-level features to the shallower layers of the network. Similarly, MS-G3D [21] has proposed multi-scale graph convolutions for long-range feature extraction.

Unfortunately, the high computational complexity of these GCN-based methods makes them infeasible in real-time applications and resource-constrained online inference settings. Multiple approaches have been explored to increase the efficiency of skeleton-based action recognition recently: GCN-NAS [22] and PST-GCN [23] are neural architecture search based methods which try to find an optimized ST-GCN architecture to increase the efficiency of the classification task; ShiftGCN [24] replaces graph and temporal convolutions with a zero-FLOPs shift graph operation and point-wise convolutions as an efficient alternative to the feature-propagation rule for GCNs [25]; ShiftGCN++[26] boost the efficiency of ShiftGCN further via progressive architecture search, knowledge-distillation, explicit spatial positional encodings, and a Dynamic Shift Graph Convolution; SGN [27] utilizes semantic information such as 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; TA-GCN [28] tries to make inference more efficient by selecting a subset of key skeletons, which hold the most important features for action recognition, from a sequence to be processed by the spatio-temporal convolutions.

Yet, none of the above-described GCN-based methods are tailored to online inference, were the input is a continual stream of skeletons and step-by-step predictions are required. During online inference, these methods would need to rely on sliding window-based processing, i.e., storing the $T-1$ prior skeletons, appending the newest skeleton to get a sequence of length T, and then performing their prediction on the whole sequence. In this paper, we reduce such redundant computations by reformulating the ST-GCN and its derived methods as a Continual Inference Network, which processes skeletons one by one and produces updated predictions for each time-step without the need to include past skeletons in every input as is the case for the prior GCN-based methods. This is achieved by using Continual Convolutions in place of regular ones for aggregating temporal information. In particular, we propose the *Continual* Spatio-Temporal Graph Convolutional Network (*Co*ST-GCN), *Co*AGCN, and *Co*TR-S and evaluate them on the skeleton-based action recognition datasets NTU RGB+D 60 [29], NTU RGB+D 120 [30], and Kinetics Skeleton 400 [31] with striking results: Our continual models achieve up to 108× FLOPs reduction, 26× speedup, and 52% reduction in max allocated GPU memory compared to the corresponding non-continual models.

The remainder of the paper is structured as follows: Section II provides an introduction to skeleton-based action recognition and of the related methods, from which we derive a continual counterpart, Section III describes Continual Inference Networks, and Section IV presents our proposed *Continual* Spatio-temporal Graph Convolutional Networks. Experiments on weight transfer strategies, performance benchmarks, and comparisons with prior works are offered in Section V, and a conclusion is given in Section VI.

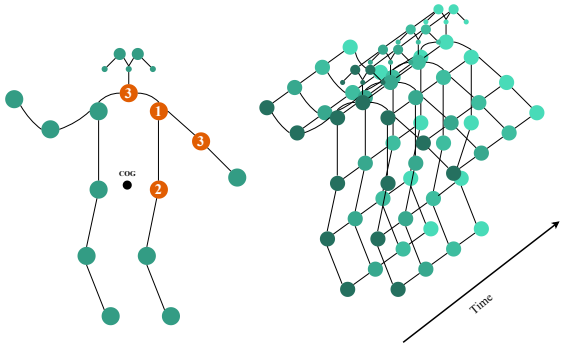


Fig. 2: **Graph illustration** for a spatially partitioned skeleton (left) and spatio-temporal graph (right).

II. Related Works

A. Spatio-Temporal Graph Convolutional Network

GCN-based models for skeleton-based action recognition [15, 16, 18, 22, 23, 27, 28] operate on sequences of skeleton graphs. The spatio-temporal graph of skeletons $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ has the human body joint coordinates as nodes $\mathcal{V}$ and the spatial and temporal connections between them as edges $\mathcal{E}$. Figure 2 (right) illustrates such a spatio-temporal graph where the spatial graph edges encode the human bones and the temporal edges connect the same joints in subsequent time-steps. We model this graph as a tensor $\mathbf{X} \in \mathbb{R}^{C^{(0)} \times T \times V}$, where $C^{(0)}$ is the number of input-channels of each joint, T denotes the number of skeletons in a sequence, and V is the number of joints in each skeleton. A binary adjacency matrix $\mathbf{A} \in \mathbb{R}^{V \times V}$ encodes the skeleton-structure with ones in positions connecting two vertices in a skeleton and zeros elsewhere.

The ST-GCN [15] and AGCN [16] methods refine the spatial structure of each skeleton by employing a partitioning method which categorizes neighboring nodes of each body joint into three subsets: (1) the root node itself, (2) the root's neighboring nodes which are closer to the skeleton's center of gravity (COG) than the root itself, and (3) the remaining neighboring nodes of the root node. An example of this subset partitioning is shown in Figure 2 (left). Accordingly, the graph-structure of each skeleton is represented by three normalized binary adjacency matrices $\left\{\mathbf{A}_p \in \mathbb{R}^{V \times V} \mid p = 1, 2, 3\right\}$, each of which is defined as

$$\hat{\mathbf{A}}_p = \mathbf{D}_p^{-\frac{1}{2}} \mathbf{A}_p \mathbf{D}_p^{-\frac{1}{2}}, \tag{1}$$

where $\mathbf{D}_p$ denotes the degree matrix of the neighboring subset p. Inspired by the GCN aggregation rule [25], the spatial graph convolution receives the hidden representation of the previous

layer $\mathbf{H}^{(l-1)}$ as input, where $\mathbf{H}^{(0)} = \mathbf{X}$, and performs the following graph convolution (GC) transformation:

$$\text{GC}\left(\mathbf{H}^{(l-1)}\right) = \sigma\left(\text{Res}(\mathbf{H}^{(l-1)}) + \text{BN}\left(\sum_p (\hat{\mathbf{A}}_p \otimes \mathbf{M}_p^{(l)})\mathbf{H}^{(l-1)}\mathbf{W}_p^{(l)}\right)\right) \tag{2}$$

where $\sigma(\cdot)$ denotes a ReLU non-linearity, $\mathbf{W}_p^{(l)} \in \mathbb{R}^{C^{(l)} \times C^{(l-1)}}$ is the weight matrix which transforms the features of the neighboring subset p and $BN(\cdot)$ denotes batch normalization. Moreover, a learnable matrix $\mathbf{M}_p^{(l)} \in \mathbb{R}^{V \times V}$ is multiplied element-wise with its corresponding adjacency matrix $\hat{\mathbf{A}}_p$ as an attention mechanism that highlights the most important connections in each spatial graph. In order to retain the model's stability, the input to a layer is added to the transformed features through a residual connection $\text{Res}(\mathbf{H}^{(l-1)})$ which is defined as:

$$\text{Res}(\mathbf{H}^{(l-1)}) = \begin{cases} \mathbf{H}^{(l-1)}, & C^{(l)} = C^{(l-1)}, \\ \mathbf{H}^{(l-1)}\mathbf{W}_{res}^{(l)}, & \text{otherwise}, \end{cases} \tag{3}$$

where $\mathbf{W}_{res}^{(l)} \in \mathbb{R}^{C^{(l)} \times C^{(l-1)}}$ is a learnable mapping matrix which transforms the layer's input to have the same channel dimension as the layer's output.

The graph convolution block is followed by a temporal convolution, $\text{TC}(\cdot)$, which propagates the features of the graph nodes through different time steps to capture the motions taking place in an action. In the temporal graph, each node only has two fixed neighbors which are its corresponding nodes in the previous and next skeletons. The adjacency matrices and partitioning process are not involved in temporal feature propagation. In practice, the temporal convolution is a standard 2D convolution which receives the output of the graph convolution obtained in Eq. 2 and performs a transformation with a kernel of size $C^{(l)} \times K \times 1$ to keep the node feature dimension unchanged and aggregate the features through K consecutive time steps.

The whole spatio-temporal convolution block has the form

$$\mathbf{H}^{(l)} = \sigma\left(\text{Res}(\mathbf{H}^{(l-1)}) + \text{BN}(\text{TC}(\text{GC}(\mathbf{H}^{(l-1)})))\right). \tag{4}$$

The ST-GCN model is composed of multiple such spatio-temporal convolutional blocks. A global average pool and fully connected layer perform the final classification.

B. Adaptive Graph Convolutional Neural Networks

The fixed graph structure used in Eq. (2) is defined based on natural connections in the human body skeleton which restricts the model's capacity and flexibility in representing different action classes. However, for some action classes such as "touching head" it makes sense to model a connection between hand and head even though such a connection is not naturally present in the skeleton. AGCN [16] allows for such possibilities by adopting an adaptive graph convolution which utilizes a data-dependent graph structure as follows:

$$\text{AGC}\left(\mathbf{H}^{(l-1)}\right) = \sigma\left(\text{Res}(\mathbf{H}^{(l-1)}) + \text{BN}\left(\sum_p (\hat{\mathbf{A}}_p + \mathbf{M}_p^{(l)})\mathbf{H}^{(l-1)}\mathbf{W}_p^{(l)}\right)\right), \tag{5}$$

where $\mathbf{M}_p^{(l)}$ is defined as:

$$\mathbf{M}_p^{(l)} = \mathbf{B}_p^{(l)} + \mathbf{C}_p^{(l)} \tag{6}$$

The attention matrix in this definition is composed of two learnable matrices which are optimized along with other model parameters in an end-to-end manner. $\mathbf{B}_p^{(l)} \in \mathbb{R}^{N \times N}$ is a squared matrix that can be unique for each layer and each sample, and $\mathbf{C}_p^{(l)} \in \mathbb{R}^{N \times N}$ is a similarity matrix whose elements determine the strength of the pair-wise connections between nodes. This matrix is computed by first transforming the feature matrix $\mathbf{H}^{(l-1)} \in \mathbb{R}^{C^{(l-1)} \times T \times V}$ with two embedding matrices $\mathbf{W}_{p\theta}^{(l)}$, $\mathbf{W}_{p\phi}^{(l)}$ of size $C^{de} \times C^{(l-1)}$. The obtained feature maps are then reshaped to $C^{de}T \times V$ and multiplied to obtain the $\mathbf{C}_p^{(l)} \in \mathbb{R}^{N \times N}$ matrix as follows:

$$\mathbf{C}_p^{(l)} = \text{softmax}({\mathbf{H}^{(l-1)}}^\top {\mathbf{W}_{p\theta}^{(l)}}^\top \mathbf{W}_{p\phi}^{(l)} \mathbf{H}^{(l-1)}), \tag{7}$$

where softmax normalizes the matrix values. The additive attention mechanism in Eq. (5), thus, lets the adaptive graph convolution in Eq. (7) model the skeleton structure as a fully connected graph.

C. Skeleton-based Spatial Transformer Networks

S-TR [32] is an attention-based method which models dependencies between body joints at each time step using the self-attention operation found in Transformers [33]. In this method, a Spatial Self-Attention (SSA) module is designed to adaptively learn data-dependent pairwise body joint correlations using multi-head self-attention.

The SSA module at each layer l applies trainable query, key, and value transformations $\mathbf{W}_q^{(l)} \in \mathbb{R}^{C^{(l-1)} \times dq}$, $\mathbf{W}_k^{(l)} \in \mathbb{R}^{C^{(l-1)} \times dk}$, $\mathbf{W}_v^{(l)} \in \mathbb{R}^{C^{(l-1)} \times dv}$ on the feature vector $\mathbf{h}_i^t \in \mathbb{R}^{C^{(l-1)}}$ of node i at time step t to obtain the query, key, and value vectors $\mathbf{q}_i^t \in \mathbb{R}^{dq}$, $\mathbf{k}_i^t \in \mathbb{R}^{dk}$, $\mathbf{v}_i^t \in \mathbb{R}^{dv}$. The correlation weight for each pair of i, j nodes at time t is obtained using a query-key dot product

$$\alpha_{ij}^t = {\mathbf{q}_i^t}^\top \mathbf{k}_j^t. \tag{8}$$

The updated feature vector of node i at time t has size $C^{(l)}$ and is obtained using a weighted feature aggregation of value vectors:

$$\bar{\mathbf{h}}_i^t = \sum_j \text{softmax}_j\left(\frac{\alpha_{ij}^t}{\sqrt{dk}}\right)\mathbf{v}_j^t. \tag{9}$$

For each attention head, the feature transformation is performed with a different set of learnable parameters while the transformation matrices are shared across all the nodes. The output features of the SSA module are finally computed by

applying a learnable linear transformation on the concatenated features from S attention heads:

$$\bar{\mathbf{h}}_i^t = (\Big\|_{s=1}^{S} \bar{\mathbf{h}}_{is}^t)\mathbf{W}_o. \tag{10}$$

SSA has similarities to a graph convolution operation on a fully connected graph for which the node connection weights are learned dynamically. The first three layers of the S-TR model extract features with GC and TC blocks as defined in Eq. 4 while in the remaining layers of the model SSA substitutes GC.

III. Continual Inference Networks

First introduced in [34] and subsequently formalized in [35], Continual Inference Networks are Deep Neural Networks that can operate efficiently on both fixed-size (spatio-)temporal batches of data, where the whole temporal sequence is known up front, as well as on continual data, where new input steps are collected continually and inference needs to be performed efficiently in an online manner for each received frame.

Definition (Continual Inference Network). *A Continual Inference Network is a Deep Neural Network, which*

- *is capable of continual step inference without computational redundancy,*
- *is capable of batch inference corresponding to a non-continual Neural Network,*
- *produces identical outputs for batch inference and step inference given identical receptive fields,*
- *uses one set of trainable parameters for both batch and step inference.*

Recurrent Neural Networks (RNNs) are a common family of Deep Neural Networks, which possess the above-described properties. 3D Convolutional Neural Networks (3D CNNs), Transformers, and Spatio-Temporal Graph Convolutional Networks are not Continual Inference Networks since they cannot make predictions time-step by time-step without considerable computational redundancy; they need to cache a sliding window of prior input frames and assemble them into a fixed-size sequence that is subsequently passed through the network to make a new predictions during online inference.

Recently, *Continual* 3D CNNs were made possible through the proposal of *Continual* 3D Convolutions [34]. Likewise, shallow *Continual* Transformers based on *Continual* Dot-product Attentions were introduced in [35]. We continue this line of work by extending Spatio-Temporal Graph Convolutional Networks (ST-GCNs) with a *Continual* formulation as well. To do so, let us first present and expand on the theory on Continual Convolutions.

A. Continual Convolution

The Continual Convolution operation produces the exact same output as the regular convolution does, but performs the computation in a streaming fashion while caching intermediary results. Consider a single channel 2D convolution over an input $\mathbf{X} \in \mathbb{R}^{T \times V}$ with temporal dimension T and a dimension of V vertices. Given a convolutional kernel with weights

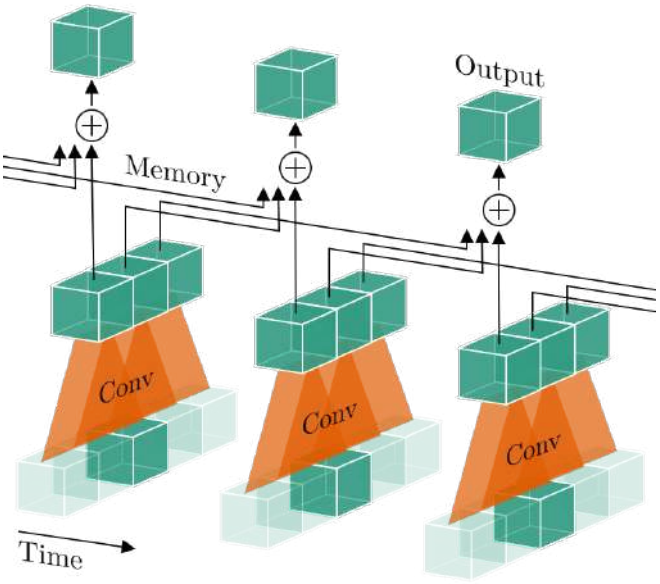


Fig. 3: **Continual Convolutions** are performed in two stages: First, the input is zero-padded and convolved with the convolutional kernel ($K = 3$ in illustration) to produce intermediary results. Subsequently, these are cached and summed up to produce the final output.

$\mathbf{W} \in \mathbb{R}^{K \times V}$, where K is the temporal kernel size, and a bias w_0, a regular convolution would compute the output $\mathbf{y}^{(t)}$ for time-step $t \in K..T$ as

$$\mathbf{y}^{(t)} = w_0 + \sum_{k=1}^{K} \sum_{v=1}^{V} \mathbf{W}_{k,v} \cdot \mathbf{X}_v^{(t-k-1)}. \tag{11}$$

Considering this computation in the context of online processing, where $T \to \infty$ and one input slice $\mathbf{X}^{(t)}$ is revealed in each time step, we find that $K-1$ previous slices, i.e. $(K-1) \cdot V$ values, need to be stored between time-steps.

An alternative computational sequence is used in Continual Convolutions. Here, the input slice $\mathbf{X}^{(t)}$ is convolved with the kernel $\mathbf{W}$ in the same time-step it is received. This is specified in Eq. (12a). The intermediate results are then cached in memory $\mathbf{m}$ ($K-1$ values stored between time-steps) and aggregated according to Eq. (12b).

$$\mathbf{m}^{(t)} = \left[\sum_{v=1}^{V} \mathbf{W}_{k,v} \cdot \mathbf{X}_v^{(t)} : \; k \in 1..K \right] \tag{12a}$$

$$\mathbf{y}^{(t)} = w_0 + \sum_{k=1}^{K} \mathbf{m}_k^{(t-k-1)} \tag{12b}$$

A graphical representation of this is shown in Fig. 3.

B. Delayed Residual

The temporal convolutions of regular Spatio-Temporal Graph Convolution blocks usually employ zero-padding to ensure equal temporal shape for input and output feature maps. This zero-padding is discarded for Continual Convolutions to

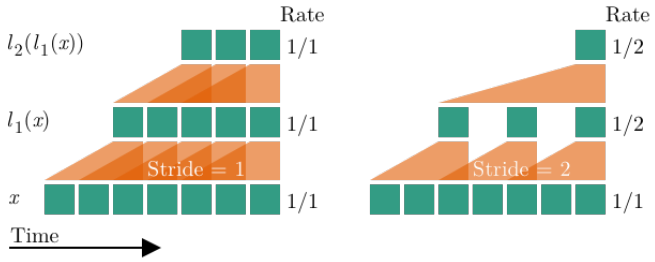


Fig. 4: **Temporal stride** in a Continual Convolution layer l_1 with temporal stride larger than one (right) reduces the prediction rate compared to a layer with stride one (left). The rate reduction is inherited by subsequent layers.

avoid continual redundancies [34]. To retain weight compatibility between the regular and continual networks, a delay to the residual connection is necessary. This delay amounts to

$$k_T + (k_T - 1)(d_T - 1) - p_T - 1 \tag{13}$$

steps, where k_T, d_T, and p_T are respectively the temporal kernel size, dilation, and zero-padding of the corresponding regular convolution.

C. Temporal Stride

In Section III-A, it is assumed that one output is produced for each input received. However, many spatio-temporal networks including ST-GCN [15], AGCN [16], and S-TR [32], use temporal stride > 1 in their temporal convolutions. For offline computation, this has the beneficial effect of reducing the computational and memory complexity, but in the online computational setting, it also reduces the prediction rate. This is illustrated in Fig. 4. For a neural network with L layers, each with a temporal stride s, the effective network stride is given by

$$s_{NN} = \prod_{l=1}^{L} s_l \tag{14}$$

and the corresponding network prediction rate is

$$r_{NN} = 1/s_{NN}. \tag{15}$$

Since a ST-GCN network has two layers with stride two, the corresponding Continual ST-GCN (*Co*ST-GCN) has a prediction rate one fourth the input rate.

IV. Continual Spatio-Temporal Graph Convolutional Networks

Many well-performing methods for skeleton-based action recognition, including the ST-GCN [15], AGCN [16], and S-TR [32], share a common block structure, which can be described by Eq. (4). Here, the main difference between methods lies in how the graph information is processed, i.e. in their definition of GC$(\cdot)$.

The regular skeleton-based methods successively extract complete spatio-temporal skeleton features from the whole sequence with each block before classifying an action. Considering one block in isolation, the spatio-temporal feature extraction is given by a spatial (graph) convolution followed by a regular temporal convolution. Here, graph convolutions operate locally within a time-step[1], whereas the temporal convolution does not. Since the next block l takes as input $\mathbf{H}^{(l-1)}$, the output of the prior block and thereby its temporal convolution, the output of the next spatial (graph) convolution becomes a function of multiple prior time-steps. With regular temporal convolutions, features produced by multiple blocks cannot be trivially disentangled and cached in time. Accordingly online operation with per-skeleton predictions can be attained by caching $T-1$ prior skeletons, concatenating these with the newest skeleton, and performing regular spatio-temporal inference. However, this comes with significant computational redundancy, where the complexity of online frame-wise inference is the same as for clip-based inference.

To alleviate this issue, we propose to employ Continual Convolutions in the temporal modeling of Spatio-temporal Graph Convolutional Networks. By restricting the GC$(\cdot)$ function to only operate locally within a time-step, we can define a *Continual* Spatio-Temporal block by replacing the original temporal 2D convolution with a continual one. To retain weight-compatibility with regular (non-continual) networks we moreover need to delay the residual to keep temporal alignment. Given $\mathbf{H}_{l-1}^{(t)}$, i.e. the features of layer $l-1$ in a time-step t, the feature in layer l at time t is given by

$$\mathbf{H}_{l}^{(t)} = \sigma\left(\text{Delay}(\text{Res}(\mathbf{H}_{l-1}^{(t)})) + \text{BN}(\mathit{Co}\text{TC}(\text{GC}(\mathbf{H}_{l-1}^{(t)})))\right). \tag{16}$$

Here, Delay(Res($\mathbf{H}_{l-1}^{(t)}$)) outputs the delayed residual in a first-in-first-out manner corresponding to the delay of the *Continual* Temporal Convolutional as computed by Eq. (13). A graphical illustration of such a block is seen in Fig. 5. It should be noted that the restriction of temporal locality does influence the computations of some skeleton-based action recognition methods. For example, the AGCN originally computes one vertex attention weighting based on the whole spatio-temporal feature-map, whereas a *Continual* AGCN (*Co*AGCN) computes separate vertex attentions for each time-step.

The resulting *Continual* Spatio-temporal Graph Convolutional Network is defined by stacking multiple such blocks[2] followed by *Continual* Global Average Pooling [34] and a fully connected layer. The Continual Inference Networks retain the same computational complexity as regular networks during clip-based inference, but can perform online frame-by-frame predictions much more efficiently, as detailed in Section IV-A. We should note that all methods, which share the the same structure as ST-GCN, i.e. a decoupled temporal and spatial convolution to perform feature transformation and aggregation over the time domain can be transformed to continual version using the approach outlined above.

A. Computational Complexity

Denote the time complexity of passing a single skeleton frame through the convolutional blocks with stride 1 by $\mathcal{O}(B)$

[1]AGCN is an exception to this, since the additive attention considers a node's features over all time-steps.

[2]Following the original ST-GCN, AGCN, and S-TR architectures, ten blocks were used for the networks in this paper.

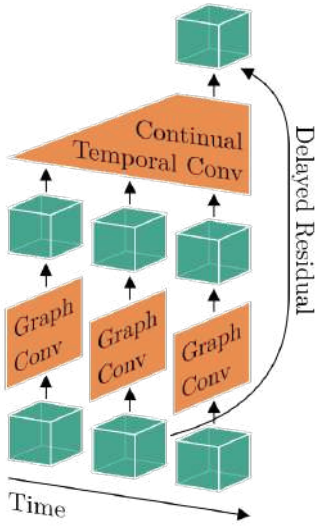


Fig. 5: **Continual Spatio-temporal Graph Convolution Blocks** consist of an in-time Graph Convolution followed by an across-time Continual Convolution (here a kernel size of three is depicted). The residual connection is delayed to ensure temporal alignment with the continual temporal convolution that is weight-compatible with non-continual networks.

and time complexity of utilizing the prediction head by $\mathcal{O}(H)$. Given an effective clip-size T, the complexity of producing a prediction with a regular CNN is approximately $\mathcal{O}(\text{CNN}) \approx T \cdot \mathcal{O}(B) + \mathcal{O}(H)$. For a Continual CNN, the corresponding complexity is $\mathcal{O}(Co\text{CNN}) \approx \mathcal{O}(B) + \mathcal{O}(H)$. Computational savings thus scale linearly with the effective clip-size T and are more prominent the larger $\mathcal{O}(B)$ is compared to $\mathcal{O}(H)$.

V. Experiments

A. Datasets

a) NTU RGB+D 60 [29]*:* A large indoor-captured dataset which is widely used for evaluating skeleton-based action recognition methods. This dataset contains 56,880 action clips and their corresponding 3D skeleton sequences captured by three Microsoft Kinect-v2 cameras from three different views. The clips are performed by 40 different subjects and constitute 60 action classes. The NTU RGB+D 60 dataset comes with two benchmarks, Cross-View (X-View) and Cross-Subject (X-Sub). The X-View benchmark provides 37,920 skeleton sequences coming from the camera views #2 and #3 as training data, and 18,960 skeleton sequences coming from the first camera view as test set. The X-Sub benchmark provides 40,320 skeleton sequences from 20 subjects as training data and 16,560 skeleton sequences from the other 20 subjects as test data. In this dataset, each skeleton has 25 body joints with three different channels each, and each action clip comes with a sequence of 300 skeletons.

b) NTU RGB+D 120 [30]*:* An extension of the NTU RGB+D 60 dataset containing an additional 57,600 skeleton sequences from extra 60 classes. NTU RGB+D 120 is currently the largest dataset providing 3D body joint coordinates for skeletons and in total, it contains 114,480 skeleton sequences from 120 action classes. The action clips in this dataset are performed by 106 subjects and 32 different camera setups are used for capturing the videos. This dataset comes with two benchmarks: Cross-Subject (X-Sub) and Cross-Setup (X-Set). The X-Sub benchmark provides the skeleton sequences of 53 subjects as training data and the remaining skeleton sequences from the other 53 subjects as test data. In the X-Set benchmark, the skeleton sequences with even camera setup IDs are provided as training data and test data contains the remaining skeleton sequences with odd camera setup IDs.

c) Kinetics Skeleton 400 [31]*:* A widely used dataset for action recognition containing 300,000 video action clips of 400 different classes which are collected from YouTube. Skeletons were extracted from each frame of these video clips using the OpenPose toolbox [2]. Each skeleton is represented by 18 body joints and each body joint contains spatial 2D coordinates and the estimation confidence score as its three features. We use the dataset version provided by [15], which contains 240,000 skeleton sequences as training data and 20,000 skeleton sequences as test data, in our experiments.

Conversion Strategy	**Acc. (%)**	**FLOPs (G)**
$\text{Reg}_{s=4}^{p=\text{eq}}$ (baseline)	93.4	16.73
$\text{Reg}_{s=4}^{p=\text{eq}} \xrightarrow{FT} \text{Reg}_{s=1}^{p=0}$	93.8 (+0.4)	36.90 (↑ 2.2×)
$\text{Reg}_{s=4}^{p=\text{eq}} \rightarrow \text{Co}_{s=4}^{p=0}$	93.1 (−0.3)	0.27 (↓ 63.2×)
$\text{Reg}_{s=4}^{p=\text{eq}} \rightarrow \text{Co}_{s=1}^{p=0}$	24.0 (−69.4)	0.16 (↓ 107.7×)
$\text{Reg}_{s=4}^{p=\text{eq}} \rightarrow \text{Co}_{s=1}^{p=0} \xrightarrow{FT} \text{Co}*$	93.2 (−0.2)	0.16 (↓ 107.7×)
$\text{Reg}_{s=4}^{p=\text{eq}} \xrightarrow{FT} \text{Reg}_{s=1}^{p=0} \rightarrow \text{Co}*$	93.8 (+0.4)	0.16 (↓ 107.7×)

TABLE I: **Conversion Strategies** from regular (Reg) to Continual (Co) ST-GCN. Noted is the top-1 X-View validation accuracy on NTU RGB+D 60 and the FLOPs per prediction. The superscript p and subscript s indicate network padding and stride respectively. The arrows $\rightarrow$ and $\xrightarrow{FT}$ denote direct conversion and conversion with subsequent fine-tuning. Parentheses show the change relative to the baseline with colours indicating improvement / deterioration.

B. Experimental Settings

All models were implemented within the PyTorch framework [36] using the Ride library [37]. Models were trained using a SGD optimizer with learning rate 0.1 at batch size 64, momentum of 0.9, and a one-cycle learning rate policy [38] using a cosine annealing strategy. For models which could not fit a batch size of 64 on a Nvidia RTX 2080 Ti, the learning rate was adjusted following the linear scaling rule [39]. Our source code is available at www.github.com/lukashedegaard/continual-skeletons.

C. Conversion and Fine-tuning Strategies

Though regular and Continual CNNs are weight-compatible, the direct transfer of weights is imperfect if the regular CNN was trained with zero-padding [34]. As in most CNNs, it is common practice to utilize padding in skeleton-based spatio-temporal networks to retain the temporal feature size in

Model	Frames per pred	Accuracy (%)		Params (M)	Max mem. (MB)	FLOPs per pred (G)	Throughput (preds/s)	
		X-Sub	X-View				CPU	GPU
ST-GCN	300	86.0	93.4	3.14	45.3	16.73	2.3	180.8
ST-GCN*	300	86.3 (+0.3)	93.8 (+0.4)	3.14	72.6 (160%)	36.90 (↑ 2.2×)	1.1 (↓ 2.1×)	90.4 (↓ 2.0×)
*Co*ST-GCN	4	85.3 (−0.7)	93.1 (−0.3)	3.14	36.0 (79%)	0.27 (↓ 63.2×)	32.3 (↑ 14.0×)	2375.2 (↑ 13.1×)
*Co*ST-GCN*	1	86.3 (+0.3)	93.8 (+0.4)	3.14	36.1 (80%)	0.16 (↓ 107.7×)	46.1 (↑ 20.0×)	4202.2 (↑ 23.2×)
AGCN	300	86.4	94.3	3.47	48.4	18.69	2.1	146.2
AGCN*	300	84.1 (−2.3)	92.6 (−1.7)	3.47	76.4 (158%)	40.87 (↑ 2.2×)	1.0 (↓ 2.1×)	71.2 (↓ 2.0×)
*Co*AGCN	4	86.0 (−0.4)	93.9 (−0.4)	3.47	37.3 (77%)	0.30 (↓ 63.2×)	25.0 (↑ 11.9×)	2248.4 (↑ 15.4×)
*Co*AGCN*	1	84.1 (−2.3)	92.6 (−1.7)	3.47	37.4 (77%)	0.17 (↓ 108.8×)	30.4 (↑ 14.5×)	3817.2 (↑ 26.1×)
S-TR	300	86.8	93.8	3.09	74.2	16.14	1.7	156.3
S-TR*	300	86.3 (−0.5)	92.4 (−1.4)	3.09	111.5 (150%)	35.65 (↑ 2.2×)	0.8 (↓ 2.1×)	85.1 (↓ 1.8×)
*Co*S-TR	4	86.5 (−0.3)	93.3 (−0.5)	3.09	35.9 (48%)	0.22 (↓ 63.2×)	30.3 (↑ 17.8×)	2189.5 (↑ 14.0×)
*Co*S-TR*	1	86.3 (−0.3)	92.4 (−1.4)	3.09	36.1 (49%)	0.15 (↓ 107.6×)	43.8 (↑ 25.8×)	3775.3 (↑ 24.2×)

TABLE II: **NTU RGB+D 60 transfer accuracy and performance benchmarks**. Noted is the top-1 validation accuracy using joints as the only modality. Max mem. is the maximum allocated memory on GPU during inference noted in megabytes. Max. mem, FLOPs, and throughput on CPU account for one new prediction with batch size one while throughput on GPU uses the largest fitting power of two as batch size. Parentheses indicate the improvement / deterioration relative to the original model.

consecutive layers (though temporal shrinkage is not a concern given the long input clips).

Another common design choice, which has a significant impact in on the performance of Continual Inference Networks, is the utilization of temporal stride larger than one. For regular networks, this has the benefit of reducing the computational complexity per clip prediction. In Continual Inference Networks, however, it reduces the prediction rate, and actually increases the complexity per prediction (see Section III-C). In the continual case, it would thus be computationally beneficial to reduce the stride of all layers to one. However, this results in a stride-inflicted *model-shift*.

Thus far, the *model-shift* inflicted by padding removal and stride reduction, as well as how to best perform the conversion from a regular CNN to a Continual CNN in such cases has not been studied. In this set of experiments, we explore strategies on how to best convert and fine-tune regular networks to achieve good frame-by-frame performance. We use a standard ST-GCN [15] trained on joints only as our starting-point, and explore the accuracy achieved by:

1) Converting to from regular network with equal padding and stride four ($\text{Reg}_{s=4}^{p=\text{eq}}$) to a Continual Inference Network, where zero-padding is omitted ($\text{Co}_{s=4}^{p=0}$).
2) Reducing the network stride to one without fine-tuning ($\text{Co}_{s=1}^{p=0}$).
3) Fine-tuning the $\text{Co}_{s=1}^{p=0}$ network (= Co$*$).
4) Fine-tuning a conversion-optimal regular network which has no zero-padding and a stride of one ($\text{Reg}_{s=1}^{p=0}$).
5) Converting from $\text{Reg}_{s=1}^{p=0}$ to Continual (= Co$*$).

As seen in Table I, the direct transfer of weights was found to have a modest negative impact on the accuracy (by -0.3%) due the removal of zero-padding. This is considerably less than was found in [34]. Our conjecture is that the smaller amount of zeros relative to clip size used in skeleton-based recognition (8 zeros per 300 frames or 2.67%) compared to video-based recognition (e.g., 2 zeros per 16 frames or or 12.5%) makes the removal of zero-padding less detrimental since zeros contribute relatively less to the downstream features. Lowering the stride to one and removing zero-padding reduced accuracy by a substantial amount but allowed the Continual Inference Network to operate at much lower FLOPs. This accuracy drop is alleviated equally effectively by either (a) initializing the $\text{Co}_{s=1}^{p=0}$ with standard weights and fine-tuning in the continual regime or (b) first fine-tuning the conversion-optimal regular network ($\text{Reg}_{s=1}^{p=0}$) and subsequently converting to a Continual Inference Network, though the latter had lower training times in practice. We fine-tuned the networks using the settings described in Section V-B. As visualised in Fig. 6, we found 20 epochs of fine-tuning using the settings described in Section V-B recover accuracy on NTU RGB+D 60 with additional training yielding only marginal differences. Following this approach the (padding zero, stride one) optimized Continual ST-GCN (*Co*ST-GCN*) achieves a similar prediction accuracy while reducing the computational complexity by a factor $107.7\times$ relative to original ST-GCN!

D. Conversion of Attention Architectures

As we explored in Section V-C, the ST-GCN network architecture can easily be modified and fine-tuned to achieve high accuracy for frame-by-frame predictions with exceptionally low computational complexity. A natural follow-up question is whether this conversion is equally successful for more complicated spatio-temporal architectures that employ attention mechanisms. To investigate this, we conduct a similar transfer for two recent ST-GCN variants, the Adaptive GCN (AGCN) [16] and the Spatial Transformer Network (S-TR) [32]. While S-TR is easily converted to a Continual Inference Network (*Co*S-TR) by replacing convolutions, residuals and pooling operators with Continual ones, the AGCN requires additional care. In the original version of AGCN, the vertex attention matrix $\mathbf{C}_p$ (see Eq. (7)) is computed from the global representations in the layer over all time-steps. Since this operation would be acausal in the context of a Continual Inference Network, we restrict it to utilize only the frame-specific subset of features. As a fine-tuning strategy, we first make the conversion from regular network to a conversion-

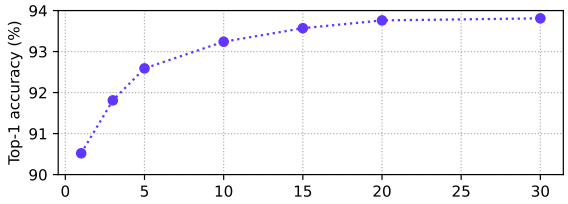


Fig. 6: **Fine-tuning epochs** and associated top-1 accuracy on NTU RGB+D 60 X-View for a transfer from a pre-trained ST-GCN with zero-padding and accumulated stride of four to an equivalent (*Co*)ST-GCN* with no zero-padding and stride one.

optimal network, and subsequently convert and evaluate the continual version.

Our results are presented in Table II. Here we see that all three architectures can be successfully converted to continual versions. The fine-tuned conversion-optimal models (marked by *) generally exhibit a higher computational complexity than their source models due to their stride decrease. While the ST-GCN* attained increased performance by lowering stride, AGCN* and S-TR* suffer slight accuracy deterioration. This may be due to smaller receptive fields of their attention mechanisms, which likely benefit from observing a larger context. Unlike the transfer from the original models with padding and stride four to continual models, the continual models with weights from ST-GCN*, AGCN*, and S-TR*, i.e. *Co*ST-GCN*, *Co*AGCN*, and *Co*S-TR* attain the exact same accuracy as their source models on both the X-Sub and X-View benchmarks, with two orders of magnitude less FLOPs per prediction during online inference.

E. Speed and Memory

Diving deeper into the differences between regular and continual networks, we conduct throughput benchmarks on a MacBook Pro 16" with a 2.6 GHz 6-Core Intel Core i7 CPU and a NVIDIA RTX 2080 Ti GPU. Here, we measure the prediction-time as the time it takes to transfer an input of batch size one from CPU to GPU (if applicable), perform inference, and transfer the results back to CPU again. On CPU, a batch size of one is used, while for GPU, the largest fitting power of two is employed (i.e. {128, 64, 256, 256} for the {Reg, Reg*, *Co*, and *Co**} models). We measure the maximum allocated memory during inference on GPU for batch size one.

As seen in Table II, the change in speed relative to the original models follow a similar trend to those seen for FLOPs. The non-continual stride one variants (denoted by *) exhibit roughly half the speed of the original models, while the continual models enjoy more than a magnitude speed up on both CPU and GPU. As expected, the continual stride one models (*Co**) attain the largest inference throughput. These relative speed-ups are lower than the relative FLOPs reductions due to the read/writes of internal intermediary features in the Continual Convolutions since these are not accounted for by the FLOPs metric while still adding to the runtime. This gap could be reduced on hardware with in- or near-memory computing.

	Model	S.	Accuracy (%) X-Sub	Accuracy (%) X-View	FLOPs (G)
Clip	SGN [27]	1	89.4	94.5	-
	MS-G3D [21]	1	89.4	95.0	-
		2	**91.5**	96.2	-
	ST-TR [32]	1	89.2	95.8	-
		2	90.3	***96.3***	-
	MS-AAGCN [17]	4	90.0	96.2	-
	Hyper-GNN [19]	3	89.5	95.7	-
	FGCN [20]	4	90.2	***96.3***	-
	DGNN [18]	4	89.9	96.1	126.80
	AS-GCN [40]	1	86.8	94.2	27.00
	AGC-LSTM [41]	2	89.2	95.0	54.40
	ShiftGCN [24]	1	87.8	95.1	2.50
		2	89.7	96.0	5.00
		4	***90.7***	**96.5**	10.00
	ShiftGCN++ [26]	1	87.9	94.8	**0.40**
		2	89.7	95.7	***0.80***
		4	90.5	***96.3***	1.70
	ST-GCN†	1	86.0	93.4	16.73
		2	88.1	94.9	33.46
	AGCN†	1	86.4	94.3	18.69
		2	88.3	95.3	37.38
	S-TR†	1	86.8	93.8	16.20
		2	89.1	95.3	32.40
Frame	Deep-LSTM [29]	1	60.7	67.3	-
	VA-LSTM [13]	1	79.2	87.7	-
	*Co*ST-GCN (ours)	1	86.0	93.4	0.27
		2	88.1	94.8	0.54
	*Co*ST-GCN* (ours)	1	86.3	93.8	***0.16***
		2	88.3	***95.0***	0.32
	*Co*AGCN (ours)	1	86.4	94.2	0.30
		2	88.2	**95.3**	0.60
	*Co*AGCN* (ours)	1	84.1	92.6	0.22
		2	86.0	93.1	0.44
	*Co*S-TR (ours)	1	86.5	93.5	0.17
		2	**88.8**	**95.3**	0.34
	*Co*S-TR* (ours)	1	86.3	92.4	**0.15**
		2	***88.9***	94.8	0.30

TABLE III: **NTU RGB+D 60** comparison with recent methods, grouped by clip- and frame-based inference. Noted are the number of streams (S.), top-1 validation accuracy, and FLOPs per prediction. †Results for our implementation. Highlights indicate **best**, ***next-best*** and pareto-optimal results.

Considering the maximum allocated memory at inference, we find that the continual models reduce memory by 20-52%. While the Continual Convolution and -Pooling layers do add some internal state that adds to the memory consumption, the intermediary features that are passed between network layers are much smaller, i.e. one frame instead of 75 to 300 frames.

F. Comparison with Prior Works

Most current state-of-the-art methods for skeleton-based action recognition are not able to efficiently perform frame-by-frame predictions in the online setting, since they are constrained to operate on whole skeleton-sequences. Some RNN-based methods, e.g. Deep-LSTM [29] and VA-LSTM [13], can be used for redundancy-free frame-wise predictions, but

	Model	S.	Accuracy (%) X-Sub	Accuracy (%) X-Set	FLOPs (G)
Clip	Part-Aware LSTM [42]	1	25.5	26.3	-
	ST-LSTM [10]	1	55.7	57.9	-
	TSRJI [43]	1	67.9	62.8	-
	SGN [27]	1	79.2	81.5	-
	MS-G3D [21]	2	**86.9**	**88.4**	-
	FGCN [20]	4	85.4	87.4	-
	ShiftGCN [24]	1	80.9	83.2	2.50
		2	85.3	86.6	5.00
		4	***85.9***	***87.6***	10.00
	ShiftGCN++ [26]	1	80.5	83.0	**0.40**
		2	84.9	86.2	***0.80***
		4	85.6	87.2	1.70
	ST-GCN†	1	79.0	80.7	16.73
		2	83.7	85.1	33.46
	AGCN†	1	79.7	80.7	18.69
		2	84.0	85.4	37.38
	S-TR†	1	80.2	81.8	16.20
		2	84.8	86.2	32.40
Frame	*Co*ST-GCN (ours)	1	78.9	80.7	0.27
		2	83.7	85.1	0.54
	*Co*ST-GCN* (ours)	1	79.4	81.6	***0.16***
		2	84.0	85.5	0.32
	*Co*AGCN (ours)	1	79.6	80.7	0.30
		2	84.0	85.3	0.60
	*Co*AGCN* (ours)	1	77.3	79.1	0.22
		2	80.4	82.0	0.44
	*Co*S-TR (ours)	1	80.1	81.7	0.17
		2	***84.8***	**86.1**	0.34
	*Co*S-TR* (ours)	1	79.7	81.7	**0.15**
		2	**84.8**	***86.1***	0.30

TABLE IV: **NTU RGB+D 120** comparison with recent methods, grouped by clip- and frame-based inference. Noted are the number of streams (S.), top-1 validation accuracy, and FLOPs per prediction. †Results for our implementation. Highlights indicate **best**, ***next-best*** and pareto-optimal results.

	Model	S.	Accuracy (%) Top-1	Accuracy (%) Top-5	FLOPs (G)
Clip	Feature Enc. [15, 44]	1	14.9	25.8	-
	Deep LSTM [11, 15]	1	16.4	35.3	-
	TCN [4, 15]	1	20.3	40.0	-
	AS-GCN [40]	1	34.8	56.5	-
	ST-GR [45]	1	33.6	56.1	-
	DGNN [18]	4	***36.9***	59.6	-
	MS-G3D [21]	2	**38.0**	***60.9***	-
	MS-AAGCN [17]	4	37.8	**61.0**	-
	Hyper-GNN [19]	3	37.1	60.0	-
	ST-GCN†	1	33.4	56.1	***12.04***
		2	34.4	57.5	24.09
	AGCN†	1	35.0	57.5	13.45
		2	***36.9***	59.6	26.91
	S-TR†	1	32.0	54.9	**11.62**
		2	34.7	57.9	23.24
Frame	*Co*ST-GCN (ours)	1	31.8	54.6	0.16
		2	33.1	56.1	0.32
	*Co*ST-GCN* (ours)	1	30.2	52.4	**0.11**
		2	32.2	54.5	0.22
	*Co*AGCN (ours)	1	33.0	55.5	0.18
		2	**35.0**	**57.3**	0.36
	*Co*AGCN* (ours)	1	23.3	44.3	***0.12***
		2	27.5	49.1	0.25
	*Co*S-TR (ours)	1	29.7	52.6	0.16
		2	***32.7***	***55.6***	0.31
	*Co*S-TR* (ours)	1	27.4	49.7	**0.11**
		2	29.9	52.7	0.22

TABLE V: **Kinetics Skeleton 400** comparison with recent methods, grouped by clip- and frame-based inference. Noted are the number of streams (S.), top-1 and top-5 validation accuracy, and FLOPs per prediction. †Results for our implementation. Highlights indicate **best**, ***next-best*** and pareto-optimal results.

their reported accuracy has been sub-par relative to newer methods that sprung from ST-GCN. The recently proposed AGC-LSTM [41] does report results on-par with CNN-based methods, and might also be able to provide redundancy-free frame-wise results, but we cannot validate this due to the lack of publicly available source code and details in the published paper. While ShiftGCN and ShiftGCN++ offer impressively low FLOPs, it should be noted that the shift operation, which is a significant part of their operational load, is not accounted for by the FLOPs metric. Due to the non-causal nature of the temporal shift operation in ShiftGCN and ShiftGCN++, they cannot be transformed into Continual Inference Networks in their current form, though a *Continual* Shift operation could plausibly be devised.

Many works have shown that the inclusion of multiple modalities leads to increased accuracy [15, 16, 18, 21, 24]. In our context, these modalities amount to *joints*, which are the original coordinates of the body joints, and *bones*, which are the differences between connected joints. Additional *joint motion* and *bone motion* modalities can be retrieved by computing the differences between adjacent frames in time for the joint and bone streams respectively. Models are trained individually on each stream and combined by adding their softmax outputs prior to prediction.

We evaluate and compare our proposed continual models, *Co*ST-GCN, *Co*AGCN, *Co*S-TR, with prior works on the NTU RGB+D 60, NTU RGB+D 120, and Kinetics Skeleton 400 datasets as presented in Table III, Table IV, and Table V.

The *Co*ST-GCN and *Co*S-TR models transfer well across all datasets both with (*) and without padding and stride modifications. For *Co*AGCN, we find that the change to stride one deteriorates accuracy. We surmise that the attention matrix in Eq. (7) may need a larger receptive field (basing the attention on more nodes as in AGCN) to provide beneficial adaptations; a per-step change in attention might provide more noise than clarity in middle and late network layers. As found in prior works, the multi-stream approach with ensemble predictions gives a meaningful boost in accuracy across all experiment.

The Continual Skeleton models provide competitive accuracy at multiple orders of magnitude reduction of FLOPs per prediction in the online setting compared to the original non-continual models. While none of our results beat prior state-of-the-art accuracy in absolute terms, this was never the intent with the method. Rather, we have successfully shown that online inference can be greatly accelerated for models in the ST-GCN family with state-of-the-art accuracy/complexity trade-offs to follow. For instance, our one and two-stream *Co*S-

TR* achieve pareto optimal results on all subsets of the NTU RGB+D 60 and NTU RGB+D 120 datasets meaning that no other model improves on either accuracy and FLOPs without reducing the other. Pareto-optimal models have been highlighted in Tables III, IV, and V accordingly. Our approach may be used similarly to accelerate other architectures for skeleton-based human action recognition with temporal convolutions.

VI. Conclusion

In this paper, we proposed *Continual* Spatio-Temporal Graph Convolutional Networks, an architectural enhancement for skeleton-based human action recognition methods, which augments prior methods with the ability to perform predictions frame-by-frame during online inference while attaining weight compatibility for batch inference. We re-implement and benchmark three prominent methods, the ST-GCN, AGCN, and S-TR, as novel Continual Inference Networks, *Co*ST-GCN, *Co*AGCN, and *Co*S-TR, and propose architectural modifications to maximize their frame-by-frame inference speed. Through experiments on three widely used human skeleton datasets, NTU RGB+D 60, NTU RGB+D 120, and Kinetics Skeleton 400, we show up to 26× on-hardware speedups, 109× reduction in FLOPs per prediction, and 52% reduction in maximum memory allocated memory during online inference with similar accuracy to those of the original networks. Our proposed architectural modifications are generic in nature and can be used for many methods in skeleton-based action recognition. It is our hope, that this innovation will make skeleton-based action recognition a viable option for online recognition systems on recourse-constrained devices and systems with real-time requirements.

Acknowledgment

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] F. Han, B. Reily, W. Hoff, and H. Zhang, "Space-time representation of people based on 3D skeletal data: A review," *Computer Vision and Image Understanding*, vol. 158, pp. 85–105, 2017.

[2] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.

[3] H. Liu, J. Tu, and M. Liu, "Two-stream 3D convolutional neural network for skeleton-based action recognition," *arXiv preprint arXiv:1705.08106*, 2017.

[4] T. S. Kim and A. Reiter, "Interpretable 3D human action analysis with temporal convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 1623–1631.

[5] Q. Ke, M. Bennamoun, S. An, F. Sohel, and F. Boussaid, "A new representation of skeleton sequences for 3D action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3288–3297.

[6] M. Liu, H. Liu, and C. Chen, "Enhanced skeleton visualization for view invariant human action recognition," *Pattern Recognition*, vol. 68, pp. 346–362, 2017.

[7] B. Li, Y. Dai, X. Cheng, H. Chen, Y. Lin, and M. He, "Skeleton based action recognition using translation-scale invariant image mapping and multi-scale deep CNN," in *IEEE International Conference on Multimedia & Expo Workshops*, 2017, pp. 601–604.

[8] C. Li, Q. Zhong, D. Xie, and S. Pu, "Skeleton-based action recognition with convolutional neural networks," in *IEEE International Conference on Multimedia & Expo Workshops*, 2017, pp. 597–600.

[9] Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1110–1118.

[10] J. Liu, A. Shahroudy, D. Xu, and G. Wang, "Spatio-temporal LSTM with trust gates for 3D human action recognition," in *European Conference on Computer Vision*. Springer, 2016, pp. 816–833.

[11] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "NTU RGB+D: A large scale dataset for 3D human activity analysis," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1010–1019.

[12] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu, "An end-to-end spatio-temporal attention model for human action recognition from skeleton data," in *AAAI Conference on Artificial Intelligence*, 2017, pp. 4263–4270.

[13] P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue, and N. Zheng, "View adaptive recurrent neural networks for high performance human action recognition from skeleton data," in *IEEE International Conference on Computer Vision*, 2017, pp. 2117–2126.

[14] L. Li, W. Zheng, Z. Zhang, Y. Huang, and L. Wang, "Skeleton-based relational modeling for action recognition," *arXiv preprint arXiv:1805.02556*, vol. 1, no. 2, p. 3, 2018.

[15] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *AAAI Conference on Artificial Intelligence*, 2018, pp. 7444–7452.

[16] L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Two-stream adaptive graph convolutional networks for skeleton-based action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 026–12 035.

[17] ——, "Skeleton-based action recognition with multi-stream adaptive graph convolutional networks," *IEEE Transactions on Image Processing*, vol. 29, pp. 9532–9545, 2020.

[18] ——, "Skeleton-based action recognition with directed graph neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7912–7921.

[19] X. Hao, J. Li, Y. Guo, T. Jiang, and M. Yu, "Hypergraph neural network for skeleton-based action recognition," *IEEE Transactions on Image Processing*, vol. 30, pp. 2263–2275, 2021.

[20] H. Yang, D. Yan, L. Zhang, Y. Sun, D. Li, and S. J. Maybank, "Feedback graph convolutional network for skeleton-based action recognition," *IEEE Transactions on Image Processing*, vol. 31, pp. 164–175, 2021.

[21] Z. Liu, H. Zhang, Z. Chen, Z. Wang, and W. Ouyang, "Disentangling and unifying graph convolutions for skeleton-based action recognition," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 140–149, 2020.

[22] W. Peng, X. Hong, H. Chen, and G. Zhao, "Learning graph convolutional network for skeleton-based human action recognition by neural searching." in *AAAI Conference on Artificial Intelligence*, 2020, pp. 2669–2676.

[23] 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)*, 2021, pp. 3220–3224.

[24] 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.
[25] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *International Conference on Learning Representations*, 2017.
[26] K. Cheng, Y. Zhang, X. He, J. Cheng, and H. Lu, "Extremely lightweight skeleton-based action recognition with shiftgcn++," *IEEE Transactions on Image Processing*, vol. 30, pp. 7333–7348, 2021.
[27] 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.
[28] 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.
[29] 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.
[30] J. Liu, A. Shahroudy, M. Perez, G. Wang, L.-Y. Duan, and A. C. Kot, "Ntu rgb+d 120: A large-scale benchmark for 3d human activity understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.
[31] 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.
[32] C. Plizzari, M. Cannici, and M. Matteucci, "Skeleton-based action recognition via spatial and temporal transformer networks," *Computer Vision and Image Understanding*, vol. 208, p. 103219, 2021.
[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017, pp. 5998–6008.
[34] L. Hedegaard and A. Iosifidis, "Continual 3d convolutional neural networks for real-time processing of videos," *preprint, arXiv:2106.00050*, pp. 1–12, 2021.
[35] L. Hedegaard, A. Bakhtiarnia, and A. Iosifidis, "Continual Transformers: Redundancy-Free Attention for Online Inference," *preprint, arXiv:2201.06268*, 2022.
[36] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NeurIPS Workshop*, 2017.
[37] L. Hedegaard, "Ride the lightning," *GitHub. Note: https://github.com/LukasHedegaard/ride*, 2021.
[38] L. N. Smith and N. Topin, "Super-convergence: very fast training of neural networks using large learning rates," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, vol. 11006, International Society for Optics and Photonics. SPIE, 2019, pp. 369 – 386.
[39] P. Goyal, P. Dollár, R. B. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," *preprint, arXiv:1706.02677*, 2017.
[40] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, and Q. Tian, "Actional-structural graph convolutional networks for skeleton-based action recognition," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3590–3598, 2019.
[41] C. Si, W. Chen, W. Wang, L. Wang, and T. Tan, "An attention enhanced graph convolutional lstm network for skeleton-based action recognition," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1227–1236, 2019.
[42] J. Liu, A. Shahroudy, M. Perez, G. Wang, L.-Y. Duan, and A. C. Kot, "Ntu rgb+ d 120: A large-scale benchmark for 3d human activity understanding," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 10, pp. 2684–2701, 2019.
[43] C. Caetano, F. Brémond, and W. R. Schwartz, "Skeleton image representation for 3d action recognition based on tree structure and reference joints," in *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, 2019, pp. 16–23.
[44] B. Fernando, E. Gavves, M. José Oramas, A. Ghodrati, and T. Tuytelaars, "Modeling video evolution for action recognition," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5378–5387.
[45] B. Li, X. Li, Z. Zhang, and F. Wu, "Spatio-temporal graph routing for skeleton-based action recognition," in *AAAI*, 2019.

Lukas Hedegaard is a PhD candidate at Aarhus University, Denmark. He received his M.Sc. degree in Computer Engineering in 2019 and B.Eng. degree in Electronics in 2017 at Aarhus University, specialising in signal processing and machine learning. With a common theme of efficient deep learning, his research endeavours span from online inference acceleration and human activity recognition to transfer learning and domain adaptation.

Negar Heidari is a Postdoctoral researcher at Aarhus University, Denmark. She completed her PhD in Signal Processing and Machine Learning at the Department of Electrical and Computer Engineering, Aarhus University in 2022. Her current research interests include machine learning, deep learning and computer vision with a focus on computational efficiency.

Alexandros Iosifidis (SM'16) is a Professor at Aarhus University, Denmark. He serves as Associate Editor in Chief for Neurocomputing (for Neural Networks research area), as an Area Editor for Signal Processing: Image Communication, and as an Associate Editor for IEEE Transactions on Neural Networks and Learning Systems. He was an Area Chair for IEEE ICIP 2018-2022 and EUSIPCO 2019,2021, and Publicity co-Chair of IEEE ICME 2021. He was the recipient of the EURASIP Early Career Award 2021 for contributions to statistical machine learning and artificial neural networks. His research interests focus on neural networks and statistical machine learning finding applications in computer vision, financial modelling and graph analysis problems.

8.5 Structured Pruning Adapters

The appended paper follows.

Structured Pruning Adapters

Lukas Hedegaard[1*] Aman Alok[2] Juby Jose[2] Alexandros Iosifidis[1]
[1]Aarhus University [2]Cactus Communications
{lhm,ai}@ece.au.dk {aman.alok, juby.jose}@cactusglobal.com

Abstract

We propose Structured Pruning Adapters (SPAs), a family of compressing, task-switching network adapters, that accelerate and specialize networks using tiny parameter sets. Specifically, we propose a channel- and a block-based SPA and evaluate them with a suite of pruning methods on both computer vision and natural language processing benchmarks. Compared to regular structured pruning with fine-tuning, our channel-SPA improves accuracy by 6.9% on average while using half the parameters at 90% pruned weights. Alternatively, it can learn adaptations with 17× fewer parameters at 70% pruning with 1.6% lower accuracy. Similarly, our block-SPA requires far fewer parameters than pruning with fine-tuning. Our experimental code and Python library of adapters are available at github.com/lukashedegaard/structured-pruning-adapters.

1. Introduction

Fine-tuning is an established approach to parameter-based transfer learning from a source model pre-trained on a large dataset to a target task with limited training data. However, the resulting model retains the same parameter count and computational characteristics as the source model, even when solving a considerably simpler task. A group of *fine-pruning* methods [21, 23, 30, 40, 42, 45] have combined pruning with fine-tuning to reduce model size while learning parameters for the target task. Generally, this results in compressed models that retain performance down to at least half the relative weight density and which may be better suited for resource-constrained deployments, such as mobile devices, robotics applications, and settings necessitating low-latency predictions.

Meanwhile, Adapters [36, 37] have emerged as a viable alternative to fine-tuning for multi-domain deep neural networks (DNNs), where a single source network is specialized and sequentially used for multiple tasks. Instead of continuing training of the source DNN weights directly, Adapters introduce parameter-efficient layer add-ons, which are trained instead. As these add-ons are much more compact than the source DNN weights, they can be transmitted and stored at low cost. However, prior work has largely ignored the computational efficiency aspects of Adapters, which either increase the computational complexity of the network [11, 17, 24, 28, 33, 48] or leave it unaltered, at best, by utilizing structures that can be fused with the original weights [18, 36, 37].

In general, a deployed model must strike a balance between predictive performance, storage requirements, inference speed, and flexibility. While the combination of pruning and fine-tuning can produce compressed models with good performance at an order of magnitude fewer parameters compared to the source model, *Structured Pruning Adapters (SPAs)* can improve upon this by another order of magnitude for task-switching networks. Here, we present SPAs for both channel-based [23, 30, 42, 45] and block-based [21, 40] pruning.

For channel-based pruning, we propose the *Structured Pruning Low-rank Adapter* (SPLoRA) and compare its performance and parameter count to pruning with fine-tuning in weight-based transfer learning from ResNet weights pre-trained on ILSVRC 2012 [39] to the image classification benchmarks CIFAR-10 [19], Oxford Flowers 102 [32], and Cats and Dogs [2]. Considering four different pruning methods, we find that SPLoRA not only reduces parameter requirements per task massively, but also retains predictive accuracy better than fine-tuning under aggressive pruning. For block-based pruning, we propose the *Structured Pruning Low-rank Parameterized Hypercomplex Multiplication Adapter* (SPLoPA) and examine the transfer of BERT-base weights to SQuAD [34] using four pruning methods. Compared to fine-pruning SPLoPA requires only a fraction of learned weights at a modest decrease in F1.

In the remainder of this paper, we describe related work on Adapters (Sec. 2.1) and Pruning (Sec. 2.2), a general framework for SPAs (Sec. 3), the SPLoRA (Sec. 3.1) and SPLoPA (Sec. 3.2) adapters, experimental comparisons with fine-pruning (Sec. 4), and conclusions (Sec. 5).

*Work initiated during research visit at Cactus Communications

2. Related Work

2.1. Adapter methods

When multiple specialized versions of a network are deployed on the same device and storage requirements are strict, Adapters [36] provide a low-parameter-count alternative to fine-tuning. Instead of deploying multiple sets of full network weights, a single set of full weights can be deployed alongside multiple adapter weights, which augment the main network. For Convolutional Neural Networks (CNNs), point-wise convolutions can be introduced in series [36] or parallel [37] with a residual connection to adapt fixed source weights to new tasks. For Transformer-based networks, prior work explored bottleneck projections with [48] and without [18] low-dimensional non-linearity in parallel with the fixed dense layers of the original network. Prefix-tuning [24], which learns task-specific prompt tokens to adapt the network, may be considered a parallel adapter as well [11]. Adapter blocks can also be interspersed in series with existing layers [17, 28, 33]. He at al. [11] proposed a combination of the above. Finally, several works [33, 38, 41] explored the use of multi-task adapters.

While the above-described methods succeed in learning parameter-efficient network add-ons with very small storage requirements, these adapters often incur an additional computational cost beyond the original network. Considering that the adapted target tasks are often simpler than the source task, it is reasonable to assume that a derived network adaptation can be learned, which reduces computational complexity as well.

2.2. Pruning methods

Multiple approaches have been proposed to reduce the compute and memory footprint of neural networks. Knowledge distillation approaches[1] [5, 15] utilize a large network as a teacher for a smaller network, which has more desirable memory and computational characteristics. Efficient architectures [3, 16, 43] define and optimize expressive yet efficient architectural blocks from random initialization under a multi-metric optimization goal. Low-rank factorizations [7, 44] approximate large tensor weights by factorizing them into multiple lower-rank weights. Continual Inference Networks [12, 13] reuse the network weights of prior DNNs with a temporal component and accelerate them for online stream processing via optimized computational sequences and appropriate intra-layer caching. Quantization approaches [6, 25] reduce model size and run-time costs via low-resolution numerical representations of network weights. Finally, Pruning methods [4, 9, 22] entirely remove unnecessary network weight from a pre-trained model. While all of these are interesting research avenues both in isolation and combination, we focus on pruning-methods hereafter.

DNNs can be pruned at multiple levels: *Unstructured* pruning of individual weights results in sparse weight matrices which reduce parameter count but require specialized hardware to improve inference characteristics due to the performance disadvantages of sparse matrix multiplication kernels on graphical processingn units (GPUs). On the other hand, *structured* pruning approaches, such as the pruning of entire channels [45] or blocks [21] of networks weights, generally provide inference speedup across computational devices.

Many studies have proposed criteria on *what* to prune. Early methods [10, 22] proposed the use of second-order Taylor expansion of the loss Hessian for weight selection. As computing the inverse of the Hessian may be computationally intractable, a more recent approach uses a first-order Taylor approximation of the loss change due to the pruning of units [30]. Similarly, the gradient of a weight with respect to the loss can be used for pruning selection [26, 40, 42]. We employ Sun et al.'s method [42], which considers the magnitude of the gradient, as a baseline pruning method in Sec. 4.1. L_0 Regularization pruning [27], Movement pruning [40], and Soft Movement pruning [40] are employed in Sec. 4.2. Among the simplest approaches is the use of weight magnitudes in pruning selection. In our experimental comparisons, we employ Magnitude Pruning [9], which uses a simple average of weight magnitudes, and Weight Pruning [23], which uses the L_p-norm of weights, for channel selection. Yeom et al. [45] proposed an explainability-inspired approach, computing the relevance of each network component by means of Layer-wise Relevance Pruning (LRP) [1, 31]. We use this method in Sec. 4.1.

For all the above methods assigning pruning scores, another consideration is whether to rank and select structural units locally within a layer (keeping pruning evenly spread throughout the network) or globally, with a contest among all network layers. We utilize the latter in Sec. 4.1 and the former in Sec. 4.2.

Beyond the pruning criterion, multiple studies proposed *when* to prune. One popular approach [26, 30, 45] is to use an iterative pruning and fine-tuning schedule, masking a predefined fraction of units at a time. Alternatively, Automated Gradual Pruning [47] allows all weights to be updated throughout the pruning schedule, enabling prior masking choices to be altered. We use the former in Sec. 4.1 and the latter in Sec. 4.2, alongside the cubic sparsity scheduler as employed in previous studies [40, 47].

[1] Knowledge distillation [15] using the unpruned model as the teacher has been found to help pruning methods [21, 40] retain accuracy better using fine-pruning. We expect this to be true for SPAs as well, but we leave the validation to future work as none of the considered fine-pruning baselines employed knowledge distillation.

2.3. Transfer pruning

Pruning is useful not only for compressing a model while retaining predictive performance, but also for transfer learning [21, 40, 45]. In fact, a task can be "learned" simply by selecting the appripriate subset of weights in a network [29, 35].

Consider a large (pre-trained) source model f_s and a set of T target tasks for which we desire specialized target models $f_t, t \in \{1..T\}$. Under the framework of transfer learning with pruning ("transfer pruning"), we can concurrently update and mask weights from a source model to benefit a target task t. Consider $g : \boldsymbol{W}_s \times \Delta\boldsymbol{W}_t \times \mathbb{M}_t \rightarrow \boldsymbol{W}_t$, a function that generates target model weights $\boldsymbol{W}_t$, given learned update weights $\Delta\boldsymbol{W}_t$, source weights $\boldsymbol{W}_s$, and a learned masking set $\mathbb{M}_t$ of retained weight indices. Given available source weights $\boldsymbol{W}_s$, every task-specific model f_t can be stored as the parameters $\Phi_t = \{\Delta\boldsymbol{W}_t, \mathbb{M}_t\}$.

Under *fine-pruning* [40], i.e., concurrent pruning and fine-tuning, the generation function degenerates into a direct assignment of weights, $\boldsymbol{W}_t := g(\boldsymbol{W}_s, \Delta\boldsymbol{W}_t, \mathbb{M}_t) = \Delta\boldsymbol{W}_t$, where update weights are learned based on a pruned subset, $\{\boldsymbol{W}_s^{(i)}, i \in \mathbb{M}_t\}$. Here, the parameters of the task-specific model are $\Phi_t = \{\boldsymbol{W}_t, \mathbb{M}_t\}$, and the size of the target weights is determined by the weight density $d \in (0, 1]$ and the size of source weights: $\|\boldsymbol{W}_t\|_0 = d\|\boldsymbol{W}_s\|_0$.

3. Structured Pruning Adapters

Although fine-pruning can successfully produce smaller target weights, i.e., $\|\boldsymbol{W}_t\|_0 < \|\boldsymbol{W}_s\|_0$, the set of weights for all tasks $\{\boldsymbol{W}_t\}$ may still consume an intractable amount of storage if many tasks T are involved and/or the average density $\bar{d}$ is large due to high predictive performance requirements. Instead, we seek to utilize adapters alongside pruning to produce an extremely compressed parameter set.

Consider the concurrent pruning and adaptation of a single source projection matrix $\boldsymbol{W}_s \in \mathbb{R}^{n\times m}$ with an index mask $\boldsymbol{M} \in \{0, 1\}^{n\times m}$ and an adapter function a. While applicable adapters have been extensively studied (see Sec. 2.1), we restrict ourselves to fusible parallel adapters to minimize the run-time of the resulting model. Accordingly, pruning adapters take the following basic form:

$$\boldsymbol{W}_t = (\boldsymbol{W}_s + a(\Delta\boldsymbol{W}_t)) \odot \boldsymbol{M}. \tag{1}$$

3.1. Structured Pruning Low-rank Adapter

Unlike unstructed methods, *structured* pruning methods remove groups of weights and increase computational efficiency. *Channel* pruning, in particular, maps a dense source matrix to a dense pruned matrix with computational improvements proportional to the number of removed parameters. A mask $\boldsymbol{M}$ in this case can be decomposed as row and column masks $\boldsymbol{m}_{\text{row}} \in \{0, 1\}^{n\times 1}$ and $\boldsymbol{m}_{\text{col}} \in \{0, 1\}^{m\times 1}$, respectively. Then, Eq. (1) can be expressed as follows:

$$\boldsymbol{W}_t = (\boldsymbol{W}_s + a(\Delta\boldsymbol{W}_t)) \odot \boldsymbol{m}_{\text{row}}\boldsymbol{m}_{\text{col}}^\top. \tag{2}$$

A simple realization of a fusible parallel adapter is the Low-rank Adapter (LoRA) [18]:

$$\boldsymbol{W}_t = \boldsymbol{W}_s + \boldsymbol{W}_{\text{down}}\boldsymbol{W}_{\text{up}}, \tag{3}$$

where $\boldsymbol{W}_{\text{down}} \in \mathbb{R}^{n\times r}$ and $\boldsymbol{W}_{\text{up}} \in \mathbb{R}^{r\times m}$ are adapter weights and r is the rank hyper-parameter. Following the derivation in Appendix A.1, we utilize Eq. (2) and Eq. (3) to define the **S**tructured **P**runing **Lo**w **R**ank **A**dapter (SPLoRA):

$$\boldsymbol{W}_t = \boldsymbol{W}_s \odot \boldsymbol{m}_{\text{row}}\boldsymbol{m}_{\text{col}}^\top + (\boldsymbol{W}_{\text{down}} \odot \boldsymbol{m}_{\text{row}}\boldsymbol{1}^\top)(\boldsymbol{W}_{\text{up}} \odot \boldsymbol{1}\boldsymbol{m}_{\text{col}}^\top). \tag{4}$$

In this form, we see that channel-pruning affects not only the source weights $\boldsymbol{W}_s$, but also the adapter parameters $\boldsymbol{W}_{\text{up}}$ and $\boldsymbol{W}_{\text{down}}$. This effect is illustrated in Fig. 1.

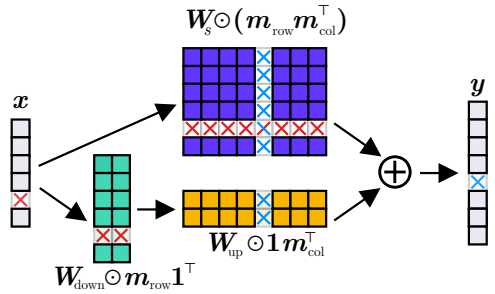


Figure 1. Structured Pruning Low-rank Adapter (SPLoRA). Pruning of in/out channels affects the adapter as well as source weights.

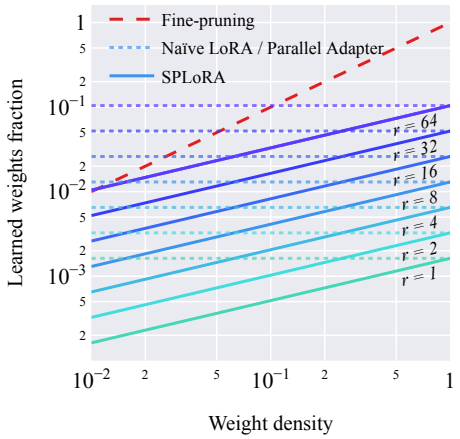


Figure 2. Learned weight fraction ($\|\Delta\boldsymbol{W}_t\|_0/\|\boldsymbol{W}_s\|_0$) versus weight density ($\|\boldsymbol{W}_t\|_0/\|\boldsymbol{W}_s\|_0$) for 768×3072 weights under channel-pruning.

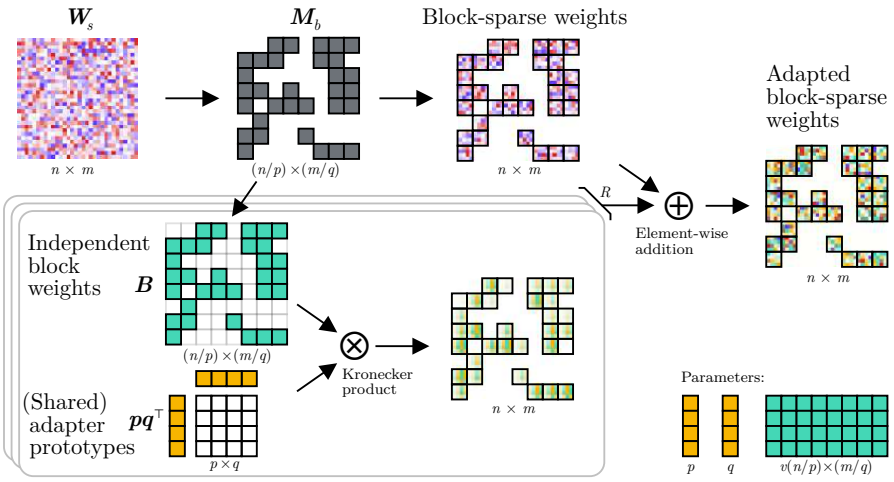


Figure 3. Structured Pruning Low-rank PHM Adapter (SPLoPA) visualized for $n = m = 32$, $p = q = 4$, and 50% density.

Adapters should generally be initialized to perform a near-zero mapping to avoid destabilizing gradients during initial training. Accordingly, we use the initialization $\boldsymbol{W}_{\text{up}}^{(i,j)} \sim \mathcal{U}(-10^{-4}, 10^{-4})$, although other near-zero initialization choices may be equally successful. Experiments validating this choice are presented in Sec. 4.1.

For a fine-pruned model weight $\boldsymbol{W}_t$, the updated parameter count is $\|\boldsymbol{m}_{\text{row}}\|_0 \|\boldsymbol{m}_{\text{col}}\|_0$. The corresponding SPLoRA has $r(\|\boldsymbol{m}_{\text{row}}\|_0 + \|\boldsymbol{m}_{\text{col}}\|_0)$ parameters. Any comparison of parameters depends on the weight retention fraction, weight matrix shape, and SPLoRA bottleneck dimension r. Figure 2 visualizes the parameters achieved by varying the hyper-parameters for a 768×3072 weight matrix. Here, we have depicted the learned parameter count of fine-pruning and SPLoRA alongside the naïve application of LoRA or, equivalently, the Parallel Adapter [11, 48] for reference. While naïve adapter usage requires far-fewer parameters than fine-tuning at high weight density, the parameter reduction achieved by fine-pruning can produce equally few parameters at low weight densities. SPLoRA combines the benefits of both approaches to produce parameter sets far more compact than those produced by either.

3.2. Structured Pruning Low-rank Parameterized Hypercomplex Multiplication Adapter

Another type of structured pruning is *block pruning*. Here, we consider blocks of weights of size $p \times q$, which are pruned in unison. Instead of masking each weight individually, we utilize a block mask $\boldsymbol{M}_b$ of shape $n/p \times m/q$. In this case, Eq. (1) can be expressed as follows:

$$\boldsymbol{W}_t = (\boldsymbol{W}_s + a(\Delta \boldsymbol{W}_t)) \odot (\boldsymbol{M}_b \otimes \boldsymbol{J}), \tag{5}$$

where $\otimes$ denotes the Kronecker product and $\boldsymbol{J} \in \{1\}^{p \times q}$. Row or column pruning can be viewed as special cases of block pruning in which block dimensions are $n \times 1$ or $1 \times m$, respectively.

To devise a pruning-sensitive adapter under block pruning, we follow a previous study [28] and draw on Parameterized Hypercomplex Multiplication (PHM) layers [46]. Conceptually, we consider the adaptation of each $p \times q$ block of weights separately as a weighted sum of R low-rank prototypes $\boldsymbol{P} = \boldsymbol{p}\boldsymbol{q}^\top \in \mathbb{R}^{p \times q}$, where $\boldsymbol{p} \in \mathbb{R}^{p \times r}$ and $\boldsymbol{q} \in \mathbb{R}^{q \times r}$. These prototypes can be shared throughout network, but their block weights $\boldsymbol{B} \in \mathbb{R}^{n/p \times m/q}$ are unique for each adapted weight matrix. Our resulting **S**tructured **P**runing **Lo**w-rank **P**arameterized Hypercomplex Multiplication **A**dapter (SPLoPA) is defined as follows:

$$\boldsymbol{W}_t = (\boldsymbol{W}_s \odot [\boldsymbol{M}_b \otimes \boldsymbol{J}]) + \sum_{i=1}^{R} (\boldsymbol{M}_b \odot \boldsymbol{B}_i) \otimes \boldsymbol{p}_i \boldsymbol{q}_i^\top. \tag{6}$$

Here, we use the initialization $\boldsymbol{B}^{(i,j)} \sim \mathcal{U}(-10^{-4}, 10^{-4})$ to achieve a near-zero behavior initially. Fig. 3 shows an example adaptation of block-sparse weights with SPLoPA.

For block-structured fine-pruning, the learned parameter count per layer is linear in the block size and number of retained blocks: $pq\|\boldsymbol{M}_b\|_0$. For SPLoPA, the hyper-parameter R linearly controls the memory complexity. Moreover, the complexity is reduced when more layers L share the prototypes among them: $N(p+q+\|\boldsymbol{M}_b\|_0/L)$. Figure 4 visualizes the number of learned parameters considering different rank choices and model densities.

3.3. Limitations

While each set of SPA weights has significantly fewer learned parameters than each set of fine-pruning weights, the superiority of SPAs is dependent on the deployment at hand. When only a single model is deployed, the two approaches are equivalent in terms of model size, considering

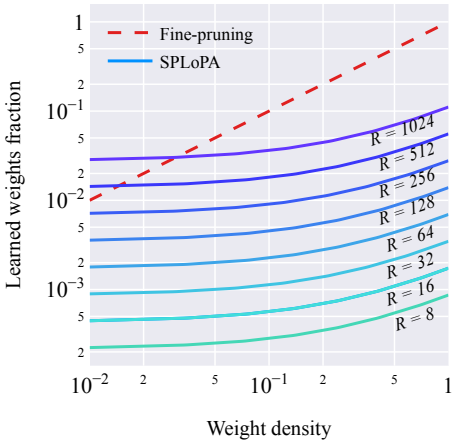


Figure 4. Learned weight fraction ($\|\Delta \boldsymbol{W}_t\|_0 / \|\boldsymbol{W}_s\|_0$) versus weight density ($\|\boldsymbol{W}_t\|_0 / \|\boldsymbol{W}_s\|_0$) for 768×3072 weights under block-pruning for $r = 1$. SPLoPA prototype sharing is not accounted for.

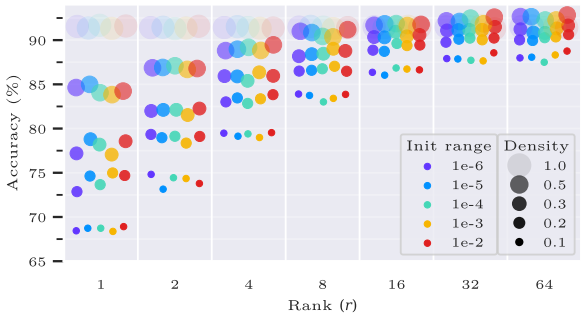


Figure 5. ResNet-18 accuracy on CIFAR-10 using SPLoRA of varying ranks (r) and adapter weight initialization ranges at different model densities (d). Point size $\propto d$ with alpha $\propto -d$.

that the fused parameter count of adapters is identical to that of the fine-pruned network. For T models deployed on one device, the adapter format saves space when $T > 1/\overline{d}$, assuming available source weights $\boldsymbol{W}_s$. Thus, the feasibility depends on the average density $\overline{d}$, with more aggressive pruning requiring more deployed models for SPA formats to consume less storage space.

3.4. Pruning of adapter-specific parameters

The design choices made for SPLoRA (Sec. 3.1) and SPLoPA (Sec. 3.2) let us utilize prior channel- and block-based pruning without modification by pruning fused weights, which are identical in form to the source weights, and subsequently pruning the adapter-weights with the same mask. The adapters themselves could also be subject to pruning [38]. For SPLoRA, the bottleneck contains r channels that could be pruned per layer to produce a heterogeneous set of layer adapters with different ranks per layer. For SPLoPA, both the number of prototypes (R) and prototype rank (r) could undergo similar pruning. However, such pruning adds an additional layer of complexity and hyper-parameters to the already complicated pruning methods, which may reduce the attractiveness of adapters. Moreover, the pruning of intra-adapter channels only reduces the parameter count and does not yield computational benefits. Therefore, we limit the pruning in subsequent experiments to the approach we believe has the best usability in practice; we refrain from pruning over internal adapter structures.

4. Experiments

We seek to compare structured pruning with fine-tuning (fine-pruning) to our proposed Structured Pruning Adapter methods. As both have identical acceleration benefits during inference, our experimental comparison focuses on prediction performance and the number of learned parameters (ΔParams). Our experiments are divided into seperate groups that asses SPAs for channel-based (Sec. 4.1) and block-based (Sec. 4.2) pruning.

4.1. Channel-based Pruning

Experimental setup In this set of experiments, we reuse and augment a previously reported setup [45] to perform transfer-pruning for ResNet models pretrained on ILSVRC 2012 [39] to the image classification benchmarks CIFAR-10 [19], Oxford Flowers 102 [32], and Cats and Dogs [2]. As no publicly available test split was available for the latter, we defined train-test splits and preprocessed data using DatasetOps [14] to match the 8005 training and 2023 test samples reported previously [45]. For transfer pruning, we first train the network without pruning for $\{30, 100, 100\}$ epochs and then perform incremental pruning at increments of 5% until 5% of weights remain in total; the incremental pruning is interspersed with $\{20, 50, 50\}$ epochs of training for the {CIFAR-10, Oxford Flowers 102, Cats and Dogs} datasets. Appendix A.2 presents an overview of training times. We employ the Strochastic Gradient Descent optimizer with a momentum of 0.9, weight decay of $5 \cdot 10^{-4}$, and learning rate of 0.01 at a batch size 256 or downscaled rates following the *linear scaling rule* [20] when GPU memory limitations must be accomodated. In each training block, we use a step learning rate reduction of $5\times$ after each quarter of epochs. The above setup is used for either fine-pruning, in which all model weights are updated, or adaptation and pruning, which freezes the original network weights and only trains the adapter weights, normalization, and prediction head.

SPLoRA initialization and rank choice To gauge the sensitivity of SPLoRA hyper-parameters and their effect on

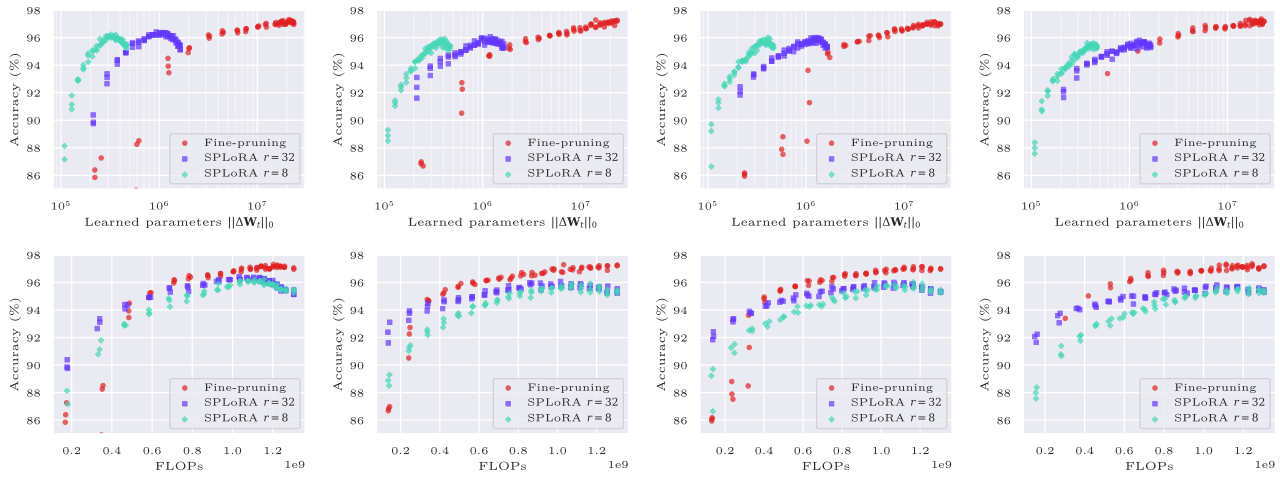
Table 1. Channel-based transfer-pruning from ResNet-50 pre-trained on ImageNet to Cats and Dogs, Oxford Flowers 102, and CIFAR-10. ΔParams and floating point operations (FLOPs) are shown for CIFAR-10. Mean ± standard deviation is shown for each metric. Changes specified relative to closest fine-pruning density with same pruning method. Best metric per pruning-method is highlighted with bold.

Pruning method	**Learning method**	**Density**	**ΔParams** (K)	**FLOPs** (M)	**CIFAR-10** Acc. (%)	**Oxford Flowers 102** Acc. (%)	**Cats & Dogs** Acc. (%)
Unpruned	Fine-tuning	100%	$23{,}520.8_{\pm 0.0}$	$1{,}304.7_{\pm 0.0}$	$\mathbf{97.10}_{\pm 0.12}$	$\mathbf{92.20}_{\pm 0.00}$	$\mathbf{99.30}_{\pm 0.02}$
	LoRA-$r32$	100%	$1{,}644.5_{\pm 0.0}$ (↓ 14.3×)	$1{,}304.7_{\pm 0.0}$	$95.32_{\pm 0.13}$ (−1.8)	$78.57_{\pm 5.93}$ (−13.6)	$98.60_{\pm 0.43}$ (−0.5)
	LoRA-$r8$	100%	$\mathbf{466.3}_{\pm 0.0}$ (↓ 50.4×)	$1{,}304.7_{\pm 0.0}$	$95.35_{\pm 0.10}$ (−1.8)	$80.96_{\pm 5.83}$ (−11.2)	$98.84_{\pm 0.52}$ (−0.46)
Weight [23]	Fine-pruning	30%	$4{,}427.1_{\pm 72.5}$	$785.4_{\pm 5.4}$	$\mathbf{96.38}_{\pm 0.12}$	$93.64_{\pm 2.15}$	$\mathbf{98.64}_{\pm 0.04}$
		10%	$599.1_{\pm 20.1}$	$352.1_{\pm 3.9}$	$87.23_{\pm 2.00}$	$72.83_{\pm 0.93}$	$95.42_{\pm 0.31}$
	SPLoRA-$r32$	30%	$618.3_{\pm 2.5}$ (↓ 7.2×)	$778.6_{\pm 2.9}$	$95.59_{\pm 0.22}$ (−0.8)	$\mathbf{94.57}_{\pm 0.58}$ (+0.9)	$98.43_{\pm 0.13}$ (−0.2)
		10%	$294.8_{\pm 0.4}$ (↓ 2.0×)	$\mathbf{335.4}_{\pm 7.3}$	$93.04_{\pm 0.37}$ (+5.8)	$89.36_{\pm 1.09}$ (+16.5)	$96.25_{\pm 0.99}$ (+0.8)
	SPLoRA-$r8$	30%	$210.0_{\pm 1.9}$ (↓ 21.1×)	$773.4_{\pm 2.1}$	$94.91_{\pm 0.24}$ (−1.5)	$92.13_{\pm 0.15}$ (−1.5)	$98.40_{\pm 0.12}$ (−0.2)
		10%	$\mathbf{128.9}_{\pm 0.1}$ (↓ 4.6×)	$338.9_{\pm 7.0}$	$91.24_{\pm 0.51}$ (+4.0)	$86.49_{\pm 0.74}$ (+13.7)	$96.07_{\pm 0.69}$ (+0.6)
Gradient [42]	Fine-pruning	30%	$3{,}719.8_{\pm 59.2}$	$571.9_{\pm 6.5}$	$\mathbf{95.95}_{\pm 0.09}$	$\mathbf{94.21}_{\pm 0.74}$	$\mathbf{98.22}_{\pm 0.00}$
		10%	$615.7_{\pm 4.3}$	$244.7_{\pm 3.3}$	$91.83_{\pm 1.17}$	$73.06_{\pm 0.70}$	$95.84_{\pm 0.20}$
	SPLoRA-$r32$	30%	$601.0_{\pm 0.4}$ (↓ 6.2×)	$564.6_{\pm 1.5}$	$94.91_{\pm 0.09}$ (−1.0)	$93.58_{\pm 0.43}$ (−0.6)	$98.17_{\pm 0.08}$ (−0.0)
		10%	$293.1_{\pm 0.4}$ (↓ 2.1×)	$\mathbf{244.0}_{\pm 1.9}$	$93.65_{\pm 0.36}$ (+1.8)	$91.35_{\pm 0.47}$ (+18.3)	$97.54_{\pm 0.17}$ (+1.7)
	SPLoRA-$r8$	30%	$205.2_{\pm 0.2}$ (↓ 18.1×)	$565.2_{\pm 4.1}$	$94.09_{\pm 0.28}$ (−1.9)	$91.60_{\pm 0.30}$ (−2.6)	$98.15_{\pm 0.07}$ (−0.1)
		10%	$\mathbf{128.3}_{\pm 0.0}$ (↓ 4.8×)	$245.4_{\pm 4.0}$	$91.25_{\pm 0.20}$ (−0.6)	$87.46_{\pm 0.71}$ (+14.4)	$97.19_{\pm 0.28}$ (+1.4)
Taylor [30]	Fine-pruning	30%	$3{,}392.8_{\pm 81.1}$	$559.9_{\pm 0.7}$	$\mathbf{95.71}_{\pm 0.02}$	$92.91_{\pm 0.56}$	$\mathbf{98.22}_{\pm 0.18}$
		10%	$576.8_{\pm 9.9}$	$236.9_{\pm 3.3}$	$88.07_{\pm 0.66}$	$65.67_{\pm 4.12}$	$95.30_{\pm 0.21}$
	SPLoRA-$r32$	30%	$599.7_{\pm 0.9}$ (↓ 5.7×)	$555.5_{\pm 6.7}$	$94.88_{\pm 0.21}$ (−0.8)	$\mathbf{93.41}_{\pm 0.06}$ (+0.5)	$97.84_{\pm 0.48}$ (−0.4)
		10%	$292.6_{\pm 0.5}$ (↓ 2.0×)	$\mathbf{242.0}_{\pm 1.5}$	$93.27_{\pm 0.12}$ (+5.2)	$91.30_{\pm 0.10}$ (+25.6)	$97.21_{\pm 0.10}$ (+1.9)
	SPLoRA-$r8$	30%	$205.3_{\pm 0.1}$ (↓ 16.5×)	$566.2_{\pm 10.9}$	$93.98_{\pm 0.24}$ (−1.7)	$91.51_{\pm 0.49}$ (−1.4)	$97.90_{\pm 0.13}$ (−0.3)
		10%	$\mathbf{128.4}_{\pm 0.1}$ (↓ 4.5×)	$243.2_{\pm 9.8}$	$91.22_{\pm 0.32}$ (+3.2)	$86.76_{\pm 0.42}$ (+21.1)	$96.83_{\pm 0.30}$ (+1.5)
LRP [45]	Fine-pruning	30%	$4{,}428.1_{\pm 20.6}$	$719.9_{\pm 0.7}$	$\mathbf{96.54}_{\pm 0.14}$	$\mathbf{95.37}_{\pm 0.08}$	$\mathbf{98.65}_{\pm 0.07}$
		10%	$599.0_{\pm -}$	$302.1_{\pm -}$	$93.39_{\pm -}$	$87.56_{\pm 2.75}$	–
	SPLoRA-$r32$	30%	$592.6_{\pm 0.9}$ (↓ 7.5×)	$585.5_{\pm 6.7}$	$94.85_{\pm 0.13}$ (−1.7)	$93.62_{\pm 0.38}$ (−1.8)	$98.09_{\pm 0.11}$ (−0.6)
		10%	$290.9_{\pm 0.2}$ (↓ 2.1×)	$\mathbf{270.4}_{\pm 6.4}$	$93.47_{\pm 0.36}$ (+0.1)	$91.22_{\pm 0.46}$ (+3.7)	$97.01_{\pm 0.27}$
	SPLoRA-$r8$	30%	$203.3_{\pm 0.5}$ (↓ 21.8×)	$591.1_{\pm 12.4}$	$93.53_{\pm 0.18}$ (−3.0)	$91.26_{\pm 0.19}$ (−4.1)	$97.80_{\pm 0.20}$ (−0.8)
		10%	$\mathbf{128.0}_{\pm 0.1}$ (↓ 4.7×)	$281.6_{\pm 1.7}$	$90.94_{\pm 0.39}$ (−2.5)	$85.69_{\pm 1.39}$ (−1.9)	$96.88_{\pm 0.52}$

predictive performance, we perform a set of adaptation and pruning runs using L_2-normalized Taylor pruning [30] on CIFAR-10 with a ResNet-18 network. Here, we vary the rank $r \in 2^{[0,6]}$ and initialization range in $10^{[-6,-2]}$ and evaluate along densities $\{1.0, 0.5, 0.3, 0.2, 0.1\}$. As illustrated in Fig. 5, we observe a clear and expected trend of increasing accuracy as the rank is increased. The increases exhibit diminishing returns with limited benefit beyond $r = 32$ for CIFAR-10. While all tested ranks show similar accuracy at a density of $d = 1.0$, the lowest-rank adapters are more severely affected by a lower d than higher-rank ones. This follows intuition, considering that lower-rank adapters have fewer parameters that might prove redundant during pruning. In some cases ($r \geq 16$) pruning at $d \approx 0.5$ increases accuracy. SPLoRA is generally robust to the chosen initialization range, showing only random variations within the tested set of values without clear trends in favor of particular ranges. We will use the initialization $\boldsymbol{W}_{\text{up}}^{(i,j)} \sim \mathcal{U}(-10^{-4}, 10^{-4})$ in subsequent experiments.

Comparison with fine-pruning In our comparison of SPLoRA and fine-pruning, we train a ResNet-50 network for weight-based transfer learning from ILSVRC 2022 to CIFAR-10, Oxford Flower 102, and Cats and Dogs across four structured pruning works, namely, the normalized Weight [23], Taylor [30], Gradient [42], and LRP [23] pruning methods. This comparison is conducted for both fine-pruning and SPLoRA with the ranks $r = 8$ and $r = 32$. To accommodate the stochastic nature of pruning and neural network training, we repeat each experiment three times and report the mean and standard deviation of each metric. The results of our experiments are presented in Table 1 for model densities of 100%, 30%, and 10% retained weights and visualized for CIFAR-10 in Fig. 6 as well as for Oxford Flowers 102 and Cats and Dogs in Appendix A.3.

While most combinations of pruning and learning meth-



(a) Weight pruning [23] (b) Gradient pruning [42] (c) Taylor pruning [30] (d) LRP pruning [45]

Figure 6. CIFAR-10 accuracy versus total learned parameter count $\|\Delta \boldsymbol{W}_t\|_0$ (top row) and FLOPs (bottom row) using fine-pruning and SPLoRA with ranks 32 and 8 for the (a) Weight [23], (b) Gradient [42], (c) Taylor [30], and (d) LRP [45] channel-pruning methods.

ods were successful, we found LRP with fine-pruning to be unstable at lower model densities. Despite attempts with multiple different seeds, only a single model was successfully pruned down to 10% density on Oxford Flowers 102 and none succeeded for Cats and Dogs. Unattainable results in Table 1 are noted with "-".

Comparing SPLoRA with fine-pruning, we observed competitive transfer adaptations despite learning a fraction the weights. While fine-pruning generally resulted in higher accuracy at 30% model density (on average 0.6% and 1.6% higher than SPLoRA ranks 32 and 16), SPLoRA had far fewer learned parameters (on average 6.2× and 17.0×). In the low-density regimen at 10%, SPLoRA was both more robust to pruning (achieving 6.9% and 4.7% higher average accuracy than fine-pruning for ranks 32 and 16), while reducing the number of learned parameters (by 2.0× and 4.2× on average). The difference in relative parameter reduction between the 30% and 10% density targets is explained partly by the normalization and linear parameters (73K for ResNet-50), that begin to dominate the learned weight budget at low densities, and partly by the remaining weights of adapter bottleneck channels. Even fewer parameters would be needed if a lower rank hyper-parameter was chosen or if, as discussed in Sec. 3.3, pruning of bottleneck channels was conducted as well. As FLOPs follow model densities, these are approximately equal for each learning method given equal densities.

4.2. Block-based Pruning

Experimental setup In our exploration of SPLoPA for block-based pruning, we use Magnitude Pruning [8] of a BERT-base model for hyper-parameter investigations considering 32 × 32 weight blocks per layer. Subsequently, we compare SPLoPA with fine-pruning on the Stanford Question Answering Dataset 1.1 (SQuAD) [34] using L_0 Regularization-based pruning (L_0 Reg.) [27], Movement Pruning (MvP) and Soft Movement Pruning (SMvP) [40]. We adopt a previously reported training-setup and hyper-parameters [40] and utilize Automated Gradual Pruning [47], where masked weights are continually updated and can be recovered according to the pruning criterion.

SPLoPA prototypes, rank, and weight sharing The SPLoPA method has multiple hyper-parameters, which impact the parameter count. The parameter count scales linearly with both the number of prototypes R and the prototype rank r. Moreover, memory usage can be decreased further by sharing prototypes among layers. In the subsequent experiments, we gauge the impact of these hyper-parameters on the predictive performance by conducting a sweep over a range of values, one parameter at a time, using Magnitude Pruning [8] at 60% density.

As shown in the top-left panel of Fig. 7, the F1 score increases with an increasing prototype count R until $R = 128$, beyond which the added expressiveness of additional prototypes does not translate to increased performance. We suspect this is dependent on the dataset size, with larger datasets benefiting from higher capacities. As SPLoPA has parameters in one-to-one proportion with R (6.96M for $R = 128$, $r = 1$), an R lower than 64 may be attractive if parameter count is of prime concern despite the associated performance reduction. The prototypes only produce a minor fraction of the SPLoPA parameters for BERT-base with 32×32 weight blocks. Accordingly, prototype sharing

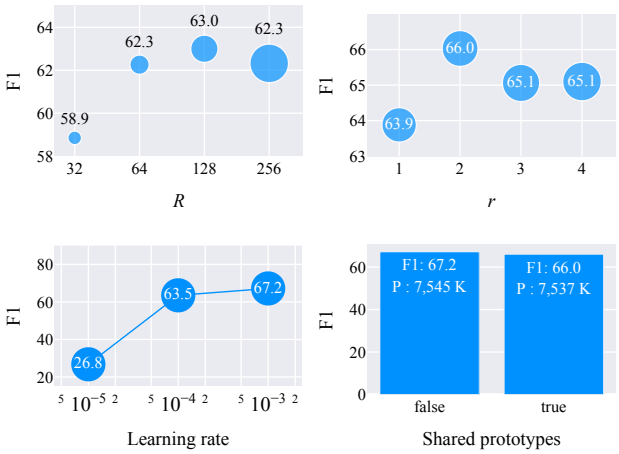


Figure 7. Hyper-parameter sweeps for BERT-base with MaP [8] and SPLoRA at 60% density. The bubble-sizes are proportional to ΔParams. Default parameters: $lr = 10^{-3}$, $R = 128$, $r = 1$ (top), and $r = 2$ (bottom).

yields very modest parameter reductions (by approximately 1‰), while reducing the F1 score[2]. Consequently, the effect of the prototype rank r in practice is primarily on the predictive power. As depicted in the top right corner of Fig. 7, $r = 2$ was found to produce the highest F1 score. We expect this to depend tightly on the block size, with larger blocks requiring higher ranks.

Comparison with fine-pruning In this set of experiments, we parameterize SPLoPA with $r = 2$ without prototype sharing. Table 2 presents the learned parameter count (ΔParams) and F1-score of SPLoPA and fine-pruning on the SQuAD dataset across various pruning methods and model densities near 30% and 10%[3]. For all pruning methods, SPLoPA learned adaptation weights and masks with significantly fewer parameters as compared to fine-pruning. Both fine-pruning and SPLoPA generally suffer F1 losses as model density is lowered. Comparing the two learning approaches, fine-pruning achieved slightly higher F1 (on average 0.8% and 2.5% higher than SPLoPA $R128$ and $R64$). On the other hand, SPLoPA required far fewer parameters (between 2.0× and 11.6× fewer).

5. Conclusion

We proposed Structured Pruning Adapters (SPAs) as an alternative to fine-tuning during structured pruning. Instead of updating all model weights, SPAs consist of prunable lightweight add-on modules, which are learned in place of the original weights but can be fused with them at run-time to obtain the same computational enhancements as regular structured pruning with fine-tuning. To accommodate a wide range of strutured pruning approaches, we proposed SPAs for both channel- and block-based pruning. These were shown to achieve competitive performance across a battery of pruning methods on computer vision and natural language processing benchmarks while requiring a fraction of the learned parameters per task. Thus, SPAs are ideal for task-switching storage-constrained and/or network-limited usage scenarios, where the per-model size should be small.

[2]We also explored sharing of block weights as reported in [28], but we found this to result in severely reduced reduced performance.

[3]Although we selected hyper-parameters for each pruning method with a target of 30% and 10% density, multiple methods did not allow for the exact specification of pruning ratios.

Table 2. Block-based transfer-pruning of BERT-base to the SQuAD dataset under block-based L_0 Regularization pruning [27], Movement Pruning [40], and Soft Movement Pruning (SMvP) [40]. Changes specified relative to closest fine-pruning density with same pruning method. Best metric per pruning-method is highlighted with bold.

Pruning method	**Learning method**	**Density**	**ΔParams** (M)	**SQuAD F1** (%)
L_0 R. [27]	Fine-pruning	29.6%	25.21	**81.49**
		10.1%	9.28	74.92
	SPLoPA-$^{R128}_{r2}$	29.5%	4.30 (↓ 5.9×)	80.75 (−0.7%)
		10.7%	2.30 (↓ 4.0×)	76.22 (+1.3%)
	SPLoPA-$^{R64}_{r2}$	30.3%	**2.19** (↓ 11.5×)	80.56 (−0.9%)
MvP [40]	Fine-pruning	30.1%	25.56	**82.97**
		10.1%	8.57	73.77
	SPLoPA-$^{R128}_{r2}$	31.7%	6.49 (↓ 3.9×)	79.98 (−3.0%)
		14.1%	4.36 (↓ 2.0×)	74.29 (+0.5%)
	SPLoPA-$^{R64}_{r2}$	10.1%	**2.20** (↓ 3.9×)	71.00 (−2.8%)
SMvP [40]	Fine-pruning	28.8%	24.48	**84.18**
		6.8%	5.84	75.61
	SPLoPA-$^{R128}_{r2}$	26.8%	4.02 (↓ 6.1×)	82.90 (−1.3%)
		6.0%	1.80 (↓ 3.2×)	73.79 (−1.8%)
	SPLoPA-$^{R64}_{r2}$	28.6%	2.10 (↓ 11.6×)	82.51 (−1.7%)
		6.9%	**0.95** (↓ 6.2×)	70.75 (−4.9%)

Acknowledgments

Lukas Hedegaard and Alexandros Iosifidis acknowledge funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR).

An acknowledgment also goes to Martin Damgaard Nielsen and Jens Dalgaard Nielsen for the early-stage conversations relating to this project.

References

[1] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier

decisions by layer-wise relevance propagation. *PLoS One*, 10(7):1–46, 2015. 2

[2] Jeremy Elson, John (JD) Douceur, Jon Howell, and Jared Saul. Asirra: A captcha that exploits interest-aligned manual image categorization. In *Proceedings of 14th ACM Conference on Computer and Communications Security (CCS)*. Association for Computing Machinery, Inc., 2007. 1, 5

[3] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[4] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations, (ICLR)*, 2019. 2

[5] Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, Jun 2021. 2

[6] R.M. Gray and D.L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6):2325–2383, 1998. 2

[7] Kailing Guo, Xiaona Xie, Xiangmin Xu, and Xiaofen Xing. Compressing by learning in a low-rank and sparse decomposition form. *IEEE Access*, 7:150823–150832, 2019. 2

[8] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. 2016. 7, 8

[9] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NeurIPS)*, page 1135–1143. MIT Press, 2015. 2

[10] Babak Hassibi and David G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Proceedings of the 5th International Conference on Neural Information Processing Systems*, NIPS'92, page 164–171, 1992. 2

[11] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*, 2022. 1, 2, 4

[12] Lukas Hedegaard, Arian Bakhtiarnia, and Alexandros Iosifidis. Continual Transformers: Redundancy-free attention for online inference. In *Advances in Neural Information Systems (NeurIPS) Workshop on Vision Transformers: Theory and Applications*, 2022. 2

[13] Lukas Hedegaard and Alexandros Iosifidis. Continual 3d convolutional neural networks for real-time processing of videos. In *European Conference on Computer Vision (ECCV)*, pages 1–18, 2022. 2

[14] Lukas Hedegaard, Illia Oleksiienko, and Christian Møldrup Legaard. DatasetOps, 2022. 5

[15] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015. 2

[16] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *preprint, arXiv:2203.15556*, 2022. 2

[17] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799, 2019. 1, 2

[18] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. 1, 2, 3

[19] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 1, 5

[20] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *preprint, arXiv:1404.5997*, 2014. 5

[21] François Lagunas, Ella Charlaix, Victor Sanh, and Alexander Rush. Block pruning for faster transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10619–10629, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics. 1, 2, 3

[22] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989. 2

[23] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations (ICLR)*, 2017. 1, 2, 6, 7, 12

[24] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, 2021. 1, 2

[25] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461(C):370–403, oct 2021. 2

[26] Congcong Liu and Huaming Wu. Channel pruning based on mean gradient for accelerating convolutional neural networks. *Signal Processing*, 156:84–91, 2019. 2

[27] Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations (ICLR)*, 2018. 2, 7, 8

[28] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. In *Advances in Neural Information Processing Systems*, 2021. 1, 2, 4, 8

[29] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 3

[30] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations (ICLR)*, 2017. 1, 2, 6, 7, 12

[31] Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. *Layer-Wise Relevance Propagation: An Overview*, pages 193–209. Springer International Publishing, 2019. 2

[32] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729, 2008. 1, 5

[33] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, 2021. 1, 2

[34] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, Nov. 2016. Association for Computational Linguistics. 1, 7

[35] Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 3

[36] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, volume 30, 2017. 1, 2

[37] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1, 2

[38] Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. AdapterDrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021. 2, 5

[39] 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 (IJCV)*, 115(3):211–252, Dec 2015. 1, 5

[40] Victor Sanh, Thomas Wolf, and Alexander M. Rush. Movement pruning: Adaptive sparsity by fine-tuning. In *NeurIPS*, 2020. 1, 2, 3, 7, 8

[41] Asa Cooper Stickland and Iain Murray. BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5986–5995, 2019. 2

[42] Xu Sun, Xuancheng Ren, Shuming Ma, and Houfeng Wang. meProp: Sparsified back propagation for accelerated deep learning with reduced overfitting. In *Proceedings of the 34th International Conference on Machine Learning (ICLR)*, volume 70 of *Proceedings of Machine Learning Research*, pages 3299–3308, International Convention Centre, Sydney, Australia, 2017. 1, 2, 6, 7, 12

[43] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 09–15 Jun 2019. 2

[44] Dat Thanh Tran, Alexandros Iosifidis, and Moncef Gabbouj. Improving efficiency in convolutional neural network with multilinear filters. *Neural Networks*, 105:328–339, 2018. 2

[45] Seul-Ki Yeom, Philipp Seegerer, Sebastian Lapuschkin, Alexander Binder, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Pruning by explaining: A novel criterion for deep neural network pruning. *Pattern Recognition*, 115, 2021. 1, 2, 3, 5, 6, 7, 12

[46] Aston Zhang, Yi Tay, SHUAI Zhang, Alvin Chan, Anh Tuan Luu, Siu Hui, and Jie Fu. Beyond fully-connected layers with quaternions: Parameterization of hypercomplex multiplications with $1/n$ parameters. In *International Conference on Learning Representations (ICLR)*, 2021. 4

[47] Michael Zhu and Suyog Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression. In *International Conference on Learning Representations (ICLR) Workshop Track Proceedings*, 2018. 2, 7

[48] Yaoming Zhu, Jiangtao Feng, Chengqi Zhao, Mingxuan Wang, and Lei Li. Counter-interference adapter for multilingual machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2812–2823, 2021. 1, 2, 4

A. Appendix

A.1. SPLoRA Derivation

Utilizing Eq. (3) in the context of channel-SPAs of the form expressed in Eq. (2), we can derive the Structured Pruning Low-rank Adapter defined in Eq. (4), where adapter parameters are pruned alongside source weights. This derivation is straightforward, considering that the application of a structured pruning mask $\boldsymbol{M} = \boldsymbol{m}_{\text{row}}\boldsymbol{m}_{\text{col}}^{\top}$ via Hadamard products is equivalent to a projection with diagonalized masking vectors:

$$\boldsymbol{W} \odot \boldsymbol{m}_{\text{row}}\boldsymbol{m}_{\text{col}}^{\top} = \text{diag}(\boldsymbol{m}_{\text{row}})\ \boldsymbol{W}\ \text{diag}(\boldsymbol{m}_{\text{col}}). \quad (7)$$

Similarly, a single diagonalized mask can be expressed via Hadamark products:

$$\text{diag}(\boldsymbol{m}_{\text{row}})\ \boldsymbol{W} = \boldsymbol{W} \odot \boldsymbol{m}_{\text{row}}\mathbf{1}^{\top}. \quad (8)$$

Utilizing Eq. (3), Eq. (7), and Eq. (8), we can rewrite Eq. (2) as follows:

$$\begin{aligned}
\boldsymbol{W}_t &= (\boldsymbol{W}_s + a(\Delta\boldsymbol{W}_t)) \odot \boldsymbol{m}_{\text{row}}\boldsymbol{m}_{\text{col}}^{\top} \\
&= (\boldsymbol{W}_s + \boldsymbol{W}_{\text{down}}\boldsymbol{W}_{\text{up}}) \odot \boldsymbol{m}_{\text{row}}\boldsymbol{m}_{\text{col}}^{\top} \\
&= \text{diag}(\boldsymbol{m}_{\text{row}})\ (\boldsymbol{W}_s + \boldsymbol{W}_{\text{down}}\boldsymbol{W}_{\text{up}})\ \text{diag}(\boldsymbol{m}_{\text{col}}) \\
&= \text{diag}(\boldsymbol{m}_{\text{row}})\ \boldsymbol{W}_s\text{diag}(\boldsymbol{m}_{\text{col}}) \\
&\quad + (\text{diag}(\boldsymbol{m}_{\text{row}})\boldsymbol{W}_{\text{down}})(\boldsymbol{W}_{\text{up}}\text{diag}(\boldsymbol{m}_{\text{col}})) \\
&= \boldsymbol{W}_s \odot \boldsymbol{m}_{\text{row}}\boldsymbol{m}_{\text{col}}^{\top} \\
&\quad + (\boldsymbol{W}_{\text{down}} \odot \boldsymbol{m}_{\text{row}}\mathbf{1}^{\top})(\boldsymbol{W}_{\text{up}} \odot \mathbf{1}\boldsymbol{m}_{\text{col}}^{\top}),
\end{aligned}$$

where the final result is equivalent to Eq. (4).

A.2. Training Durations

In this section, we provide a brief overview of approximate training durations for the methods tested in the present paper. As training times are comparable among different pruning methods, we report a single metric approximated from multiple pruning methods. These are presented in Table 3 for our experiments using ResNet-50 in image recognition tasks and Table 4 for BERT-base for question answering.

For computer vision experiments, the pruning methods gradually reduce the network density while producing pruned models at a predefined step reduction in density, cycling the learning rate for each density reduction step. Accordingly, the noted training times for the pruned learning methods in Table 3 includes the training of all models with densities ranging from 100% to 5% at 5% intervals.

For the natural language processing experiments, on the other hand, one triangular learning rate schedule is employed when moving from the unpruned state to the desired model density (i.e., the training durations for pruning down to 5% and 50% density are equal). Therefore, each result for SPLoPA is from an individual run.

Table 3. Training durations for the ResNet-50 model on image recognition transfer tasks using a NVIDIA RTX 2080 Ti GPU. For each dataset, the batch size (BS) and training duration (T) are presented.

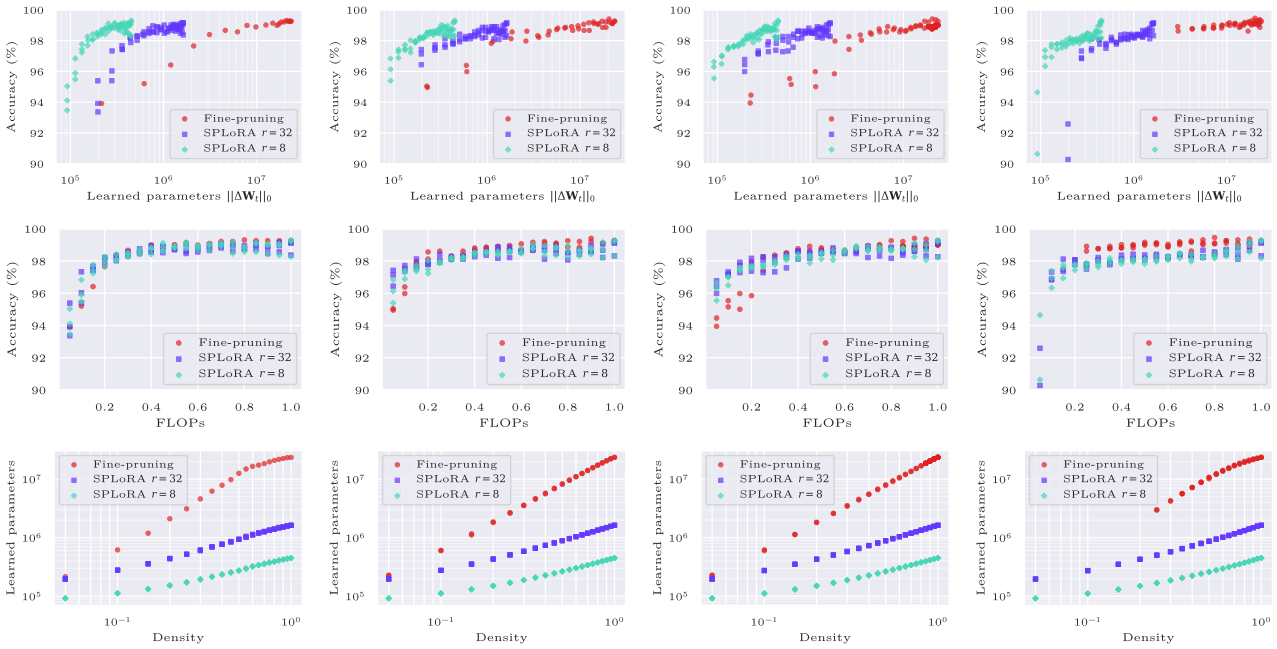
Pruning method	**Learning method**	**CIFAR-10**		**Oxf. Fl. 102**		**C. & D.**	
		BS	T	BS	T	BS	T
Unpruned	Fine-tuning	64	0:30	64	1:05h	64	2:20h
	SPLoRA-$r32$	64	0:30	32	1:10h	32	2:30h
	SPLoRA-$r8$	64	0:30	32	1:10h	32	2:30h
Pruned	Fine-pruning	64	10	4	25h	6	42h
	SPLoRA-$r32$	64	11	4	33h	6	48h
	SPLoRA-$r8$	64	11	4	31h	6	45h

Table 4. Training durations for BERT-base on natural language processing transfer tasks using a NVIDIA Tesla V100. For each dataset, the batch size (BS) and training time (T) are presented.

Learning method	**SQuAD**	
	BS	T
Fine-tuning	16	7h
SPLoPA	16	15h

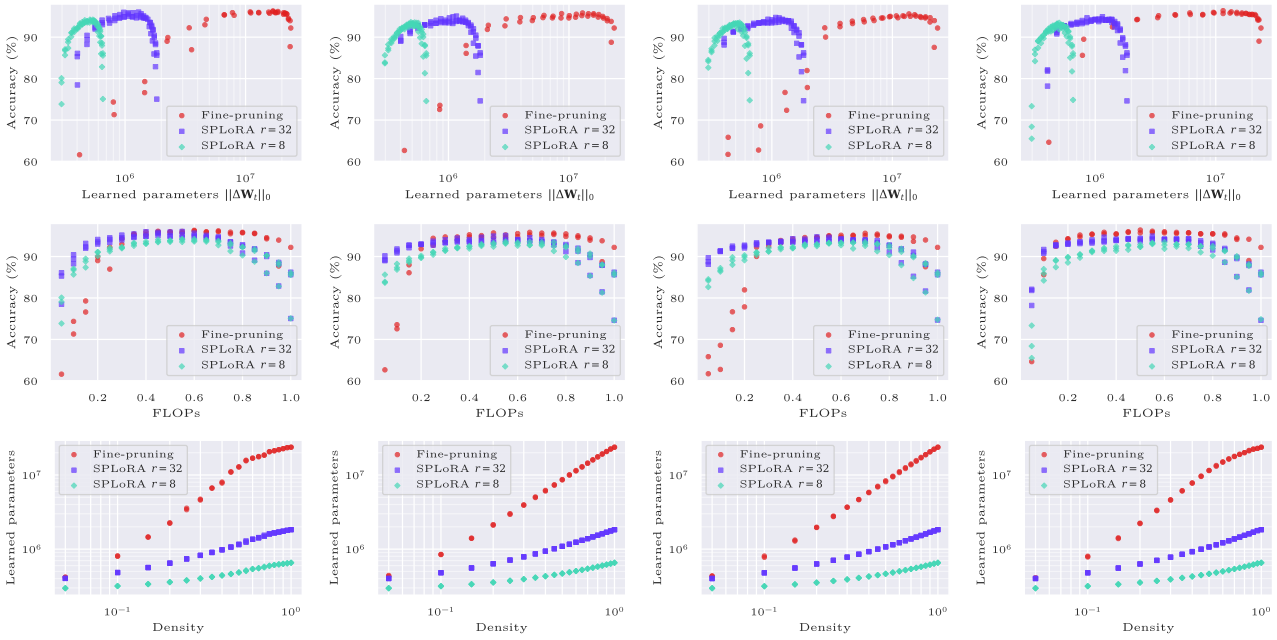
A.3. Supplemental Visualisations of experimental results with SPLoRA

For completeness, Fig. 8 and Fig. 9 illustrate trade-offs among accuracy, learned parameters, and FLOPs.



(a) Weight pruning [23]. (b) Gradient pruning [42]. (c) Taylor pruning [30]. (d) LRP pruning [45].

Figure 8. Cats and Dogs accuracy versus learned parameter count $\|\Delta \boldsymbol{W}_t\|_0$ (top row) and FLOPs (middle row) as well as learned parameter count versus model density (bottom row) using fine-pruning and SPLoRA with ranks 32 and 8 for various channel-pruning methods.



(a) Weight pruning [23]. (b) Gradient pruning [42]. (c) Taylor pruning [30]. (d) LRP pruning [45].

Figure 9. Oxford Flowers 102 accuracy versus learned parameter count $\|\Delta \boldsymbol{W}_t\|_0$ (top row) and FLOPs (middle row) as well as learned parameter count versus model density (bottom row) using fine-pruning and SPLoRA with ranks 32 and 8 for various channel-pruning methods.

8.6 Facial Expression Recognition with Learning Diversified Feature Representations

The appended paper follows.

LEARNING DIVERSIFIED FEATURE REPRESENTATIONS FOR FACIAL EXPRESSION RECOGNITION IN THE WILD

Negar Heidari and Alexandros Iosifidis

Department of Electrical and Computer Engineering, Aarhus University, Denmark

ABSTRACT

Diversity of the features extracted by deep neural networks is important for enhancing the model generalization ability and accordingly its performance in different learning tasks. Facial expression recognition in the wild has attracted interest in recent years due to the challenges existing in this area for extracting discriminative and informative features from occluded images in real-world scenarios. In this paper, we propose a mechanism to diversify the features extracted by CNN layers of state-of-the-art facial expression recognition architectures for enhancing the model capacity in learning discriminative features. To evaluate the effectiveness of the proposed approach, we incorporate this mechanism in two state-of-the-art models to (i) diversify local/global features in an attention-based model and (ii) diversify features extracted by different learners in an ensemble-based model. Experimental results on three well-known facial expression recognition in-the-wild datasets, AffectNet, FER+ and RAF-DB, show the effectiveness of our method, achieving state-of-the-art performance of 89.99% on RAF-DB, 89.34% on FER+ and the competitive accuracy of 60.02% on AffectNet dataset.

***Index Terms*—** Facial Expression Recognition, Feature Representation, Feature Diversity, Attention Network

1. INTRODUCTION

Facial expression as a fundamental natural signal for human social communication plays an important role in different applications of artificial intelligence, such as Human Computer Interaction (HCI), healthcare, and driver fatigue monitoring. Deep Convolutional Neural Networks (CNNs) have led to considerable progress in automatic Facial Expression Recognition (FER) on large-scale datasets in real-world scenarios. FER methods aim to solve a visual perception problem by learning feature representations from facial images/videos to be classified as an emotional category, i.e. happiness, sadness, fear, anger, surprise, disgust, neutral, and contempt. In laboratory-controlled datasets, such as CK+ [1] and JAFFE [2], where the facial images are in fixed frontal pose without any occlusion, FER methods have achieved excellent performance. However, these methods confront challenges for in-the-wild datasets, such as AffectNet [3], FER+ [4], and RAF-DB [5], where facial images come with illumination, occlusion and pose variations causing considerable change in facial appearance. To address that, many recent methods rely on transfer learning to exploit the feature representations learned for other visual perception tasks, such as object recognition, with well-designed networks, like ResNet-18 [6], trained on large datasets, like VGG-Face [7] and MS-Celeb-1M [8], to be transferred for facial expression recognition in challenging in-the-wild datasets. However, considering that many face datasets are small and imbalanced, these deep neural networks are mostly over-parameterized and tend to overfit on the training data, which can degrade their generalization ability on unseen data.

Increasing the diversity of features learned by different network layers/neurons has been recognized as an effective way to improve model generalization [9]. It is theoretically shown in [10, 11] that the within-layer activation diversity improves the generalization performance of neural networks and lowers the effect of overfitting. In this paper, we propose a mechanism for learning diversified facial feature representations by encouraging the learner to extract diverse spatial and channel-wise features. This mechanism can be used in different CNN architectures to increase the features diversity between layers or branches, spatial regions, and/or channels of feature maps. We incorporate our proposed optimization mechanism into two state-of-the-art models, i.e., the MA-Net [12] and the ESR [13], and conduct experiments on three well-known in-the-wild datasets, i.e., AffectNet, FER+ and RAF-DB. Experimental results demonstrate the effectiveness of learning diversified features in improving the accuracy and generalization of the pretrained state-of-the-art models on new samples.

The contributions of the paper can be summarized as follows:

- We propose a mechanism for learning diversified features in spatial and channel dimensions of CNNs to improve the model's accuracy in discriminating facial expressions.
- We evaluate our feature extraction mechanism by incorporating it into two state-of-the-art models which have different properties, i.e., one benefits from a region-based attention mechanism and transfer learning, and the other one is an efficient ensemble-based architecture. In both cases, our diversified feature learning mechanisms boost the performance.
- Conducted experiments on three benchmark in-the-wild datasets, including the large-scale dataset AffectNet, indicate the effectiveness and adaptability of our method, which can be used in different types of models. Our code is publicly available at https://github.com/negarhdr/Diversified-Facial-Expression-Recognition.

2. RELATED WORKS

Recent studies are focused on addressing the challenges of in-the-wild facial expression recognition by training models with multi-pose examples [14], and extracting key facial features based on facial landmarks and region-based attention mechanisms [15, 16, 17]. Learning facial features from global and local perspectives simulates the human brain's perception mechanism and helps achieving better performance in visual perception problems. MA-Net [12] is a global

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

multi-scale and local attention network which extracts features with different receptive fields, to increase the diversity and robustness of global features. This state-of-the-art method comprises of a backbone based on ResNet-18 for extracting preliminary features which are fed into a two-branch network with global multi-scale and local attention modules for high-level feature extraction. The first branch receives the preliminary feature maps as input and applies several multi-scale convolutions to extract both deeper semantic and shallower geometry features. The second branch of the network also receives the preliminary feature maps extracted by the backbone network as input, divides the feature maps into several local spatial regions without overlap, and then applies several parallel local attention networks to highlight the most important facial features in each region. At the end, a decision-level fusion strategy is employed to classify the extracted multi-scale and local attention features into different facial expression categories. However, this large network with 50.54 M parameters needs to be trained on a large dataset, and consistent with other state-of-the-art methods [18, 16], this network is first trained on MS-Celeb-1M dataset, and then finetuned on in-the-wild facial datasets AffectNet and RAF-DB.

ESR [13] has solved this issue by proposing an efficient ensemble-based method which reduces the residual generalization error on the AffectNet and FER+ datasets, and achieves state-of-the-art performance while training from scratch on these datasets. ESR model consists of two building blocks: 1) the base network which is composed of a stack of convolutional layers and is responsible for extracting low/middle-level features, 2) the ensemble network composed of several network branches which are supposed to learn distinctive features. All branches in this ensemble module receive the same feature maps extracted by the base network as input, and they compete for a common resource which is the base network. The training algorithm of ESR starts with training the base network and one ensemble branch. Thereafter, more convolutional branches are added one by one while training, so that the base network leads and speeds up learning by providing all ensemble branches with shared preliminary feature maps which are suitable for all the branches. Therefore, this method reduces redundancy in low-level feature learning and focuses on learning high-level discriminative features to be classified. Finally, the input facial image is classified to an emotion category by fusing the predictions of all the ensemble branches and applying majority voting.

In this paper we propose to complement discriminative feature extraction by increasing the features diversity between attention-regions, channel dimensions, and ensemble branches. In the next section, the diversified feature learning mechanism is introduced, and accordingly the modified learning mechanism for MA-Net and ESR methods is described.

3. PROPOSED METHOD

Diversity of feature representations is important in deep learning for enhancing the model generalization on unseen data, and improving model's accuracy in perceptual tasks by extracting non-redundant and discriminative features. Inspired by [10], we propose to increase spatial and channel-wise feature diversity in CNN architectures and ensemble-based models for facial expression recognition.

Let us assume that $\Phi_l, l \in \{1, 2, ..., L\}$ is a feature map of size $C \times H \times W$ extracted by a CNN learner. The diversity between different feature maps obtained by different learners or different layers of a CNN model can be obtained in channel and spatial dimensions as illustrated in Figure 1 by first applying pooling on spatial and channel dimensions and then computing the average similarity between every two pooled feature maps l, k using radial basis function as follows:

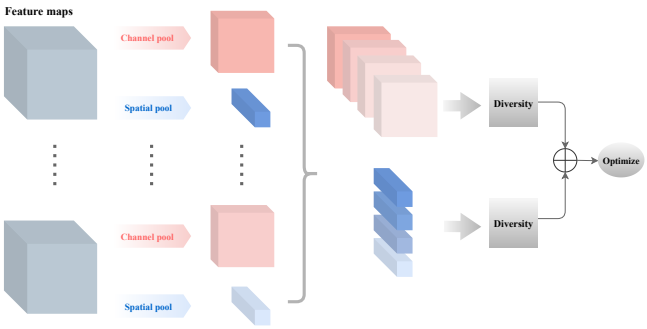


Fig. 1. Illustration of diversity mechanism over the channel and spatial dimensions of feature maps. In this mechanism, spatial and channel pooling are applied on the feature maps of different learners/layers, and the diversity between different pooled features are then computed and optimized by the training algorithm.

$$S_{lk} = \frac{1}{N} \sum_{i=1}^{N} exp(-\gamma \left\| \phi_l(\mathbf{x}_i) - \phi_k(\mathbf{x}_i) \right\|^2), \quad (1)$$

where N denotes the number of samples from which feature maps are extracted, γ is a hyperparameter, $\phi_l(\cdot)$ and $\phi_k(\cdot)$ denote the pooled feature maps of the l^{th} and k^{th} learners, respectively. The feature maps are of size $1 \times H \times W$ and $C \times 1 \times 1$ when similarity is measured on spatial (Figure 1 top row) and channel (Figure 1 bottom row) dimensions, respectively. Similar feature maps indicate low diversity of the learner. Accordingly, using the pairwise similarities between feature maps, the model diversity is obtained by computing the determinant of the matrix $\mathbf{S}$ indicating pairwise similarities of learners as S_{lk}, i.e.,:

$$\mathcal{D} = det(\mathbf{S}). \quad (2)$$

The model can be optimized in an end-to-end manner by minimizing the combined loss function comprising of classification loss and diversity. That is, the overall loss value to be minimized is:

$$Loss = \mathcal{L} - (\mathcal{D}_{ch} + \mathcal{D}_{sp}), \quad (3)$$

where $\mathcal{L}$ denotes the cross-entropy classification loss, and $\mathcal{D}_{ch}$, $\mathcal{D}_{sp}$ denote the feature diversity computed through channel and spatial dimensions, respectively using Eq. (1) and Eq. (2).

This mechanism can be used in CNN-based models to increase diversity between the feature maps at different levels. Considering the fact that diversity of learners is important in ensemble learning, encouraging each branch of ESR to learn complementary features of data can lead to better ensemble classification. To reach this goal, we modified the architecture of ESR by adding the CBAM attention mechanism [19] into each layer of the network and maximizing the diversity of both channel and spatial attention maps between different branches. The combined loss function of the modified ESR model is defined as a summation of the diversity loss between branches and the cross-entropy loss of each branch as follows:

$$\mathcal{L}_{esr} = \sum_{b} \mathcal{L}(f_b(\mathbf{X})) - (\mathcal{D}_{ch} + \mathcal{D}_{sp}), \quad (4)$$

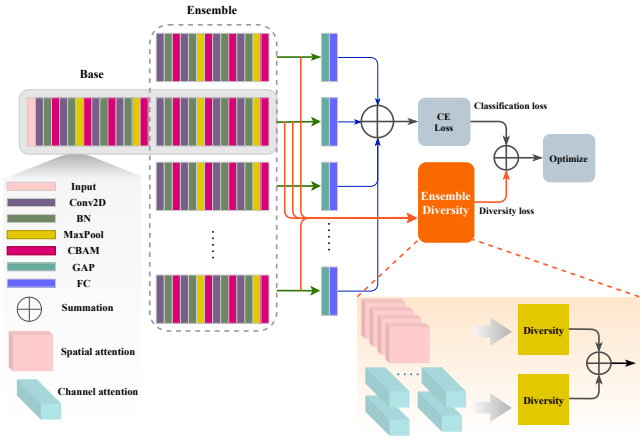


Fig. 2. Illustration of the modified ESR structure with added CBAM attention modules in each layer and the ensemble diversity block which diversifies the channel and spatial attention maps of the ensemble branches.

where $\mathcal{L}$ denotes the cross-entropy classification loss function for each of the ensemble learners f_b which is combined with a negative summation of spatial and channel diversity of the whole ensemble model. In other words, by optimizing $\mathcal{L}_{esr}$, the features of ensemble branches (learner) are diversified, while each branch is encouraged to classify features with minimum loss. Figure 2 illustrates the new structure of ESR with added attention modules in each layer and our augmented module which computes the ensemble diversity to be optimized with cross entropy loss.

In MA-Net, the focus is on exploiting both local and global feature in two model branches. CBAM attention mechanism is originally employed in this method to highlight the key global and local facial regions for recognizing the expression. As illustrated in Figure 3, the first branch of the network employs the feature map tensor in its initial shape to extract the global features, but the second branch divides the feature map into four patches to learn and highlight the local feature in each of the patches separately. We modified MA-Net structure to encourage the local branch to learn diversified regional features. In this regard, channel and spatial pooling operations are applied on divided patches and the diversity between them are computed to be added to the model classification loss function. Besides, in order to make the two branches of the network as effective as ensemble learner, the global and the local features extracted by these two branches can be diversified as well. In this regard, the local feature patches are concatenated and introduced to the global average pooling layer, along with the global feature map, and the pooled features are diversified by the branch diversity block and then classified by the fully connected layer. The whole model is optimized by minimizing a combined loss function comprising of local and global classification loss in addition to the branch and patch diversity as follows:

$$Loss = \lambda\mathcal{L}_{local} + (1-\lambda)\mathcal{L}_{global} - (\mathcal{D}_b + (\mathcal{D}_{sp} + \mathcal{D}_{ch})), \quad (5)$$

where $\mathcal{L}_{local}$, $\mathcal{L}_{global}$ denote the cross-entropy classification loss in the local and global branches, respectively, $0 \leq \lambda \leq 1$ is a hyper-parameter balancing the two parts which is set to 0.6 in MA-Net, $\mathcal{D}_b$ is the diversity between the two branches and $\mathcal{D}_{sp}$, $\mathcal{D}_{ch}$ indicate spatial and channel diversity between the local feature patches.

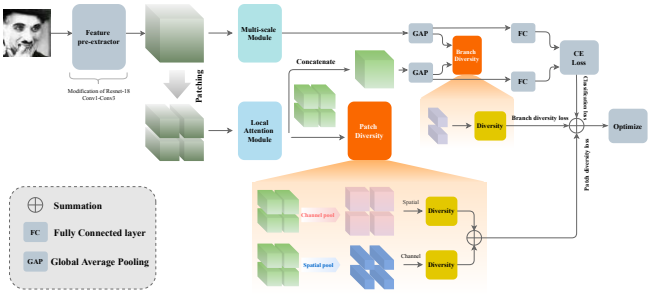


Fig. 3. Illustration of the modified MA-Net structure by adding patch diversity and branch diversity blocks to diversify local region-based features in each feature map and also increase diversity of the extracted global and local features before passing them to the classification layers.

4. EXPERIMENTS

We conducted experiments on three widely used in-the-wild datasets, AffectNet [3], FER+ [4], RAF-DB [5]. **AffectNet** is the largest in-the-wild dataset containing more than one million images collected from the Internet by querying emotion keywords in different languages. Following the same experimental setting as in RAN [16], ESR [13], and SCN [18], we used 450,000 images of this dataset which are manually annotated with 8 discrete expressions containing 6 basic ones (happiness, surprise, sadness, anger, disgust, fear) plus neutral and contempt. 287,568 images are used as training data and 4,000 images are used as test data. **FER+** is an extension of FER2013 [20] dataset, which is a large-scale dataset containing 35,887 facial images collected by Google search engine with 7 expressions. FER+ annotators re-labeled the FER2013 by crowdsourcing and added contempt expression to the dataset. All the face images in this dataset are aligned and annotated with 8 expressions. **RAF-DB** dataset comprises of 30,000 facial images annotated with basic or compound expressions. Similar to the experimental setting in state-of-the-art methods, we used images with basic expressions, including 12,271 training and 3,068 test image, in our experiments.

The experiments are conducted on PyTorch deep learning framework [21] with one GRX 1080-ti GPU, SGD optimizer with a momentum of 0.9 and cross-entropy loss function. We followed the experimental setting of ESR and MA-Net methods for reproducing their results and also training the modified version of the models on all the datasets. Table 1 shows a summary of our experimental results for all the three datasets, and Tables 2, 3, 4 compare the performance of the state-of-the-art methods with our proposed diversified ESR and MA-Net models in AffectNet, FER+ and RAF-DB, respectively.

To reproduce the ESR results with 9 ensemble branches, referred as ESR-9, the model is trained from scratch on AffectNet dataset and then finetuned on FER+. This led to 58.8% accuracy on AffectNet which is 0.5% less than the reported result in [13], however our reproduced result for FER+ is around 0.85% higher than their originally reported accuracy. Although the performance of this method is not reported in [13] for RAF-DB dataset, we finetuned the pre-trained ESR-9 model on RAF-DB as well. To evaluate the effect of attention layers added to the ESR structure, we did an ablation study to compare the performance of ESR-9 with and without CBAM attention layers, and spatial/channel-wise diversities. According to the results reported in Table 1, the best ensemble classification accuracy is obtained by adding CBAM attention layers, as well as maximiz-

Table 1. Comparisons of the classification accuracy of two state-of-the-art methods, ESR and MA-Net, and their modified versions with and without (channel and spatial) diversity computing on AffectNet, FER+, and RAF-DB datasets. $*$ indicates our proposed version of the method.

Method	Attention	Diversity		Dataset		
		Spatial	Channel	AffectNet	FER+	RAF-DB
ESR-9 [13]	×	×	×	**59.3**	87.17	-
ESR-9*	×	×	×	58.8	88.40	77.96
	✓	×	×	58.95	88.56	82.39
	✓	✓	×	59.25	88.59	82.92
	✓	✓	✓	**59.3**	**89.15**	**82.95**
ESR-15*	×	×	×	58.7	88.59	77.5
	✓	×	×	59.25	88.78	82.82
	✓	✓	×	59.47	89.21	82.92
	✓	✓	✓	**60.00**	**89.34**	**83.00**
MA-Net [12]	✓	×	×	**60.29**	-	88.4
MAN-Net*	✓	×	×	59.85	87.49	88.68
	✓	✓	✓	60.02	**88.34**	**89.99**

Table 2. Comparisons of the classification accuracy of the state-of-the-arts with our proposed version of ESR and MA-Net methods on AffectNet dataset with 8 classes. $*$ indicates our proposed version of the method.

Methods	Pretrained	Acc.(%)
MobileNet [22]	-	56.00
VGGNet [22]	-	58.00
AlexNet-WL [3]	-	58.00
RAN [16]	MS-Celeb-1M	59.50
SCN [18]	MS-Celeb-1M	60.23
ESR-9 [13]	AffectNet	59.30
ESR-9*	AffectNet	59.30
ESR-15*	AffectNet	**60.00**
MA-Net [12]	MS-Celeb-1M	**60.29**
MA-Net*	MS-Celeb-1M	60.02

Table 3. Comparisons of the classification accuracy of the state-of-the-arts with our proposed version of ESR and MA-Net methods on FER+ dataset with 8 classes. $*$ indicates our proposed version of the method.

Methods	Pretrained	Acc.(%)
TFE-JL [23]	-	84.30
PLD [4]	-	85.10
SHCNN [24]	-	86.54
SeNet50 [25]	VGG-Face2 [26]	88.80
RAN [16]	MS-Celeb-1M	88.55
	VGG-Face [7]	89.16
SCN [18]	MS-Celeb-1M	88.01
ESR-9 [13]	AffectNet	87.17
ESR-9*	AffectNet	**89.15**
ESR-15*	AffectNet	**89.34**
MAN-Net [12]	MS-Celeb-1M	-
MAN-Net*	MS-Celeb-1M	88.34

ing the diversity of both spatial and channel-wise attention between ensemble branches.

It is mentioned in [13] that adding more than 9 branches to ESR does not improve the performance. However, we assume that increasing the feature diversity between branches increases the model capacity for learning features with more ensemble branches. In this regard, we increased the number of branches both in the original ESR architecture and in our proposed version of ESR and trained the ESR-15 models on AffectNet and finetuned them on FER+ and RAF-DB. The results in Table 1 confirm our assumption, and indicate the improved performance of ESR-15 for all the three datasets compared to ESR-9. It should be noted that the maximum number of branches is chosen empirically and in some cases, the best performance is achieved in earlier branches so that at inference time, we can get the results with early exits. Considering the results in Table 3, ESR-15*, which is our modified version of ESR-15 with diversified features, outperforms all the state-of-the-arts with no need to be pretrained on large-scale datasets like MS-Celeb-1M or VGG-Face.

Table 4. Comparisons of the classification accuracy of the state-of-the-arts with our proposed version of ESR and MA-Net methods on RAF-DB dataset with 7 classes. $*$ indicates our proposed version of the method.

Methods	Pretrained	Acc.(%)
DLP-CNN [5]	-	84.22
IPA2LT [27]	AffectNet	86.77
gACNN [15]	AffectNet	85.07
LDL-ALSG [28]	AffectNet	85.53
RAN [16]	MS-Celeb-1M	86.90
SCN [18]	MS-Celeb-1M	87.03
ESR-9 [13]	AffectNet	-
ESR-9*	AffectNet	82.95
ESR-15*	AffectNet	83.00
MA-Net [12]	MS-Celeb-1M	88.40
MA-Net*	MS-Celeb-1M	**89.99**

Since MA-Net original structure includes CBAM attention layers, we did not modify the structure of the layers in this model. MA-Net is first trained on MS-Celeb-1M dataset, and the pretrained weights are then finetuned on AffectNet and RAF-DB datasets. In our experiments, we used the available pretrained weights of MA-Net on MS-Celeb-1M, provided by the authors, and finetuned the weights on all the three datasets. However the reproduced classification accuracy on AffectNet is 0.44% less than their reported accuracy of 60.29%, while for RAF-DB dataset we get 0.28% higher accuracy than the original one. After augmenting branch and patch diversity blocks into the MA-Net structure and diversifying local and global features, we achieved the state-of-the-art performance of 89.99% on RAF-DB dataset which outperforms all the other state-of-the-arts listed in Table 4.

5. CONCLUSION

In this paper we proposed a mechanism to diversify features extracted by different CNN learners for facial expression recognition. We targeted two state-of-the-art methods based on ensemble learning and multi-scale attention networks to evaluate the effect of learning diversified features in performance. Experimental results show that diversifying features extracted by different ensemble learners can enhance the overall ensemble classification performance while increasing the model capacity to include more learners for feature extraction. Furthermore, diversifying local regional features extracted by a CNN learner improves the model performance in exploiting local features and classifying facial images.

6. REFERENCES

[1] 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, pp. 94–101.

[2] Michael Lyons, Shigeru Akamatsu, Miyuki Kamachi, and Jiro Gyoba, "Coding facial expressions with gabor wavelets," in *IEEE International Conference on Automatic Face and Gesture Recognition*, 1998.

[3] Ali Mollahosseini, Behzad Hasani, and Mohammad H Mahoor, "Affectnet: A database for facial expression, valence, and arousal computing in the wild," *IEEE Transactions on Affective Computing*, vol. 10, no. 1, pp. 18–31, 2017.

[4] Emad Barsoum, Cha Zhang, Cristian Canton Ferrer, and Zhengyou Zhang, "Training deep networks for facial expression recognition with crowd-sourced label distribution," in *International Conference on Multimodal Interaction*, 2016.

[5] Shan Li, Weihong Deng, and JunPing Du, "Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[7] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman, "Deep face recognition," 2015.

[8] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao, "Ms-celeb-1m: A dataset and benchmark for large-scale face recognition," in *European Conference on Computer Vision*, 2016, pp. 87–102.

[9] Bo Xie, Yingyu Liang, and Le Song, "Diverse neural network learns true target functions," in *Artificial Intelligence and Statistics*, 2017, pp. 1216–1224.

[10] Firas Laakom, Jenni Raitoharju, Alexandros Iosifidis, and Moncef Gabbouj, "Within-layer diversity reduces generalization gap," *International Conference on Machine Learning Workshop on Information Theoretic Methods for Rigorous, Responsible, and Reliable Machine Learning*, 2021.

[11] Firas Laakom, Jenni Raitoharju, Alexandros Iosifidis, and Moncef Gabbouj, "On feature diversity in energy-based models," *International Conference on Learning Representations Workshop on Energy-Based Models*, 2021.

[12] Zengqun Zhao, Qingshan Liu, and Shanmin Wang, "Learning deep global multi-scale and local attention features for facial expression recognition in the wild," *IEEE Transactions on Image Processing*, vol. 30, pp. 6544–6556, 2021.

[13] Henrique Siqueira, Sven Magg, and Stefan Wermter, "Efficient facial feature learning with wide ensemble-based convolutional neural networks," in *AAAI Conference on Artificial Intelligence*, 2020, vol. 34, pp. 5800–5809.

[14] Feifei Zhang, Tianzhu Zhang, Qirong Mao, and Changsheng Xu, "Joint pose and expression modeling for facial expression recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3359–3368.

[15] Yong Li, Jiabei Zeng, Shiguang Shan, and Xilin Chen, "Occlusion aware facial expression recognition using cnn with attention mechanism," *IEEE Transactions on Image Processing*, vol. 28, no. 5, pp. 2439–2450, 2018.

[16] Kai Wang, Xiaojiang Peng, Jianfei Yang, Debin Meng, and Yu Qiao, "Region attention networks for pose and occlusion robust facial expression recognition," *IEEE Transactions on Image Processing*, vol. 29, pp. 4057–4069, 2020.

[17] Negar Heidari and Alexandros Iosifidis, "Progressive spatio-temporal bilinear network with monte carlo dropout for landmark-based facial expression recognition with uncertainty estimation," in *IEEE International Workshop on Multimedia Signal Processing*, 2021.

[18] Kai Wang, Xiaojiang Peng, Jianfei Yang, Shijian Lu, and Yu Qiao, "Suppressing uncertainties for large-scale facial expression recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6897–6906.

[19] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon, "Cbam: Convolutional block attention module," in *European Conference on Computer Vision*, 2018, pp. 3–19.

[20] Ian J Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, et al., "Challenges in representation learning: A report on three machine learning contests," in *International Conference on Neural Information Processing*, 2013.

[21] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in pytorch," in *Neural Information Processing Systems Workshops*, 2017.

[22] Charlie Hewitt and Hatice Gunes, "Cnn-based facial affect analysis on mobile devices," *arXiv preprint arXiv:1807.08775*, 2018.

[23] Ming Li, Hao Xu, Xingchang Huang, Zhanmei Song, Xiaolin Liu, and Xin Li, "Facial expression recognition with identity and emotion joint learning," *IEEE Transactions on Affective Computing*, vol. 12, no. 2, pp. 544–550, 2018.

[24] Si Miao, Haoyu Xu, Zhenqi Han, and Yongxin Zhu, "Recognizing facial expressions using a shallow convolutional neural network," *IEEE Access*, vol. 7, pp. 78000–78011, 2019.

[25] Samuel Albanie, Arsha Nagrani, Andrea Vedaldi, and Andrew Zisserman, "Emotion recognition in speech using cross-modal transfer in the wild," in *ACM International Conference on Multimedia*, 2018.

[26] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman, "Vggface2: A dataset for recognising faces across pose and age," in *IEEE International Conference on Automatic Face & Gesture Recognition*, 2018, pp. 67–74.

[27] Jiabei Zeng, Shiguang Shan, and Xilin Chen, "Facial expression recognition with inconsistently annotated datasets," in *European Conference on Computer Vision*, 2018.

[28] Shikai Chen, Jianfeng Wang, Yuedong Chen, Zhongchao Shi, Xin Geng, and Yong Rui, "Label distribution learning on auxiliary label space graphs for facial expression recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

8.7 Self-Attention Neural Bag-of-Features

The appended paper follows.

2022 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, AGU. 22–25, 2022, XI'AN, CHINA

SELF-ATTENTION NEURAL BAG-OF-FEATURES

Kateryna Chumachenko[1], *Alexandros Iosifidis*[2] *and Moncef Gabbouj*[1]

[1]Department of Computing Sciences, Tampere University, Tampere, Finland
[2]Department of Electrical and Computer Engineering, Aarhus University, Aarhus, Denmark
Emails: {kateryna.chumachenko,moncef.gabbouj}@tuni.fi, ai@ece.au.dk

ABSTRACT

In this work, we propose several attention formulations for multivariate sequence data. We build on top of the recently introduced 2D-Attention and reformulate the attention learning methodology by quantifying the relevance of feature/temporal dimensions through latent spaces based on self-attention rather than learning them directly. In addition, we propose a joint feature-temporal attention mechanism that learns a joint 2D attention mask highlighting relevant information without treating feature and temporal representations independently. The proposed approaches can be used in various architectures and we specifically evaluate their application together with Neural Bag of Features feature extraction module. Experiments on several sequence data analysis tasks show the improved performance yielded by our approach compared to standard methods.

1. INTRODUCTION

Sequence data modeling became an important task in the field of machine learning, finding applications in a wide range of areas. These include speech recognition [1], video processing [2], biosignal anlaysis [3], and natural language processing [4]. Multiple methods directed at solving sequence data analysis tasks were proposed to date. Notable approaches include those based on Recurrent Neural Networks, such as Gated Recurrent Units [4] or Long Short Term Memory models [5] that aim to explicitly model the sequential nature of the data with variable length and capture its temporal information. In addition, methods based on Transformers have been proposed as well, modelling the data representations as token sequences with self-attention between tokens being the main driving force of the model [6]. Besides, methods that were originally developed for other types of data, such as Convolutional Neural Networks or Neural Bag of Features [7, 8], were shown beneficial in sequential data analysis tasks.

Concurrent with the development of these methods, approaches directed towards improving robustness of baseline models have been emerging, with the attention modules [6] being one of the most notable ones. The goal of attention module is generally defined as highlighting relevant information in the model while suppressing less relevant one. This idea has been applied to a wide range of base models, and explicit definitions of different attention variants vary between specific models and data types. In CNNs, attention is generally calculated in a form of a learned mask of weights that is applied element-wise to the intermediate feature representation to facilitate learning of stronger features, where mask can be applied both in channel or spatial dimensions [9, 10, 11]. Another relevant incarnation of an attention model is that of multi-head self-attention that serves as a building block in Transformer models. In this formulation, relevance of features is quantified by their relations in the learnt latent space.

Bag of Features (BoF) [12] model has been widely used for feature extraction from image data, later emerging to other data types as well, including sequential data [13, 14]. The learning process of BoF consists of two stages, with the first stage being dictionary learning, during which a codebook of representative features (codewords) is learnt. During the second stage of BoF, the learnt codebook is used to quantize the low-level feature representation of data into a histogram. To facilitate more powerful feature extraction, Discriminant Bag of Features approaches were proposed [15, 16], while Neural Bag of Features (NBoF) was proposed as a neural network generalization of BoF [7]. NBoF can be used as an independent feature extractor or as a submodule of a bigger architecture, and can be optimized end-to-end in either case. Besides, an attention module for Neural Bag of Features has been recently proposed to address some of its limitations and increase the robustness of the model [17, 18]. Specifically, 2D Attention (2DA) proposed three attention types: input attention, with the aim of addressing the noise present in input data; codeword attention, with the aim of highlighting most relevant codewords in a codebook; and temporal attention, with the aim of highlighting most relevant temporal dimensions in the representation.

In this paper, we propose to reformulate the idea of 2D-Attention in sequence data and evaluate it in Neural Bag of Features model. Our contributions are summarized as follows:

- We revisit the definition of 2D-Attention, and propose self-attention based alternatives capable of more powerful quantification of feature relevance. We propose self-attention based formulations of both temporal and codeword attention.
- We develop codeword-temporal self-attention to facilitate learning of representation relevance in joint codeword-temporal latent space, rather than treating codeword and temporal attentions separately.
- We evaluate the developed methods on sequence data analysis tasks, including acoustic scene classification and cardiac disease recognition from ECG and PCG signals, and aciheve competitive performance.

The remainder of the paper is organized as follows. Section 2 provides an overview of the related work, Section 3 describes the proposed formulations of 2DA self-attentions, Section 4 provides experimental results evaluating their performance in a variety of time-series analysis problems against related approaches, and Section 5 concludes the paper.

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

978-1-6654-8547-0/22//$31.00 ©2022 IEEE

2. RELATED WORK

Neural Bag of Features (NBoF) [19] is a neural extension of the Bag of Features algorithm that can be utilized both as an independent learning method, as well as incorporated into larger models to facilitate more powerful feature extraction. NBoF consists of two steps, namely, dictionary learning and feature quantization. Specifically, NBoF model 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 full model architecture in an end-to-end manner. Further, aggregation step is performed, where the extracted histogram representations, known as codewords, are aggregated by averaging. To date, several feature quantization approaches have been proposed, including those based on Radial Basis Function (RBF) [19] and hyperbolic kernel [8]. Here we revisit the original definition based on RBF kernel.

Formally, NBoF with an RBF kernel is defined as follows. Given a sequence of N feature representations $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$, the quantization layer produces a sequence of quantized features $\mathbf{\Phi} = [\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_N] \in \mathbb{R}^{K \times N}$, where $\boldsymbol{\phi}_n = [\phi_{n,1}, \dots, \phi_{n,K}]^T \in \mathbb{R}^K$ is the quantized representation corresponding to feature $\mathbf{x}_n$. The output of k^{th} RBF neuron for feature $\mathbf{x}_n$ is given as follows:

$$\phi_{n,k} = \frac{\exp\big(-\|(\mathbf{x}_n - \mathbf{v_k}) \odot \mathbf{w}_k\|_2\big)}{\sum_{m=1}^{K} \exp\big(-\|(\mathbf{x}_n - \mathbf{v}_m) \odot \mathbf{w}_m\|_2\big)}, \tag{1}$$

where $\mathbf{v}_k$ is the k^{th} codeword, K is the total number of codewords, and $\mathbf{w}_k \in \mathbb{R}^D$ is a learnable parameter controlling the shape of the Gaussian kernel.

Following the quantization step, the quantized features are aggregated by averaging:

$$\mathbf{y} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{\phi}_n. \tag{2}$$

Although providing reasonable feature extraction capabilities in a variety of problems, NBoF has a number of limitations. One of such limitations is that each learned codeword is considered to be equally important in the learned representation, and hence contributes equally to the prediction, although it is reasonable to assume that certain codewords have learnt more powerful features. With respect to sequence data modeling, another limitation is that during the aggregation step, quantized features are combined by simple averaging, disregarding the relative importance of each timestamp. Nevertheless, temporal information can be of great importance in a variety of sequence learning tasks, such as speech command recognition, or dynamic activity recognition, where order of learnt feature representation can be a defining factor for the prediction.

To address these limitations, an attention mechanism for sequence data has been proposed with NBoF as a baseline in mind [17]. Specifically, the method is referred to as 2D-Attention (2DA) and defines three attention types: input attention, codeword attention, and temporal attention, that aim to emphasize the most relevant input data features, quantized features, and temporal timestamps, respectively.

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

$$A = softmax(\mathbf{\Phi}\mathbf{W}), \tag{3}$$

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

$$\tilde{\mathbf{\Phi}} = \mathcal{F}_{2DA}(\mathbf{\Phi}) = \alpha(\mathbf{\Phi} \odot \mathbf{A}) + (1 - \alpha)\mathbf{\Phi}, \tag{4}$$

where α is a learnt parameter controlling the strength of attention matrix and $\tilde{\mathbf{\Phi}}$ is the attended representation.

The first attention type introduced in 2DA is the codeword attention, the aim of which is to highlight most relevant codewords obtained at quantization step of the NBoF model while suppressing the non-relevant ones. This is desirable under the assumption that the output of each quantization neuron contributes differently to the final prediction. Formally, given the output of the quantization step $\mathbf{\Phi} \in \mathbb{R}^{K \times N}$, an attention mask $\mathbf{A} \in \mathbb{R}^{K \times N}$ of attention weights is applied to the features $\mathbf{\Phi}$ in order to highlight or suppress its rows, i.e., codewords, by applying the 2DA to $\mathbf{\Phi}^T$:

$$\tilde{\mathbf{\Phi}}_{CA} = \mathcal{F}_{2DA}(\mathbf{\Phi}^T). \tag{5}$$

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 it is desired to highlight individual series in the input data, the process is similar to that of codeword attention, and 2DA is applied to $\mathbf{X}^T$:

$$\tilde{\mathbf{X}}_{IA} = \mathcal{F}_{2DA}(\mathbf{X}^T). \tag{6}$$

This type of attention is referred to as input attention.

In turn, temporal attention aims to highlight relevant timestamps in the sequence during the aggregation step of the NBoF model to address the limitation of the representations being simply averaged during the aggregation step. Formally, it is achieved by applying 2DA on columns of $\mathbf{\Phi}$:

$$\tilde{\mathbf{\Phi}}_{TA} = \mathcal{F}_{2DA}(\mathbf{\Phi}). \tag{7}$$

3. PROPOSED METHODS

Although the 2DA attention addresses certain limitations of the NBoF model in terms of highlighting most relevant attributes in the quantized feature representation, 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 learnt and applied simultaneously, such approach does not take into account potential relationships of learned codewords with the temporal representations in the training phase as the masks are learned independently. At the same time, they are not necessarily independent in real-world problems, 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.

3.1. Codeword-temporal self-attention

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 $\mathbf{\Phi} \in \mathbb{R}^{K \times N}$, where K denotes the number of codewords and N denotes the temporal length, we obtain the attention matrix by quantifying the relations between 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 $\mathbf{\Phi}$ temporally and codeword-wise into a joint d-dimensional space.

$$\begin{aligned} \mathbf{q}_n = &\quad \mathbf{\Phi}\mathbf{W}_q^{nT}, \quad \mathbf{q}_n \in K \times d, \\ \mathbf{k}_n = &\quad \mathbf{\Phi}^T\mathbf{W}_k^{nT}, \quad \mathbf{k}_n \in N \times d. \end{aligned} \tag{8}$$

Further, to quantify the relations of learnt features in the joint space 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, to scale the values. Since at this time we do not aim to promote competition within codewords or timestamps, but rather learn a joint two-dimensional attention matrix, we choose the sigmoid activation function to scale the values to desirable range. An alternative can be using softmax over flattened 2D representation, but we empirically observed no benefit in following this approach. Further, the learnt attention matrix is applied element-wise to the input feature representation. Following the widely-used definition of multi-head self-attention [6], n attention matrices can be calculated independently, with the outputs of all heads subsequently concatenated. The attention matrix $\mathbf{A}_n$ corresponding to the head n, feature representation $\tilde{\mathbf{\Phi}}_n$, and the combined feature representation $\tilde{\mathbf{\Phi}}$ are therefore given as:

$$\mathbf{A}_n = \sigma(\frac{\mathbf{q}_n\mathbf{k}_n^T}{\sqrt{d}}) \in K \times N, \tag{9}$$

$$\tilde{\mathbf{\Phi}}_n = \alpha_n\mathbf{\Phi} + (1-\alpha_n)\mathbf{A}_n \odot \mathbf{\Phi}, \tag{10}$$

$$\tilde{\mathbf{\Phi}} = [\tilde{\mathbf{\Phi}}_1, ..., \tilde{\mathbf{\Phi}}_n]. \tag{11}$$

3.2. Codeword self-attention

A similar idea can be further developed into enhancing the independent codebook and temporal attentions in 2DA. In the standard definition, the projection matrix $\mathbf{W}$ outlined in Eq. 3 is fully learnt from scratch, with a role of highlighting relevant codewords or temporal features in $\mathbf{\Phi}$. Although by design the aim of $\mathbf{W}$ is to converge to the values that reflect the relevance of the corresponding codewords/timestamps, being optimized from scratch, nothing ensures or guides $\mathbf{W}$ towards reflecting these relations. To account for this, we propose to explicitly derive the attention matrix by means of calculating dot-product similarity of codewords in the latent space. That is, considering the codeword attention, we define two learnable projection matrices $\mathbf{W}_q^n \in d \times N$ and $\mathbf{W}_k^n \in d \times N$ from which latent representations of $\mathbf{\Phi}$ are learnt as:

$$\begin{aligned} \mathbf{q}_n = &\quad \mathbf{\Phi}\mathbf{W}_q^{nT} \in K \times d, \\ \mathbf{k}_n = &\quad \mathbf{\Phi}\mathbf{W}_k^{nT} \in K \times d. \end{aligned} \tag{12}$$

Following this, we can calculate the codeword attention as a $K \times K$ matrix following Eq. 9, where we utilize softmax as σ to promote competition between codewords.

Note that unlike 2DA, following this approach the learnable parameters are responsible for merely learning a latent space, where relevance of the codewords is explicitly calculated by means of dot product similarity, rather than directly learning the relevance of each codeword as in 2DA. The learnt attention matrix is subsequently multiplied with a feature representation $\mathbf{\Phi}$ to highlight the most relevant codewords and multi-head approach can be followed here as well:

$$\tilde{\mathbf{\Phi}}_n = \alpha_n\mathbf{\Phi} + (1-\alpha_n)\mathbf{A}_n\mathbf{\Phi} \tag{13}$$

$$\tilde{\mathbf{\Phi}} = [\tilde{\mathbf{\Phi}}_1, ..., \tilde{\mathbf{\Phi}}_n]. \tag{14}$$

3.3. Temporal self-attention

Following the same principle, temporal self-attention can be defined by quantifying temporal relevance of the representation by calculating this in a latent space. To achieve this, temporal self-attention can be calculated by simply operating on the transpose of the feature representation $\mathbf{\Phi}$, leading to $N \times N$ attention matrix encoding relative importance of each temporal dimension. Specifically, the queries, keys, and combined multi-head representation can be achieved as follows:

$$\begin{aligned} \mathbf{q}_n = &\quad \mathbf{\Phi}^T\mathbf{W}_q^{nT} \in N \times d \\ \mathbf{k}_n = &\quad \mathbf{\Phi}^T\mathbf{W}_k^{nT} \in N \times d \end{aligned} \tag{15}$$

$$\tilde{\mathbf{\Phi}}_n = \alpha_n\mathbf{\Phi}^T + (1-\alpha_n)A_n\mathbf{\Phi}^T \tag{16}$$

$$\tilde{\mathbf{\Phi}} = [(\tilde{\mathbf{\Phi}}_1)^T, ..., (\tilde{\mathbf{\Phi}}_n)^T] \tag{17}$$

4. EXPERIMENTAL EVALUATION

In this section we report the experimental evaluation of the proposed self-attention mechanisms and compare them with standard 2-DA attention. All the experiments are conducted with the logistic formulation of Neural Bag of Features [7] that uses hyperbolic kernel as a quantization layer and we use 256 codewords. We perform experiments on two tasks, namely, biosignal analysis and audio analysis. We denote by 2DA-CA and 2DA-TA the conventional 2DA attention in its codebook and temporal formulations, respectively, and by 2DA-CTSA$_d$, 2DA-TSA$_d$, and 2DA-CSA$_d$ - the proposed variants of codebook-temporal self-attention, temporal self-attention, and codebook self-attention with the dimensionality of the latent space denoted by d. Note that d is a hyperparameter which can be tuned, but we instead report the results across multiple values. Unless otherwise specified, single-head models are used.

4.1. Audio analysis

The first type of sequence data that we consider is audio. Specifically, we evaluate the NBoF models with the proposed attention approaches on the task of acoustic scene classification defined by TUT-UAS2018 dataset [20]. The dataset poses a task of classification of surrounding environments by their sounds, where 10 classes of urban environments are defined: airport, shopping mall, metro station, street pedestrian, public square, street traffic, tram, bus, metro, park. We extract mel-spectrogram feature representations with 128 frequency bands that are used as an input to a set of convolutional layers as defined in [17] to facilitate feature extraction, followed by NBoF module. The models are trained for 90 epochs with Adam optimizer and we use the batch size of 256. We utilize accuracy as the performance metric and report the accuracy of validation set on 90th epoch averaged across three runs.

The results of the proposed attention models and competing standard 2DA models are reported in Table I. Here and throughout the paper, we highlight the best result in bold and underline the results that outperform the baseline 2DA attention models. Specifically, TSA, i.e., temporal self-attention is compared with TA, i.e., standard temporal attention, CSA is compared with CA, and CTSA is considered to outperform standard 2DA if it outperforms both CA and TA, i.e., both standard codeword and temporal attention models.

As can be seen in Table I, the best result is achieved by the proposed temporal self-attention model that outperforms both 2DA baselines. All of the proposed temporal self-attention models outperform the temporal 2DA, and similar result is achieved by codeword self-attention that mostly outperforms codeword 2DA. All of

Table 1. Accuracies on TUT-UAS2018 dataset

Attention models	TUT-UAS
2DA-CA	56.15 + 0.21
2DA-TA	56.09 + 0.51
2DA-CTSA$_{512}$	56.20 + 1.11
2DA-CTSA$_{256}$	57.53 + 1.28
2DA-CTSA$_{128}$	56.84 + 1.07
2DA-CTSA$_{64}$	56.56 + 0.59
2DA-TSA$_{512}$	56.81 + 0.63
2DA-TSA$_{256}$	**57.55 + 1.40**
2DA-TSA$_{128}$	57.11 + 0.86
2DA-TSA$_{64}$	56.91 + 0.91
2DA-CSA$_{512}$	57.18 + 0.53
2DA-CSA$_{256}$	55.62 + 1.11
2DA-CSA$_{128}$	56.94 + 0.62
2DA-CSA$_{64}$	56.74 + 1.44

Table 2. F1 scores on biosignal datasets

Attention models	PCG	PCG-2	AF
2DA-CA	86.93 + 0.35	73.44 + 1.23	77.33 + 2.44
2DA-TA	87.45 + 0.74	73.39 + 1.16	76.71 + 2.06
2DA-CTSA$_{512}$	87.75 + 0.78	73.75 + 1.81	77.56 + 1.75
2DA-CTSA$_{256}$	87.46 + 1.30	73.50 + 0.77	77.55 + 2.42
2DA-CTSA$_{128}$	87.74 + 0.65	73.62 + 1.80	76.96 + 1.24
2DA-CTSA$_{64}$	87.07 + 1.02	73.38 + 1.36	77.87 + 1.71
2DA-TSA$_{512}$	88.06 + 0.61	73.46 + 1.45	76.86 + 2.34
2DA-TSA$_{256}$	87.26 + 0.52	74.14 + 1.77	76.87 + 1.86
2DA-TSA$_{128}$	87.08 + 1.00	**74.47 + 1.03**	77.27 + 2.13
2DA-TSA$_{64}$	87.77 + 0.61	73.31 + 1.58	76.99 + 1.74
2DA-CSA$_{512}$	88.36 + 0.22	73.35 + 1.15	77.28 + 1.60
2DA-CSA$_{256}$	**88.38 + 0.55**	73.95 + 0.90	77.70 + 1.90
2DA-CSA$_{128}$	87.19 + 0.98	73.02 + 2.14	**78.70 + 1.50**
2DA-CSA$_{64}$	87.71 + 0.44	72.79 + 0.67	77.96 + 1.88

the variants of the proposed codeword-temporal attentions outperform the baseline 2DA.

4.2. Biosignal analysis

The second type of sequence data considered by our approach is biosignal data. Timely diagnosis 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. In our experiments addressed towards this task, we consider two of the widely-adopted biosignals, namely, Electrocardiogram (ECG) and Phonocardiogram (PCG). The first dataset that we consider is the Atrial Fibrillation dataset (AF) that poses the task of atrial fibrillation recognition from ECG signals which are provided as the development data (training set) in the Physionet/Computing in Cardiology Challenge 2017 [21]. Specifically, the task is formulated as a classification problem with 4 classes: normal sinus rhythm, atrial fibrillation, alternative rhythm, and noise. Each ECG signal lasts between 9 to 60 seconds, which we clip or zero-pad to achieve the length of 30 seconds. Further, prior to applying the NBoF module, we add several preprocessing convolutional layers to the model to facilitate feature extraction. Specifically, we utilize the same architecture as proposed in [17]. We perform 5-fold cross-validation and report the average F1 score across the folds. The rest of training hyperparameters are as described in audio classification task.

The second dataset considered for the task of biosignal analysis is the PCG dataset of Phonocardiograms that come from the training set provided in the Physionet/Computing in Cardiology Challenge 2016 [22]. Two different tasks are posed in this dataset: abnormal phonocardiogram detection, and phonocardiogram quality evaluation, where both tasks are binary classification problems. Due to varying lengths of signals in the datset, we extract 5 second segments for classification similarly to [17]. For feature preprocessing, we extract mel-spectrogram with 24 bands and a window of 25 ms, which are subsequently fed to several preprocessing convolutional layers similarly to [17] and then to NBoF model. Other training hyperparameters are similar to those of AF dataset, except 3-fold cross-validation is used due to the smaller daatset size.

The results of biosignal analysis tasks are shown in Table II. As can be seen, in all three cases the best result is achieved by one of the proposed variants. In PCG dataset, codeword-temporal variant in high dimensions outperforms both codeword and temporal 2DA, and codeword self-attention significantly outperforms the codeword 2-DA. At the same time, in temporal representations the proposed approach outperform the 2DA approach in quality evaluation task on PCG dataset. Similar results are observed in AF dataset, where proposed self-attention approaches outperform codeword and temporal 2DA.

We further perform evaluation of the proposed methods with respect to different parameters. Specifically, we evaluate utilization of different number of heads in the models, as well as different Neural Bag of Features formulations.

4.3. Self-attention with multiple heads

Using multiple heads in self-attention modules has been shown beneficial in a variety of tasks, as learning multiple latent spaces in parallel allows the model to jointly attend to information from different representation subspaces at different positions [6]. On the other hand, using multiple heads yields additional model parameters. Here, we evaluate the proposed self-attention modules with variants consisting of 2 and 4 heads. In these variants, we use dropout of 0.2 on the attention matrix of codeword and temporal formulations as defined in [6].

Table III shows the results on TUT-UAS dataset using 2 and 4 heads in multi-head self-attention. As can be seen, the proposed approaches mostly outperform the standard 2DA. Compared to single-head variant, the multihead model with 4 heads perform the best, leading to performance gain of tup to 2.5%. In terms of biosignal datasets shown in Table IV, it can be seen that the overall results are rather similar between the head numbers in terms of which variants perform well in which datasets. In addition, utilization of multiple heads bring an improvement similarly to the acoustic scene dataset.

4.4. Using other NBoF formulations

All experiments were performed with the logistic NBoF formulation [7]. However, our proposed self-attention module can be equally utilized with other formulations as well. Here, we evaluate our approaches and competing 2DA approaches using the temporal variant of NBoF that defines two codebooks, long-term and short-term [8]. We refer to this method as TNBoF. Here we utilize the acoustic scene classification dataset and evaluate the TNBoF baseline with our approaches with both single and multi-head variants, as shown in Tables VI and VII. We can see that using this model, codeword self-

Table 3. Accuracies on TUT-UAS2018 dataset with 2 and 4 heads

Attention models	TUT-UAS, h=2	TUT-UAS, h=4
2DA-CA	56.15 + 0.21	56.15 + 0.21
2DA-TA	56.09 + 0.51	56.09 + 0.51
2DA-CTSA$_{512}$	57.23 + 1.00	57.04 + 0.80
2DA-CTSA$_{256}$	56.15 + 1.17	**58.52 + 0.70**
2DA-CTSA$_{128}$	**57.48 + 0.64**	58.02 + 0.45
2DA-CTSA$_{64}$	54.91 + 1.22	58.07 + 1.92
2DA-CTSA$_{32}$	57.21 + 0.29	56.31 + 0.74
2DA-TSA$_{512}$	56.61 + 0.91	55.82 + 1.18
2DA-TSA$_{256}$	55.80 + 0.98	56.07 + 0.48
2DA-TSA$_{128}$	55.84 + 0.51	57.62 + 1.71
2DA-TSA$_{64}$	57.83 + 0.16	56.71 + 0.81
2DA-TSA$_{32}$	56.31 + 1.10	57.83 + 0.23
2DA-CSA$_{512}$	57.40 + 0.23	57.13 + 1.20
2DA-CSA$_{256}$	56.37 + 1.24	55.45 + 0.71
2DA-CSA$_{128}$	55.62 + 1.18	56.91 + 1.31
2DA-CSA$_{64}$	56.99 + 1.21	56.54 + 1.45
2DA-CSA$_{32}$	56.04 + 1.06	56.26 + 1.53

Table 4. F1 scores on biosignal datasets with 2 heads

Models	PCG-1	PCG-2	AF
2DA-CA	86.93 + 0.35	73.44 + 1.23	77.33 + 2.44
2DA-TA	87.45 + 0.74	73.39 + 1.16	76.71 + 2.06
2DA-CTSA$_{512}$	87.80 + 0.73	74.57 + 1.14	76.43 + 2.78
2DA-CTSA$_{256}$	87.84 + 0.12	73.14 + 0.70	76.60 + 1.70
2DA-CTSA$_{128}$	87.24 + 0.74	73.77 + 1.02	77.06 + 1.39
2DA-CTSA$_{64}$	88.04 + 0.52	73.73 + 1.16	76.98 + 1.92
2DA-CTSA$_{32}$	87.63 + 0.83	73.69 + 0.98	77.79 + 2.00
2DA-TSA$_{512}$	86.98 + 0.76	73.66 + 0.78	76.77 + 2.26
2DA-TSA$_{256}$	87.61 + 0.70	73.32 + 1.15	76.31 + 1.59
2DA-TSA$_{128}$	87.69 + 1.11	72.64 + 2.19	77.23 + 1.48
2DA-TSA$_{64}$	87.09 + 0.60	73.55 + 0.80	77.21 + 1.95
2DA-TSA$_{32}$	87.03 + 0.44	74.38 + 1.81	77.41 + 2.18
2DA-CSA$_{512}$	87.47 + 0.78	72.97 + 0.72	**77.88 + 1.43**
2DA-CSA$_{256}$	**88.31 + 0.60**	**74.94 + 1.77**	77.70 + 1.69
2DA-CSA$_{128}$	87.33 + 0.68	74.46 + 0.62	77.47 + 0.96
2DA-CSA$_{64}$	87.49 + 0.84	73.41 + 1.13	76.91 + 1.11
2DA-CSA$_{32}$	87.72 + 0.57	73.08 + 0.60	76.69 + 1.22

Table 5. F1 scores on biosignal datasets with 4 heads

Attention models	PCG-1	PCG-2	AF
2DA-CA	86.93 + 0.35	73.44 + 1.23	77.33 + 2.44
2DA-TA	87.45 + 0.74	73.39 + 1.16	76.71 + 2.06
2DA-CTSA$_{512}$	87.88 + 0.56	74.34 + 0.85	77.09 + 1.24
2DA-CTSA$_{256}$	88.04 + 0.53	73.37 + 1.94	**78.26 + 1.69**
2DA-CTSA$_{128}$	86.91 + 0.29	72.60 + 1.18	77.51 + 1.75
2DA-CTSA$_{64}$	86.65 + 0.57	73.62 + 1.58	77.87 + 2.29
2DA-CTSA$_{32}$	87.74 + 0.67	73.12 + 0.13	77.61 + 1.66
2DA-TSA$_{512}$	86.94 + 0.60	72.92 + 1.76	76.69 + 1.54
2DA-TSA$_{256}$	87.45 + 0.71	73.59 + 0.89	77.13 + 1.84
2DA-TSA$_{128}$	87.20 + 0.23	73.37 + 0.80	77.03 + 2.18
2DA-TSA$_{64}$	87.29 + 0.39	74.07 + 1.11	76.94 + 2.47
2DA-TSA$_{32}$	87.03 + 0.90	73.69 + 0.73	77.56 + 1.80
2DA-CSA$_{512}$	88.27 + 0.63	73.29 + 1.51	77.77 + 1.72
2DA-CSA$_{256}$	**88.59 + 0.81**	74.16 + 1.58	77.59 + 1.64
2DA-CSA$_{128}$	87.67 + 0.41	72.82 + 1.82	77.30 + 1.60
2DA-CSA$_{64}$	87.37 + 0.56	73.60 + 1.40	77.26 + 1.67
2DA-CSA$_{32}$	87.12 + 0.66	**74.35 + 0.97**	76.73 + 1.77

Table 6. Accuracy scores on TUT-UAS2018 datast with TNBoF model with 1 head

Attention models	TUT-UAS, TNBoF
2DA-CA	56.79 + 0.60
2DA-TA	55.89 + 0.34
2DA-CTSA$_{64}$	57.04 + 0.84
2DA-CTSA$_{128}$	56.51 + 0.46
2DA-CTSA$_{256}$	57.35 + 1.05
2DA-CTSA$_{512}$	**58.19 + 0.62**
2DA-TSA$_{64}$	56.94 + 0.63
2DA-TSA$_{128}$	56.46 + 0.63
2DA-TSA$_{256}$	56.32 + 0.26
2DA-TSA$_{512}$	56.09 + 0.62
2DA-CSA$_{64}$	55.70 + 0.20
2DA-CSA$_{128}$	56.41 + 0.44
2DA-CSA$_{256}$	56.83 + 0.57
2DA-CSA$_{512}$	56.46 + 0.29

Table 7. Accuracy scores on TUT-UAS2018 dataset with TNBoF model with 2 and 4 heads, respectively

Attention models	TUT-UAS, TNBoF	
	h=2	h=4
2DA-CA	56.79 + 0.60	56.79 + 0.60
2DA-TA	55.89 + 0.34	55.89 + 0.34
2DA-CTSA$_{512}$	56.26 + 0.92	56.36 + 0.82
2DA-CTSA$_{256}$	57.08 + 0.86	56.57 + 0.65
2DA-CTSA$_{128}$	**57.77 + 0.91**	57.31 + 1.01
2DA-CTSA$_{64}$	56.51 + 1.89	56.42 + 0.22
2DA-CTSA$_{32}$	57.25 + 0.91	57.38 + 0.79
2DA-TSA$_{512}$	56.51 + 1.03	56.79 + 0.59
2DA-TSA$_{256}$	57.45 + 0.97	56.98 + 1.79
2DA-TSA$_{128}$	56.61 + 1.16	56.78 + 0.58
2DA-TSA$_{64}$	55.95 + 0.99	56.24 + 1.24
2DA-TSA$_{32}$	56.17 + 0.31	**58.40 + 0.70**
2DA-CSA$_{512}$	55.89 + 0.52	56.15 + 1.47
2DA-CSA$_{256}$	57.13 + 1.18	56.29 + 1.97
2DA-CSA$_{128}$	57.04 + 1.11	55.45 + 0.41
2DA-CSA$_{64}$	55.77 + 1.58	55.95 + 0.91
2DA-CSA$_{32}$	55.60 + 0.68	57.09 + 0.71

attention mostly outperforms the basedline codeword 2DA, and the other variants mostly outperform the baseline. At the same time, it can be seen that overall the best performing variant is the single-head one, hence utilization of additional heads degrades the performance rather than improves it in this case.

5. CONCLUSION

In this paper, we revisited the standard formulation of a 2DA attention mechanism and proposed several ways of enhancing it. The proposed ways are based on self-attention and allow to quantify codeword and/or temporal relevances through latent spaces rather than learning them directly. We evaluated the proposed approaches together with the Neural Bag-of-Features model on a few sequence learning tasks. The experimental evaluation has shown the benefits of the proposed approaches. Since the proposed attention models are generic methods aimed towards multivariate sequence data, further work into its applications with other architectures and tasks remains as a future research direction.

6. REFERENCES

[1] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 6645–6649.

[2] Bin Zhao, Xuelong Li, and Xiaoqiang Lu, "Hsa-rnn: Hierarchical structure-adaptive rnn for video summarization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7405–7414.

[3] Shraddha Singh, Saroj Kumar Pandey, Urja Pawar, and Rekh Ram Janghel, "Classification of ecg arrhythmia using recurrent neural networks," *Procedia computer science*, vol. 132, pp. 1290–1297, 2018.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[5] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[7] Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis, "Temporal logistic neural bag-of-features for financial time series forecasting leveraging limit order book data," *Pattern Recognition Letters*, vol. 136, pp. 183–189, 2020.

[8] Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis, "Temporal bag-of-features learning for predicting mid price movements using high frequency limit order book data," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 6, pp. 774–785, 2018.

[9] Jie Hu, Li Shen, and Gang Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[10] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[11] Firas Laakom, Kateryna Chumachenko, Jenni Raitoharju, Alexandros Iosifidis, and Moncef Gabbouj, "Learning to ignore: rethinking attention in cnns," in *British Machine Vision Conference (BMVC)*, 2021.

[12] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. IEEE, 2006, vol. 2, pp. 2169–2178.

[13] Yu-Gang Jiang, Chong-Wah Ngo, and Jun Yang, "Towards optimal bag-of-features for object categorization and semantic video retrieval," in *Proceedings of the 6th ACM international conference on Image and video retrieval*, 2007, pp. 494–501.

[14] Matthew Riley, Eric Heinen, and Joydeep Ghosh, "A text retrieval approach to content-based audio retrieval," in *Int. Symp. on Music Information Retrieval (ISMIR)*, 2008, pp. 295–300.

[15] Alexandros Iosifidis, Anastasios Tefas, and Ioannis Pitas, "Multidimensional sequence classification based on fuzzy distances and discriminant analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 93, no. 6, pp. 1445–1457, 2013.

[16] Alexandros Iosifidis, Anastasios Tefas, and Ioannis Pitas, "Discriminant bag of words based representation for human action recognition," *Pattern Recognition Letters*, vol. 49, pp. 185–192, 2014.

[17] Dat Thanh Tran, Nikolaos Passalis, Anastasios Tefas, Moncef Gabbouj, and Alexandros Iosifidis, "Attention-based neural bag-of-features learning for sequence data," *arXiv preprint arXiv:2005.12250*, 2020.

[18] Firas Laakom, Nikolaos Passalis, Jenni Raitoharju, Jarno Nikkanen, Anastasios Tefas, Alexandros Iosifidis, and Moncef Gabbouj, "Bag of color features for color constancy," *IEEE Transactions on Image Processing*, vol. 29, pp. 7722–7734, 2020.

[19] Nikolaos Passalis and Anastasios Tefas, "Neural bag-of-features learning," *Pattern Recognition*, vol. 64, pp. 277–294, 2017.

[20] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen, "Tut acoustic scenes 2017, evaluation dataset," Nov. 2017.

[21] Gari D Clifford, Chengyu Liu, Benjamin Moody, H Lehman Li-wei, Ikaro Silva, Qiao Li, AE Johnson, and Roger G Mark, "Af classification from a short single lead ecg recording: The physionet/computing in cardiology challenge 2017," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[22] Gari D Clifford, Chengyu Liu, Benjamin Moody, David Springer, Ikaro Silva, Qiao Li, and Roger G Mark, "Classification of normal/abnormal heart sound recordings: The physionet/computing in cardiology challenge 2016," in *2016 Computing in cardiology conference (CinC)*. IEEE, 2016, pp. 609–612.

8.8 Self-Attention Fusion for Audiovisual Emotion Recognition with Incomplete Data

The appended paper follows.

Self-attention fusion for audiovisual emotion recognition with incomplete data

Kateryna Chumachenko[1], Alexandros Iosifidis[2] and Moncef Gabbouj[1]
[1]Department of Computing Sciences, Tampere University, Tampere, Finland
[2]Department of Electrical and Computer Engineering, Aarhus University, Aarhus, Denmark
Emails: {kateryna.chumachenko,moncef.gabbouj}@tuni.fi, ai@ece.au.dk

***Abstract*—In this paper, we consider the problem of multi-modal data analysis with a use case of audiovisual emotion recognition. We propose an architecture capable of learning from raw data and describe three variants of it with distinct modality fusion mechanisms. While most of the previous works consider the ideal scenario of presence of both modalities at all times during inference, we evaluate the robustness of the model in the unconstrained settings where one modality is absent or noisy, and propose a method to mitigate these limitations in a form of modality dropout. Most importantly, we find that following this approach not only improves performance drastically under the absence/noisy representations of one modality, but also improves the performance in a standard ideal setting, outperforming the competing methods.**

I. INTRODUCTION

Recognition of human emotional states is an important task within the field of machine learning, enabling better understanding of social signals in the wide range of applications ranging from robotics to human-computer interaction [1], [2]. Multiple approaches and emotion models have been proposed to date, ranging from the task of recognizing discrete emotional states, such as 'happy', 'angry', or 'sad', to the estimation of emotional attributes, such as arousal and valence on a continuous scale [3], [4]. The task has been approached from multiple angles with different data types used as input, including text [5], speech [6], and images [7].

With the abundance of available data, a wide range of methods aiming to fully take advantage of this data are emerging, giving momentum to the development of multi-modal methods [8], [9], [10]. Multi-modal methods are a class of methods that operate jointly on multiple data types. These include, among others, video data that consists of audio and visual modalities [11], joint RGB and depth images [12], and RGB and skeleton data [13]. Methods operating on such multi-modal representations range from simple decision-level fusion approaches to more advanced joint feature learning approaches. Although fusion of intermediate features can potentially yield better performance due to joint learning of representations of multiple modalities, late or early fusion remain a popular choice in modern architectures due to their simplicity and versatility [8]. On the other hand, early fusion is not suitable for fusion of drastically different data types, while late fusion primarily only considers features learnt in each modality independently.

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

Most multi-modal methods developed to date assume full presence of all the adopted modalities at all times during inference, and only evaluate the performance of the models in such a setting. Nevertheless, in real-world applications it is often probable for one of the modalities to be missing or having poor quality at certain times, hence robustness of the model to such scenarios is an essential factor in building multi-modal systems operating on real-world data.

In the task of multi-modal emotion recognition, especially in the recent transformer-based architectures, the trend has been largely in utilization of pre-extracted features that are subsequently fused with a learnt model, rather than creating end-to-end trainable models [14], [15], [16]. This limits the applicability of such methods in real-world scenarios, as necessary feature extraction is often challenging in unconstrained settings and introduces another point of uncertainty to the overall processing pipeline. This is especially the case for the methods adopting language information [14], [15], as text transcriptions of audio signals are rarely available in practical applications and require separate estimation. We therefore primarily target the task of audiovisual emotion recognition that does not require separate feature learning.

In our work, we aim to address these limitations of existing multi-modal emotion recognition methods by building an end-to-end model that does not require prior feature learning, and performing fusion at intermediate level, while being robust to incomplete or noisy data samples. Our contributions can be summarized as follows:

- We propose a new architecture for audiovisual emotion recognition from facial videos and speech which does not rely on separately learnt features and learns end-to-end from raw videos;
- We employ several modality fusion approaches and propose an attention-based intermediate feature fusion approach that softly attends to modality-independent features. To the best of our knowledge, such approach has not been proposed before;
- We propose a new training scheme based on modality dropout mechanisms aimed to improve the robustness of the model under incomplete or noisy data of one modality. We additionally find that the proposed approach

yields better performance also in the standard case under presence of both modalities.

II. Related Work

Emotion recognition has received a significant amount of attention by the machine learning community, and a number of methods aiming to solve this task have been proposed to date. These methods operate on various data types, such as images [7], speech [6], text [5], or biosignals [17]. At the same time, methods combining these modalities have been proposed as well [14], [16], [15] employing different multi-modal fusion techniques.

Within the field of multimodal machine learning, generally, three classes of multi-modal fusion approaches are identified: early fusion, where the input data of multiple modalities are simply combined via concatenation, addition, or any other operation and further processed together; late fusion, where modalities are treated independently and their features or softmax classification scores are only combined in the very last layers; and intermediate feature fusion, where feature sharing is performed at middle layers of the network and hence the feature representations of different modalities are learnt jointly.

A notable set of approaches in multimodal fusion rely on utilization of self-attention [18]. Recall that self-attention is formulated via calculating the dot-product similarity in the latent space, where queries $\mathbf{q}$, keys $\mathbf{k}$, and values $\mathbf{v}$ are learnt from input feature representation via learnable projection matrices $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v$, and an attented representation is calculated based on them:

$$\mathbf{A}_n = softmax\left(\frac{\mathbf{q}\mathbf{k}^T}{\sqrt{d}}\right)\mathbf{v}, \tag{1}$$

where d is the dimensionality of a latent space. Considering the task of fusion of two modalities a and b, self-attention can be utilized as a fusion approach by calculating queries from modality a and keys and values from modality b. This results in representation learnt from modality a attending to corresponding modality b and further applying the obtained attention matrix to the representation learnt from modality b.

This idea has been extensively utilized for solving a plethora of tasks involving multimodal fusion. In the context of emotion/affect recognition, notable works include [14], [16], [15]. In [14] the authors propose a multimodal transformer-based architecture for unaligned multimodal language sequences and consider fusion of three modalities, namely, audio, vision, and text. Data from each modality is first projected via a 1D convolutional layer to the desired dimensionality, and further, a set of transformer blocks is applied. Specifically, two transformer modules are utilized in each modality branch, with each of the two modules being responsible for fusion of one of the other modalities with the modality of the given branch following the above-specified approach. These representations are subsequently concatenated and another transformer module is applied in each modality branch on joint representations. The processed representations from each of three branches are subsequently concatenated for final classification.

Another relevant work is [16] where audio and visual modalities are considered for the task of emotion recognition. There, each modality is first preprocessed with a separate transformer block and representations learnt from each modality are fused with a transformer in a manner similar to Eq. (1). Compared to previous work, only one modality is fused into the other one, rather than performing fusion in a pair-wise manner in separate branches.

Another work described in [15] considers emotion recognition from speech and text modalities. Similarly, they first perform modality-specific feature learning by means of convolutional blocks, and further employ two cross-modal attention blocks with one fusion audio into text, and the other one performing fusion into opposite direction. The statistics is pooled from each branch and concatenated for prediction.

As can be noticed, all the aforementioned methods are rather similar in their fusion strategies in that the transformer fusion is the building block of each of them, and the differences between the architectures are rather nominal and dataset-specific. At the same time, it can be noted that the models focus on building multimodal fusion methods rather than end-to-end emotion recognition systems, and often employ features that require separate estimation, especially for the vision modality. For example, [14] and [16] rely on facial action units as features, and [14] and [15] utilize language modality which requires separate annotation or estimation in practical application.

It should also be noticed that all three mentioned methods perform fusion in early or intermediate stages in the pipeline, forcing joint representations to be learnt. While benefiting from the joint feature learning, such fusion can become a curse if the learnt fused representations are too co-dependent and one of the modalities is noisy, incomplete, or simply non-existing during inference. Indeed, common practice has been to only evaluate the performance of the models under the ideal scenario of both modalities being present and complete at all times, while real-world applications do not necessarily reflect such scenarios.

III. Proposed methods

Here we describe the overall architecture of the proposed audiovisual emotion recognition model as well as three self-attention based modality fusion methods. Further, we propose an approach for accounting for missing data in one modality during inference in a form of modality dropout. On a general level, the model consists of two branches responsible for learning audio and visual features, respectively, and fusion modules placed either in the end or in the middle of the two branches depending on the feature fusion type, as shown in Figure 1. In both audio and visual branches, the 1D Convolutional blocks that are applied in a temporal dimension are primarily used.

A. Feature extraction

1) Vision branch: The vision branch consists of two parts, with the first part being the visual feature extraction from

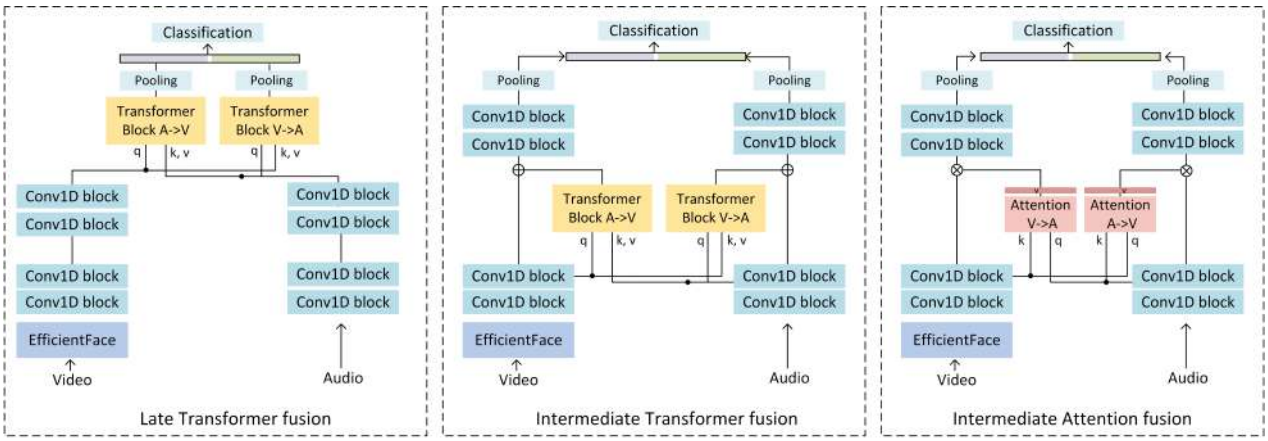


Fig. 1: Multimodal data fusion approaches.

individual video frames, followed by learning of joint representation for the whole video sequence. To achieve an end-to-end trainable model capable of learning from raw video, we employ feature extraction as part of our pipeline and optimize it jointly with the multimodal fusion module, unlike the vast majority of existing works that separate feature extraction from multimodal fusion and mostly utilize pre-extracted features, such as facial landmark locations, facial action units, or head pose information [14], [16]. We choose one of the recently proposed facial expression recognition architectures, namely, EfficientFace [7] and incorporate it for feature extraction from individual frames prior to introducing them to subsequent 1D convolutional blocks. Specifically, the 1D convolutional blocks are added after average-pooled output of the last convolutional block of EfficientFace.

Considering an input video sequence of k frames, each of the k frames is processed independently by a 2D feature extractor, resulting in a single vector descriptor for each frame. These representations are further concatenated and processed in a temporal dimension with the temporal convolution blocks described further. We choose to follow this approach, as opposed to directly employing 3D-convolutions as commonly done in video tasks, as it provides a number of advantages in the given task, with the first one being the lower computational overhead brought by 2D convolutional layers compared to 3D convolutions. It can also be argued that temporal relations are of less importance in emotion recognition task, hence 1D convolutional operations applied in temporal dimension are sufficient to capture this information. Another major benefit of following the proposed approach is the ability to employ 2D feature extractor pre-trained on larger image-based emotion recognition datasets, as such datasets are significantly less common for videos that are necessary for pre-training 3D-convolutional models.

Although we are primarily interested in building an end-to-end pipeline that can learn from raw data, the first part, i.e., visual feature extraction can be decoupled from the model and any other features can be used instead as input to the

Architecture of the visual branch

	EfficientFace module
Stage1	Reshape Conv1D [k=3, d=64, s=1] + BN1D + ReLU Conv1D [k=3, d=64, s=1] + BN1D + ReLU
Stage2	Conv1D [k=3, d=128, s=1] + BN1D + ReLU Conv1D [k=3, d=128, s=1] + BN1D + ReLU
Predict	Global Average Pooling + Linear

Architecture of the audio branch

Stage1	Conv1D [k=3, d=64] + BN1D + ReLU + MaxPool1d [2x1] Conv1D [k=3, d=128] + BN1D + ReLU + MaxPool1d [2x1]
Stage2	Conv1D [k=3, d=256] + BN1D + ReLU + MaxPool1d [k=2] Conv1D [k=3, d=128] + BN1D + ReLU + MPool1D [k=2]
Predict	Global Average Pooling + Linear

TABLE I: Architecture of the visual and audio modules

second part of the vision branch. That is, in the second part of the architecture, we assume that certain feature representation $\mathbf{X}_v^{N\times d}$ has been extracted from input visual data, where N denotes the temporal dimension, and d denotes the feature dimension. Here $\mathbf{X}_v$ can be represented by any feature types, either deep features extracted using a pre-trained model, or other features commonly used for emotion recognition, such as facial action units or landmarks. We further apply a sequence of four convolutional blocks for learning a temporal representation. Each convolutional block consists of an 1D Convoluitonal layer with a 3×3 kernel, Batch Normalization, and a ReLU activation. Further details can be seen in Table 1 that provides full details of vision branch, where k denotes the kernel size, d denotes the number of filters in a convolutional layer, and s denotes the stride. The convolutional blocks are grouped into two stages for multimodal fusion described further.

2) Audio branch: Similarly to the vision branch, the audio branch operates on a feature representation, whether pre-computed or optimized jointly, and applies four blocks of 1D convolutional layers. Each block consists of a Convolutional layer, Batch Normalization, ReLU activation, and MaxPooling, with the specifications defined in Table 1. For audio, we

primarily use mel-frequency cepstral coefficients as features. We observed no benefit in using other feature representation types, such as chroma features or spectrograms.

B. Modality fusion approaches

In this section, we describe the considered fusion approaches. We will first describe the late transformer fusion approach that is similar to previous works described in the literature, and then describe the two proposed intermediate fusion approaches.

1) Late transformer fusion: In this setup, features learnt from two branches are fused with a transformer block. Specifically, we employ two transformers at the outputs of each branch, where fusion of one modality is performed into the other one. The outputs of these transformer blocks are further concatenated and passed to the final prediction layer. Formally, this can be defined as follows.

Let $\mathbf{\Phi}_a$ and $\mathbf{\Phi}_v$ be the feature representations of audio and vision modalities after the second feature extraction stage, i.e., after the fourth convolutional block. A transformer block is added in each branch, taking representations of two modalities as inputs. Considering the audio branch as an example, the transformer block takes the vision branch representation $\mathbf{\Phi}_v$ as input and projects it to obtain keys and values, while queries are computed from the audio branch features $\mathbf{\Phi}_a$. That is, self-attention is calculated as

$$A = softmax\left(\frac{\mathbf{\Phi}_a \mathbf{W}_q \mathbf{W}_k^T \mathbf{\Phi}_v^T}{\sqrt{d}}\right)\mathbf{\Phi}_v \mathbf{W}_v, \tag{2}$$

followed by standard transformer block processing [18]. The specific architecture of the transformer block is outlined in Figure 2.

$$\hat{\mathbf{\Phi}} = \mathbf{\Phi}_v \odot \mathbf{v}_v, \tag{3}$$

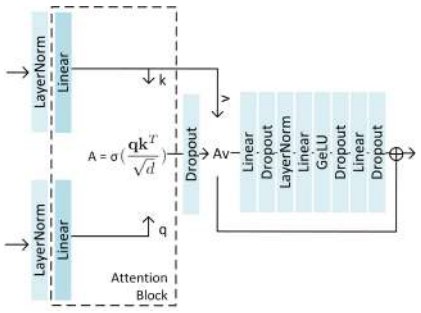


Fig. 2: Structure of the Transformer block.

The outputs of the two transformer blocks are concatenated and passed to the final layer for prediciton.

2) Intermediate transformer fusion: We propose the utilization of similar to the above-described transformer blocks for fusion at intermediate feature layers. Specifically, fusion is performed with a transformer block in each branch after the first stage of feature extraction, i.e., after two convolutional layers. Similar architecture to the one described in Figure 1 is used, and the fused feature representation is added to the corresponding branch.

Since data from complementary modality is introduced already at the earlier stage of the architecture, this allows to learn the features that are jointly meaningful for the task at hand between modalities during later convolutional layers.

3) Intermediate attention-based fusion: We further propose a fusion approach that is based merely on dot-product similarity that constitutes the attention in the transformer block. Formally, this is defined as follows. Given the two feature representations of different modalities $\mathbf{\Phi}_a$ and $\mathbf{\Phi}_b$, we compute queries and keys with learnt weights, similarly to conventional attention. The scaled dot-product similarity is subsequently calculated as

$$A = softmax\left(\frac{\mathbf{\Phi}_a \mathbf{W}_q \mathbf{W}_k^T \mathbf{\Phi}_v^T}{\sqrt{d}}\right). \tag{4}$$

Softmax activation promotes competition in the attention matrix, hence highlighting more important attributes/timestamps of each modality, and as a result providing the importance score of each key with respect to each query, i.e., each representation of modality a with respect to modality b. This allows to calculate the relative importance of each attribute of modality a by aggregating the scores corresponding to all the attributes of modality b for each attribute of modality a. As a result, we obtain an attention vector that can be used to highlight more relevant attributes of the modality a. Considering the dot-product attention between features of audio and vision modalities shown in Equation 3, attention vector of vision modality is given by $\mathbf{v}_v = \sum_{i=N_v} A[:, i]$.

Note that such a fusion approach does not directly fuse features of the two modalities. Instead, it identifies the attributes within each modality that are most relevant based to their similarity scores with data of the other modality. As a result, features that agree between the two modalities contribute the most to the final prediction, hence guiding the model towards learning modality-agnostic features or features with high level of agreement between the modalities. Such approach enables sharing of information between modalities, while not enforcing strong co-dependency of the learnt features in different branches as only attention scores are used for fusion.

C. Modality dropout

The vast majority of the multimodal learning methods described to date assume the presence of both modalities at all times during inference. Nevertheless, oftentimes in real-world applications data of one or more modalities might not be reliable or may be missing at times. In such scenarios, conventional multi-modal approaches tend to fail. Here, we aim to account for the potential cases of missing data and propose the modality dropout as a way of mitigating it. As will be shown further, utilization of this approach leads to improved performance also in situations where both modalities are present.

We propose the modality dropout, which randomly masks out or attenuates data of one of the modalities during training.

Specifically, we consider three variants. In the first variant, during training, data of one modality in each sample is randomly selected and replaced with zeros, while the representation of the other modality for a given sample is kept intact. This approach imitates missing data and can also act as a regularizer similarly to Dropout layer utilized in neural networks. Note that in the case of the third fusion approach and absence of bias terms, this results in zero dot similarity matrix in the attention block, which after softmax and summation leads to constant attention vector, hence no information transfer from the zeroed modality.

In the second variant, for each pair of data samples we generate a random scaling factor α in the range [0,1] [19] and multiply one of the modalities by α, while the other with $1 - \alpha$. The goal of this approach is to attenuate signals from different modalities at different training steps, and hence prevent the model from learning from strictly one modality. We further refer to this approach as 'soft' modality dropout. The third variant is aimed at the problem of noisy data, where the input signal of one modality is corrupted. Here, the masking is performed similarly to the first variant, except rather than zero-masking, the data is randomly generated from a normal distribution with zero mean and unit variance in one of the modalities for each sample.

IV. Experiments

In this section we describe the experimental protocol and the data used for assessing the performance of the proposed approaches. We report the results of the proposed model with three fusion variants, as well as recent multimodal emotion recognition methods, namely MULT [14] and multimodal transformer [16]. Note that both methods report results on datasets consisting of pre-extracted features. In addition, [14] considers three modalities, and [16] does not provide details on specific hyperparameters of the architecture, making direct comparison infeasible. We therefore adopt our feature extraction and compare with competing works only in terms of the fusion approaches described in these works. Specifically, to compare with [16] we employ a transformer block on top of our two convolutional branches that performs fusion either from audio to video, or in the opposite direction, and replace linear layers in the transformer block with 1D-convolutional ones. Regarding MULT [14], we want to compare with purely audiovisual model, so we remove the transformer blocks responsible for fusion from/to the language modality. This yields the architecture that is similar to our late transformer fusion, with additional single modality transformer blocks added in each branch. Other hyperparameters, such as latent space dimensionality, are kept identical between the methods for fair comparison. Similarly to the comparison with [14], we add our feature extraction blocks to the model. Unless otherwise specified, we use a single transformer block with single head everywhere to achieve a lightweight model. Naturally, better performance can be expected from adding additional blocks and parameters to the models. We used [20] for transformer block implementation.

Subsequently, we perform experiments with modality dropout in two settings. In the first one, we target the problem of missing data from one modality and apply both soft and hard modality dropout during training. That is, in this setting, in each batch the data consists of the pairs of full data, pairs without audio modality, pairs without video modality, and pairs multiplied with random coefficients as described above. We report the performance in the presence of both modalities (denoted by 'AV'), as well as in the full absence of one modality (denoted by 'A' or 'V' for presence of only audio and video modalities, respectively). We additionally report the average metric over the three modality settings (denoted by 'M') to simplify the comparison between methods. In the second setting, we consider robustness towards noise and apply the third variant of modality dropout during training, and replace one of the modalities with random noise during testing.

1) RAVDESS dataset: We choose RAVDESS dataset [21] primarily due to availability of raw data in this dataset, as opposed to others. The dataset consists of video recordings of 24 people speaking with different emotions and poses a task of classification of emotional states into 7 classes: calm, happy, sad, angry, fearful, surprise, and disgust. 60 video sequences were recorded for each actor, and we crop or zero-pad them to 3.6 seconds, which is the average sequence length. For audio processing, we extract 10 Mel-frequency cepstral coefficients for further processing. For visual data, we select 15 uniformly distributed frames from 3.6 second video, and crop the faces of actors using a face detection algorithm [22]. Images are resized into 224x224 pixels. We train the model on raw 15-frame videos. We transfer the weights of EfficientFace pre-trained on AffectNet dataset [4]. We split the data into training, validation and test sets ensuring that the identities of actors are not repeated across sets. Specifically, we used four actors for testing, four for validation, and 16 for training, and report the result averaged over five folds. The videos are scaled into [0,1] scale, and random horizontal flip and random rotation are used for data augmentation. All the models are trained for 100 epochs with SGD, learning rate of 0.04, momentum of 0.9, weight decay of 1e-3, and reduction of learning rate on plateau of 10 epochs.

2) CMU-MOSEI dataset: We additionally conduct experiments on CMU-MOSEI dataset. The dataset consists of 23,454 movie review video clips taken from YouTube and labeled by human annotators with a sentiment score in the range [-3..3]. Note that we only consider audio and visual modalities in our experiments. Since the dataset provides pre-extracted features rather than raw data (specifically, 35 facial action units are provided for vision modality and audio data is represented by mfccs, pitch tracking, glottal source and peak slope parameters resulting in 74 features), we omit the EfficientFace feature extraction in the vision branch and training is performed starting from convolutional blocks directly. We rely on the implementation of [14] for the experimental protocol on CMU-MOSEI dataset and adopt the training hyperparameters described in therein.

	RAVDESS. ACC				MOSEI. ACC				MOSEI. MAE			
	AV	A	V	M	AV	A	V	M	AV	A	V	M
LT1	**79.33**	19.83	36.41	**45.19**	63.89	48.70	62.85	58.48	0.806	0.840	1.063	0.903
LT4	76.42	**27.92**	30.00	44.78	66.56	62.63	53.16	60.78	0.806	0.839	0.831	0.825
IT1	76.41	21.16	18.33	38.63	**67.72**	37.14	62.87	55.91	**0.792**	0.843	0.809	0.815
IT4	78.50	20.33	17.33	38.72	64.91	62.60	62.85	63.45	0.817	0.840	0.832	0.830
IA1	76.00	18.58	22.83	39.13	64.94	62.08	62.86	63.29	0.802	**0.837**	0.806	0.815
IA4	77.41	20.66	29.83	42.63	**67.72**	**63.07**	**65.77**	**65.52**	0.794	**0.837**	**0.803**	**0.811**
TAV	77.75	24.25	13.33	38.44	64.94	62.08	62.86	62.18	0.814	0.841	1.093	0.916
TVA	76.00	15.16	**42.67**	44.61	66.48	37.15	56.96	53.53	0.809	0.852	0.838	0.833
MLT	74.16	22.33	35.42	43.97	62.90	62.85	64.44	63.40	0.804	0.838	0.804	0.815
MODALITY DROPOUT												
LT1	79.08	**59.16**	72.66	70.30	67.11	63.62	62.90	64.54	0.802	0.829	0.801	0.811
LT4	79.25	53.00	70.92	67.72	64.47	53.71	64.91	61.03	0.814	0.837	0.819	0.824
IT1	77.33	48.41	73.75	66.50	62.80	62.85	63.09	62.91	0.804	0.831	0.803	0.813
IT4	78.91	44.33	**74.92**	66.05	67.01	64.30	63.12	64.81	0.796	0.826	**0.797**	0.806
IA1	**81.58**	58.08	72.83	**70.83**	67.19	**64.52**	**64.91**	**65.54**	0.795	**0.816**	0.798	**0.803**
IA4	79.58	57.16	71.83	69.52	63.48	62.74	63.18	63.13	0.807	0.820	0.808	0.812
TAV	76.58	54.83	13.33	48.24	65.32	63.84	62.85	64.01	0.811	0.832	0.839	0.828
TVA	74.42	44.91	69.58	62.97	**67.61**	63.98	60.95	64.18	**0.793**	0.819	0.798	0.803
MLT	78.50	53.58	70.66	67.58	63.87	62.85	63.37	63.36	0.806	0.836	0.835	0.826
MODALITY DROPOUT with NOISE												
LT1	77.08	53.16	68.50	66.246	65.57	64.03	64.94	64.94	0.809	0.826	0.806	0.813
LT4	**80.33**	54.33	**73.00**	69.22	64.08	63.31	62.85	62.85	0.813	0.827	0.813	0.818
IT1	76.75	53.75	71.58	67.36	**68.16**	**65.98**	63.53	63.53	**0.799**	0.821	0.804	**0.808**
IT4	76.08	54.50	71.00	67.19	67.83	63.56	64.22	64.22	0.801	0.826	**0.802**	0.809
IA1	78.25	**58.25**	71.66	**69.38**	62.76	63.89	63.18	63.27	0.804	**0.819**	0.805	0.809
IA4	78.41	55.75	68.58	67.58	63.51	64.08	62.54	63.37	0.805	0.820	0.808	0.811
TAV	75.83	56.25	12.83	48.30	66.81	65.68	**65.60**	**66.03**	0.810	0.820	0.811	0.813
TVA	73.66	41.25	71.41	62.10	66.23	63.18	64.58	64.66	0.804	0.831	0.806	0.813
MLT	77.41	54.16	66.33	65.96	64.52	62.74	63.51	63.59	0.805	0.830	0.805	0.811

TABLE II: Performance of different fusion methods on RAVDESS and MOSEI.

	MOSEI. ACC				MOSEI. MAE			
	AV	A	V	M	AV	A	V	M
IT1	64.66	38.80	63.12	55.53	0.821	0.857	**0.803**	0.827
IT4	65.90	37.23	62.85	55.32	0.805	0.845	1.932	1.194
IA1	64.09	**62.85**	**63.42**	**63.45**	0.799	**0.838**	0.807	**0.815**
IA4	64.74	37.28	61.28	54.43	0.803	0.842	0.808	0.818
MLT	**67.66**	56.90	60.73	61.76	**0.787**	**0.838**	0.836	0.821
MODALITY DROPOUT								
IT1	65.41	62.85	64.06	64.11	0.805	0.838	0.805	0.816
IT4	66.57	64.78	**65.02**	65.45	0.792	**0.812**	**0.795**	**0.800**
IA1	**68.76**	**65.96**	63.92	**66.21**	**0.791**	0.815	0.799	0.802
IA4	66.18	64.67	64.22	65.02	0.794	0.815	0.801	0.803
MLT	66.12	65.41	63.62	65.05	0.801	0.831	0.808	0.813
MODALITY DROPOUT with NOISE								
IT1	64.72	54.07	**66.34**	61.71	0.798	0.839	0.797	0.812
IT4	64.69	64.33	61.83	63.61	0.801	0.826	0.799	0.808
IA1	**67.25**	**64.96**	64.74	**65.65**	0.794	**0.813**	0.799	0.802
IA4	63.40	63.23	62.85	63.16	0.806	0.820	0.806	0.811
MLT	66.18	64.19	64.39	64.92	**0.790**	**0.813**	**0.791**	**0.798**

TABLE III: Comparison with MULT [14].

A. Results and Discussion

Table II shows the results of the proposed approaches on the RAVDESS and MOSEI datasets. Here, 'LT1' and 'LT4' denote late transformer fusion with one and four heads, respectively, and similarly 'IT' denotes intermediate transformer fusion, 'IA' denotes intermediate attention fusion, 'TAV' and 'TVA' refers to the fusion approaches described in [16], and 'MULT' refers to [14]. We report categorical accuracy on RAVDESS dataset, and binary accuracy (positive vs negative sentiment) on MOSEI dataset, as well as Mean Average Error between the true and predicted sentiment scores.

As can be seen, in the setting without any type of dropout, late transformer fusion achieves the best result on RAVDESS dataset, while intermediate attention fusion achieves the best result on MOSEI dataset on both the accuracy and MAE metrics. Note that intermediate attention fusion is also the most lightweight fusion approach compared to any of the methods using full transformer blocks. At the same time, performance under the presence of only one modality is extremely poor on RAVDESS dataset. On MOSEI dataset the performance drop is not drastic in the majority of cases, likely due to the dataset consisting of already pre-extracted features, and hence guaranteeing presence of meaningful independent features in each modality even in the absence of the other one.

Further, it can be seen that utilization of modality dropout improves the performance drastically under incomplete data of one modality. This is the case for most fusion methods, while intermediate attention fusion benefits from it the most. Besides, the performance under the presence of both modalities is improved as well, with the best result on RAVDESS achieved by intermediate attention fusion. This is also the best result on this dataset among all methods and dropout settings. Similar conclusions can be made on MOSEI dataset; utilization of modality dropout improves the performance in both single-modality and two-modality case. Under the noisy setting, we still observe the intermediate attention fusion performing best on the average metric on RAVDESS.

To provide better comparison with state-of-the-art, we additionally compare with full MULT model (omitting the language modality), following the implementations provided by [14] and using their convolutional layers, transformer block implementations and other hyperparameters. Since in their implementation several dense layers are added after the fusion and prior to the output layer, we add similar dense layers to our model for fair comparison. The results are provided in Table III. As can be seen, while MULT outperforms the proposed intermediate fusion approaches in the vanilla setting with both modalities, intermediate fusion handles missing modalities better, and especially under the presence of modality dropouts. The best overall performance is achieved by intermediate attention fusion with the first modality dropout variant.

As can be seen, in the majority of the cases the best performance is achieved by the proposed intermediate attention fusion combined with one of the proposed dropout approaches. As in this approach no hard feature sharing is performed, the learnt feature representations are less likely to be co-dependent and therefore can be disentangled more easily, hence leading to better robustness of the model in incomplete data settings. This, in turn, leads to better generalization capabilities of the model overall, leading to improved performance also under the setting of both modalities.

V. Conclusion

We proposed a model for audiovisual emotion recognition that learns end-to-end and an attention-based fusion method. We evaluated the robustness of different modality fusion approaches under the absence of, or noise present in, one of the modalities and proposed an approach to improve the model's robustness. Importantly, the proposed approach also improves

the performance under the (ideal) standard setting where both modalities are present.

REFERENCES

[1] Z. Liu, M. Wu, W. Cao, L. Chen, J. Xu, R. Zhang, M. Zhou, and J. Mao, "A facial expression emotion recognition based human-robot interaction system," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 4, pp. 668–676, 2017.

[2] J. Chen, Y. Lv, R. Xu, and C. Xu, "Automatic social signal analysis: Facial expression recognition using difference convolution neural network," *Journal of Parallel and Distributed Computing*, vol. 131, pp. 97–102, 2019.

[3] A. Dzedzickis, A. Kaklauskas, and V. Bucinskas, "Human emotion recognition: Review of sensors and methods," *Sensors*, vol. 20, no. 3, p. 592, 2020.

[4] A. Mollahosseini, B. Hasani, and M. H. Mahoor, "Affectnet: A database for facial expression, valence, and arousal computing in the wild," *IEEE Transactions on Affective Computing*, vol. 10, no. 1, pp. 18–31, 2017.

[5] M. Soleymani, D. Garcia, B. Jou, B. Schuller, S.-F. Chang, and M. Pantic, "A survey of multimodal sentiment analysis," *Image and Vision Computing*, vol. 65, pp. 3–14, 2017.

[6] R. A. Khalil, E. Jones, M. I. Babar, T. Jan, M. H. Zafar, and T. Alhussain, "Speech emotion recognition using deep learning techniques: A review," *IEEE Access*, vol. 7, pp. 117 327–117 345, 2019.

[7] Z. Zhao, Q. Liu, and F. Zhou, "Robust lightweight facial expression recognition network with label distribution training," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 4, 2021, pp. 3510–3519.

[8] 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.

[9] K. Chumachenko, J. Raitoharju, A. Iosifidis, and M. Gabbouj, "Speed-up and multi-view extensions to subclass discriminant analysis," *Pattern Recognition*, vol. 111, p. 107660, 2021.

[10] Y. Wang, W. Huang, F. Sun, T. Xu, Y. Rong, and J. Huang, "Deep multimodal fusion by channel exchanging," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[11] F. Xiao, Y. J. Lee, K. Grauman, J. Malik, and C. Feichtenhofer, "Audiovisual slowfast networks for video recognition," *arXiv preprint arXiv:2001.08740*, 2020.

[12] Y. Xiao, F. Codevilla, A. Gurram, O. Urfalioglu, and A. M. López, "Multimodal end-to-end autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[13] F. Laakom, K. Chumachenko, J. Raitoharju, A. Iosifidis, and M. Gabbouj, "Learning to ignore: rethinking attention in cnns," *arXiv preprint arXiv:2111.05684*, 2021.

[14] Y.-H. H. Tsai, S. Bai, P. P. Liang, J. Z. Kolter, L.-P. Morency, and R. Salakhutdinov, "Multimodal transformer for unaligned multimodal language sequences," in *Proceedings of the conference. Association for Computational Linguistics. Meeting*, vol. 2019. NIH Public Access, 2019, p. 6558.

[15] D. Krishna and A. Patil, "Multimodal emotion recognition using cross-modal attention and 1d convolutional neural networks." in *Interspeech*, 2020, pp. 4243–4247.

[16] J. Huang, J. Tao, B. Liu, Z. Lian, and M. Niu, "Multimodal transformer fusion for continuous emotion recognition," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 3507–3511.

[17] L. Shu, J. Xie, M. Yang, Z. Li, Z. Li, D. Liao, X. Xu, and X. Yang, "A review of emotion recognition using physiological signals," *Sensors*, vol. 18, no. 7, p. 2074, 2018.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[19] X. Shen, X. Tian, T. Liu, F. Xu, and D. Tao, "Continuous dropout," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 9, pp. 3926–3937, 2017.

[20] R. Wightman, "Pytorch image models," https://github.com/rwightman/pytorch-image-models, 2019.

[21] S. R. Livingstone and F. A. Russo, "The ryerson audio-visual database of emotional speech and song (ravdess): A dynamic, multimodal set of facial and vocal expressions in north american english," *PloS one*, vol. 13, no. 5, p. e0196391, 2018.

[22] R. Gradilla, "Multi-task cascaded convolutional networks (mtcnn) for face detection and facial landmark alignment," *link]. Acessado em*, vol. 13, 2020.

8.9 Using Synthesized Facial Views for Active Face Recognition

The appended paper follows.

Using Synthesized Facial Views for Active Face Recognition

Efstratios Kakaletsis* and Nikos Nikolaidis

Department of Informatics, Artificial Intelligence & Information Analysis Laboratory, Aristotle University of Thessaloniki, Thessaloniki, GR-54124, Greece.

*Corresponding author(s). E-mail(s): ekakalets@csd.auth.gr;
Contributing authors: nnik@csd.auth.gr;

Abstract

Active perception / vision exploits the ability of robots to interact with their environment, for example move in space, towards increasing the quantity or quality of information obtained through their sensors and, thus, improving their performance in various perception tasks. Active face recognition is largely understudied in recent literature. Attempting to tackle this situation, in this paper, we propose an active approach that utilizes facial views produced by photorealistic facial image rendering. Essentially, the robot that performs the recognition selects the best among a number of candidate movements around the person of interest by simulating their results through view synthesis. This is accomplished by feeding the robot's face recognizer with a real world facial image acquired in the current position, generating synthesized views that differ by $\pm\boldsymbol{\theta}°$ from the current view and deciding, based on the confidence of the recognizer, whether to stay in place or move to the position that corresponds to one of the two synthesized views, in order to acquire a new real image with its sensor. Experimental results in three datasets verify the superior performance of the proposed method compared to the respective "static" approach, approaches based on the same face recognizer that involve face frontalization and synthesized views, as well as a state of the art active method.

Keywords: active vision ; active face recognition ; synthesized facial views; photorealistic facial synthesis

1 Introduction

In recent years, the robotics and vision communities have started researching more thoroughly the field of active vision / perception and exploration. Active perception methods try to obtain more, or better quality, information from the environment by actively choosing from where and how to observe it using a camera (or other sensors), in order to accomplish more effectively tasks such as 3D reconstruction [1, 2], [3], [4], [5] or object recognition [6], [7]. This could be achieved, for example, by moving a camera-equipped mobile robot, e.g. a wheeled robot or a UAV, in positions which offer different (and hopefully better) views of the object of interest. Although active 3D object reconstruction has attracted considerable interest, mainly towards solving the "next-best-view" problem (i.e. choosing the next viewing position in order to to obtain a detailed and complete 3D object model), active approaches for recognition tasks, especially for face recognition, are less frequent in the literature. Deep Learning has lately dominated face recognition research due to the superior performance achieved. However the vast majority of recognition methods adopt a static approach i.e., an approach that is based on an image acquired from a specific viewpoint, even in setups where an active approach could have been used. Indeed, face recognition can be combined with an active approach for controlling the movement of a camera-equipped robot towards capturing the face from more informative views and thus obtaining more robust results, at the expense of energy consumption and additional time needed. Synthesized views of faces, whose images were acquired through a camera, can be used for robot movement guidance in an active face recognition setup. Instead of having the robot move in a physical way for capturing a novel (and better) view, one can use a synthesized view as an aid towards choosing a new viewpoint and improving recognition through an acquisition procedure.

In this paper, we propose an active face recognition approach that utilizes facial views synthesized by photorealistic facial image rendering. Essentially, the camera-equipped robot that performs the recognition selects the best among a number of candidate physical 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, it feeds the face recognizer with this image as well as with synthesized views that differ by $\pm\theta^{\circ}$ from the current view. Subsequently, it either stays in the current 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 in the same manner, for this location, for one or more steps. Using synthesized facial views facilitates decision-making by providing estimates of what is to be expected (in terms of recognition accuracy) in a new robot position. The proposed method involves 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. This fact makes recognition challenging, but at the same time more easily applicable in a real-world scenario, since it does not require the existence of facial images acquired from different viewpoints in order to train a view-independent face recognizer.

The remainder of this paper is organized as follows. In Section II related work is presented, whereas in Section III we describe the details of the proposed method. In Section IV experiments conducted to measure the algorithm's performance are presented. Finally, Section V provides a discussion and conclusions.

2 Related Work

2.1 Active Computer Vision

A few recent active approaches for tasks such as object detection, recognition, 3-D reconstruction and manipulation are presented in this section. Additional methods can be found in the review paper [8] that deals in particular with the problem of view planning in robot active vision.

In [9], a robotic arm equipped with a depth camera captures information for a scene from several poses, towards understanding the environment and performing multiple object detection. Boundary Representation Models (B-Reps) are used to represent the objects. The world representation is initialized and, after generating a first set of object detection hypotheses, the approach tries to perform exploration in order to generate new hypotheses or validate existing ones. This is accomplished by finding regions of interest (regions to be inspected) and suitable new views, acquired by appropriate poses of the arm. A proof of concept using a KUKA LWR 4 arm is provided. As expected, the object recognition rate increases as the number of views increases.

In [10] the authors deal with the problem of reconstructing a scene while also identifying the objects in it using 3D scans and a dataset of 3D shapes. Towards this end, a 3D attention model is developed that selects the best views to scan from, as well as the most informative regions within in each view, so as to achieve object recognition. The region-level attention mechanism generates features which are fairly robust against occlusion. Temporal dependencies among consecutive views are encoded with deep recurrent networks.

A new approach, called 3D ShapeNets, for representing a 3D shape as a probability distribution of binary variables on a voxel grid, using a Convolutional Deep Belief Network is proposed in [11]. This representation supports joint object recognition and shape completion from depth maps and enables active object recognition through view planning. The model, learns the distribution of 3D shapes from different object categories and various poses from raw CAD data, while also discovering hierarchical compositional part representations.

Moreover, in [12], the authors present a novel methodology for optimizing a robot's vision sensor viewpoint and apply it in the tasks of object recognition and manipulation (grasping synthesis) in unstructured environments. The

algorithm uses extremum seeking control (ESC), which utilizes a task success criterion in a continuous optimization loop. In the case of object recognition, an image is captured by the robot's camera and supplied to the recognition algorithm. The algorithm generates a success rate value (probability of recognizing an object) that forms the main component of the objective function, which is to be maximized by the neural-network based ESC algorithm, towards generating velocity commands for the robot camera. The camera moves on a sphere (viewsphere) around the object, i.e., it points to the object all the time while keeping the distance fixed. The algorithm requires neither a task model nor training on offline image data for viewpoint optimization and is shown to be robust to occlusions.

In [13] another active vision-based object recognition approach is presented, among other contributions. More specifically, 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 the next viewpoint and incorporating it into a trajectory optimisation task.

In [14] a method for active object recognition that involves a deep CNN for the simultaneous prediction of the object label and the next action to be performed by the sensor so as to improve recognition performance is presented. The task is treated as a reinforcement learning problem and a generative model of object similarities is embedded in the network for encoding the state of the system. Other, older, active object recognition approaches, are reviewed in [15–17].

[1] deals with the problem of active object reconstruction. In there, a next-best-view planning scheme based on supervised deep learning is proposed. A properly trained three-dimensional convolutional neural network (3D-CNN) is used to predict the next-best-view position, given the current view.

Finally, in [18] a viewpoint planning strategy for 3D reconstruction with application in the reconstruction of blades is presented. The algorithm focuses on controlling surface overlap for the various views so as to allow for successful registration. OctoMaps are used towards this end and the method is tested in both simulation and real blade reconstruction.

2.2 Active Face Recognition

Despite the fact that active object recognition has attracted considerable interest in the computer vision and robotics communities, active face recognition has been scarcely studied. Such a simple method is described in [6] and comprises of a neural network-based face recognizer along with a decision making

controller that decides for the viewpoint changes. More specifically, a pre-trained VGG-Face CNN is used by the recognition module in order to extract facial image features and it is combined with a nearest-neighbor identity recognition criterion. The simple controller module can make three different decisions based on the uncertainty of the current result (i.e., the distance d between the input image and the closest image in the database of known persons): a) recognize the individual, if d is below a threshold t_1 b) disregard the individual as unknown, if d is above a threshold t_2 or c) reassess the subject by moving to a different viewpoint, if $t_1 < d < t_2$. The direction towards which the movement shall be performed in order to increase the probability of correct recognition is not studied by the authors.

The authors in [7] propose a deep learning-based active perception method for embedding-based face recognition and examine its behavior on a real multi-view face image dataset. The proposed approach can simultaneously extract discriminative embeddings, as well as predict the action that the robot must take (stay in place, move left or right by a certain amount, on a circle centered at the person) in order to get a more discriminative view.

2.3 Multi-view Facial Image Synthesis

A significant number of techniques for synthesizing facial images in novel views appeared in the last years since such images can have a number of applications, e.g., in improving face recognition accuracy. For example, since profile faces usually provide inferior recognition results compared to frontal faces, generative adversarial networks (GANs) based methods for the frontalization of profile facial images [19] or generation of other facial views [20] have been proposed for improving face recognition results.

A method for the generation of frontal views from any input view that utilizes a novel generative adversarial architecture called the Attention Selective Network (ASN) is described in [21]. Towards improving single-sample face recognition by both generating additional samples and eliminating the influence of external factors (illumination, pose), [22] presents an end-to-end network for the estimation of intrinsic properties of a facial image with recovery of albedo UV map and 3D facial shape. In [23], a facial image rendering technique is used both in the training and testing stages of a face recognition approach.

A method that produces photorealistic facial image views is described in [24]. 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 [25–27]), 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 [28] along with 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 for the

generation of the synthetic facial images required by the method proposed in this paper.

Although facial view synthesis can improve face recognition performance, active perception methods can be expected to provide better results, in cases where acquisition of additional "real' facial views is possible due to the existence of e.g. a wheeled robot.

3 Proposed Active Face Recognition Algorithm

3.1 Face Recognition

Let us denote as database subset G a set of training facial images for the persons that shall be recognized. Similarly, the facial images to feed the face recognizer are included in the query (test) set T. The face recognition library face.evoLVe [29] which contains many state-of-the-art deep face recognition models, is used. More specifically, an implementation of a certain face recognition approach of face.evoLVe from the OpenDr Toolkit [30, 31] was used. IR-50 (50 layers) [32] trained on MS-CELEB-1M using an ArcFace [33] loss head was used as the 512-dimensional feature extraction backbone.

For the database subset G, face detection, facial landmark extraction and face alignment was based on the face.evoLVe module that is based on MTCNN [34], whereas for the query images in T, these processing steps were based on RetinaFace [35, 36]. Face recognition is performed by a nearest-neighbor classifier that uses Euclidean distance in the 512-dimensional feature space to find the database facial image that best matches the query image.

Face recognition confidence $FRC \in [0, 1]$, is also evaluated based on the distance between the input query image and the nearest image in the database G. The FRC is given by the following formula:

$$FRC = 1 - \frac{distance}{threshold} \tag{1}$$

where *distance* is the euclidean distance of query facial image from the nearest neighbor image in the database G and *threshold* is the optimal threshold found by running a pairwise matching experiment on LFW [37].

3.2 Active Face Recognition Using Synthesized Views

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

Algorithm 1 Active Face Recognition Algorithm on Pseudocode

Input: I_r, $threshold$, θ°
Result: $Person_{ID}(I_r)$

$\alpha = Estimate\_View\_Angle(I_r)$
$I_s^- = Render(\alpha - \theta^\circ, I_r)$
$I_s^+ = Render(\alpha + \theta^\circ, I_r)$
$I = \underset{x \in \{I_r, I_s^-, I_s^+\}}{argmax}(FRC(x))$
if $I = I_r$ **then**
    $I_{ID} = I_r$
    go to 32
else
    if $I = I_s^+$ **then**
        $\theta_{incr} = +\theta^\circ$
    else
        $\theta_{incr} = -\theta^\circ$
    end if
end if

$I_r^{1step} = Move\_and\_Capture(\alpha + \theta_{incr})$
if $FRC(I_r^{1step}) > threshold$ **then**
    $I_{ID} = \underset{x \in \{I_r, I_r^{1step}\}}{argmax}(FRC(x))$
    go to 32
else
    $I_s^{2step} = Render(\alpha + 2 * \theta_{incr}, I_r^{1step})$
    if $FRC(I_s^{2step}) < FRC(I_r^{1step})$ **then**
        $I_{ID} = \underset{x \in \{I_r, I_r^{1step}\}}{argmax}(FRC(x))$
        go to 32
    else
        $I_r^{2step} = Move\_and\_Capture(\alpha + 2 * \theta_{incr})$
        $I_{ID} = \underset{x \in \{I_r, I_r^{1step}, I_r^{2step}\}}{argmax}(FRC(x))$
        go to 32
    end if
end if

$Person_{ID}(I_r) = Recognize(I_{ID})$

More specifically, given a facial query image I_r (subscript r stands for real), captured by the robot camera at the robot starting position, the face synthesis algorithm [24] is utilized to estimate the robot view angle α and then render/generate facial views in 2 different view angles i.e. $-\theta^\circ$ and $+\theta^\circ$ 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 (subscript s stands for synthetic). Then, the face recognizer is fed with these three images I_r, I_s^-, I_s^+ (one real, two synthetic ones). Depending on the image that obtained the biggest face recognition confidence FRC, the robot stays in its current position (if FRC was maximum in I_r) or physically moves $-\theta^\circ$ (or $+\theta^\circ$) (if FRC was maximum in I_s^- (or I_s^+)) and acquires through its camera a new real image I_r^- (or 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^- (or I_r^+) and the obtained FRC is compared to an experimentally evaluated threshold t. In case a high enough confidence was observed, the algorithm outputs the ID of the person it recognized in I_r^- (or I_r^+) and terminates. If not, it tries additional $+\theta^\circ$ steps (movements) in pan, in the same direction as the first step. In more detail, in this second step, it generates/synthesizes a facial view $-\theta^\circ$ (or $+\theta^\circ$) in pan from the current pan value (and the same tilt), denoted as I_s^{--} (or I_s^{++}), and evaluates (by calling the face recogniser) FRC on this synthetic image. If $FRC(I_r^-) > FRC(I_s^{--})$ (or $FRC(I_r^+) > FRC(I_s^{++})$) the algorithm decides that the robot shall stay in its current position, outputs the ID of the person it recognized in I_r^- (or I_r^+) and terminates. Otherwise, the robot physically moves $-\theta^\circ$ ($+\theta^\circ$) from its current position, acquires a new image I_r^{--} (I_r^{++}) and the algorithm outputs the ID of the person it recognized in this image. The procedure can be repeated for a number of additional steps (movements), until the predefined maximum number of steps is reached. The performance of the proposed procedure obviously depends on whether the synthesis algorithm [24] estimates with sufficient accuracy the view angle of the query image I_r and also on whether the synthesized views are of good quality. In order to limit the possibly negative effect of these factors on the performance of the algorithm (e.g. by leading it to move towards the wrong direction), the algorithm does not actually take a decision based on the last real image it has visited but does so based on the real image where it has obtained the maximum FRC value. In more detail, if the algorithm took one step of $-\theta^\circ$, it takes a decision using the real image I given by:

$$I = \underset{x \in \{I_r^-, I_r\}}{argmax} \left(FRC(x)\right) \tag{2}$$

or the equivalent expression that involves I_r^+, I_r, if a step of $+\theta^\circ$ has been taken. Similarly, if two steps of $-\theta^\circ$ each have been performed, the algorithms decides on the person ID using the real image I given by:

$$I = \underset{x \in \{I_r^{--}, I_r^-, I_r\}}{argmax} \left(FRC(x)\right) \tag{3}$$

or the equivalent expression that involves I_r^{++}, I_r^+, I_r, if two steps, of $+\theta^\circ$ each, have been taken. The pseudocode for the proposed method, when two steps are allowed, is presented in algorithm 1.

It should be noted that the actual recognition is always performed on 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 behind the proposed approach is that in case the initial robot position is far from a frontal or nearly frontal one, the algorithm will hopefully direct it to move towards a position which is closer to a frontal one. Obviously, the procedure can work, in the same way, for tilt.

Table 1 Face recognition accuracy results and comparison with the static approach and other variants

Method	HPID [38]	QMUL [39]	Synthetic(SD)[40]
Static *(only Queries)*	72.49 %	69.88%	66.95%
Proposed *(Active)* (2 steps)	82.12%	**85.57%**	68.35 %
Proposed *(Active)* (4 steps)	**88.10%**	82.85%	**80%**
Frontalization *(synthetic frontal views)*	80.75%	75.95%	66.10%
Static & Synthetic *(real & synthetic views)* (4 steps)	72.22%	66.35%	62.28%

Table 2 Comparison with [7]

Method	HPID [38]	QMUL [39]	Synthetic (SD) [40]
Proposed *(Active)* (2 steps)	82.88%	82.47%	85.00%
Proposed *(Active)* (4 steps)	**87.78%**	**84.59%**	**88.81%**
[7] *(Active)* (2 steps)	60.96%	69.94%	67.63%
[7] *(Active)* (4 steps)	61.30%	68.11%	70.41%

4 Performance Evaluation

For the evaluation of the proposed active face recognition approach, a number of experiments were conducted using the HPID dataset [38], the Queen Mary University of London Multi-view Face Dataset (QMUL)[39] and a Synthetic Dataset (SD) [40]. In all three datasets, images of all subjects were divided into two non-overlapping subsets: a database subset G (images that the face recognizer uses to decide upon the ID of the query image through the nearest neighbor classifier) and a query (test) subset T (these are meant to be the images captured by the robot camera in its initial position). This was done by choosing images with different pan ranges for G and T. With this setup we are simulating active recognition where the robot is moving only in the pan direction. Short descriptions of the three datasets are provided below.

4.1 HPID Dataset

The HPID dataset [38] is a head pose image database consisting of 2790 face images of 15 subjects captured by varying the pan and tilt angles from $-90°$ to

$+90°$, in $15°$ increments. Two series of images were captured for each person, (93 images in each series).

The database subset G (Figure 1) contains facial images with tilt in angles $[-30°, -15°, 0°, +15°, +30°]$ and pans $[-15°, 0°]$, i.e. only nearly frontal images. The query subset (Figure 2) contains face images with tilts $[-30°, -15°, 0°, +15°, +30°]$ and pans $[-90°, -75°, -60°, -45°, -30°]$. The selection of the range $[-90°... - 30°]$ in pan, instead of the full ($[-90°... - 30°]$ and $[+30°... + 90°]$) semi-circle, in this and the other two datasets, was just for simplicity. Similar results were obtained in experiments involving the full semi-circle.

Synthetic images generated from the "real" query images for use from our algorithm are depicted in Figure 3.

4.2 QMUL Dataset

Queen Mary University of London Multi-view Face Dataset (QMUL) [39] consists of automatically aligned, cropped and normalised face images of 48 persons. Images of 37 persons are in greyscale (dimensions: 100x100 pixels) whereas those of the remaining 11 subjects are in colour and of dimensions 56x56. For each person 133 facial images exist, covering a viewsphere of $-90°... + 90°$ in pan and $-30°... + 30°$ in tilt in $10°$ increments. For the Database split G, the facial images with pan in angles $[-10°, 0°]$ and tilt in angles $[-30°, ..., +30°]$ were used. The Query split T (test) includes images with pan in angles $[-90°, ..., -20°]$ and tilt in the range $[-30°, ..., +30°]$.

4.3 Synthetic Dataset

The Synthetic Dataset (SD) was generated using Unity's Perception package. It consists of 175422 cropped face images of 33 subjects taken at different environments, lighting conditions, camera distances and angles. In total, the dataset contains images for 8 environments, 4 lighting conditions, 7 camera distances (1m-4m) and 36 camera angles ($0 - 360°$ at $10°$ intervals). A subset of the dataset was used in the experiments. The subset included all 33 subjects in all environments and 1 lighting condition, at a camera distance of 1.0 m. For the Database split G, facial images with pan $[0°, +10°]$ and tilt $0°$ were used. The Query (test) split T included images with pan in the range $[+20°, ..., +90°]$ and tilt $0°$.

4.4 Results

Results (in terms of recognition accuracy) are presented in Table 1. The line marked "Static" 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. As can be seen, the proposed active method (lines "Proposed (Active), 2 steps" and "Proposed (Active), 4 steps", referring to the cases where the robot can move up to 2 or 4 times from its initial position in $\theta°$ increments) outperforms its static counterpart for both 2

and 4 algorithm steps, at the obvious expense of additional robot movements and time required to perform them. For the HPID and SD datasets the best performance is obtained for 4 steps of the algorithm and the absolute increase of accuracy with respect to the static version is 15.61% and 13.05% respectively, whereas for the QMUL dataset the best performance is obtained for 2 steps (increase of 15.69% compared to the static approach).

The proposed approach was also compared to the frontalization approach that is often used in face recognition when the recognizer is trained only on frontal views. In this case, the facial view synthesis algorithm [24] is used in order to generate a frontal (0° in pan) view from the input (query) image. This image is then provided to the recognizer. The results (line "Frontalization (synthetic frontal views)") show that although frontalization achieves improved performance with respect to the static approach in HPID and QMUL datasets and similar performance in SD, it is superseded by the proposed active approach. Indeed the best achieved results of the proposed approach correspond to an absolute increase in accuracy (with respect to the frontalization approach) of 7.35%, 9.62% and 13.9% for the HPID, QMUL and SD datasets respectively.

One can naturally wonder what is the benefit of introducing an active approach, that involves actual robot movement, over the use of synthetic images only. To answer this question we set up another experiment where for each (real) query image, captured at a view angle α we generate (where possible) 8 synthetic images at angles $\alpha \pm \theta^\circ, ..., \alpha \pm 4\theta^\circ$ around the query and feed them to the recognizer along with the query image. The result with the highest FRC is then adopted as the final decision. Results are presented in line "Static & Synthetic (real & synthetic views) (4 steps)". Obviously this approach is not viable, providing results inferior to those of the static case.

Finally, the proposed method was compared to the recent embedding-based active deep face recognition technique [7]. The experimental setup followed in [7] for the HPID datased, was used in all three datasets. More specifically, 75% of the subjects contained in each dataset was used to train the models of [7], while the remaining 25% were used for evaluating the trained models (test set). Since our approach requires no training, only the test set data were utilized in the experiments that involved it. Images in the test set were used to form the Database split G and the Query split T, in the same way (same range of pan and tilt angles) mentioned in Sections 4.1 to 4.3. Results are presented in Table 2. One can observe that the proposed method outperforms the method in [7] in both the 2 and 4 steps setups, achieving (in the 4 steps setup) an absolute increase in accuracy of 26.48%, 16.48% and 18.40% for the HPID, QMUL and SD datasets respectively.

Statistics regarding the steps taken by the proposed approach (4 steps) are presented in Table 3 for the SD dataset. Each row in this Table corresponds to the type of real image that the algorithm reached in its course, i.e., the number of steps it has taken towards the right or the left direction. These types are mentioned in the first column and follow the same naming conventions used

in Section 3.2. For example, the row marked I_r^+ includes statistics for cases where the algorithm (robot) moved by $+10°$ from its initial position (the one represented by the input query image). The pan angle increment from the initial position, the number of images and the percentage they represent over the total are presented for each case. The presented statistics show that in 24.34% of the cases the robot decided to stay in its initial position whereas in the remaining 75.66% it moved by $\pm 10°, .., \pm 40°$ (one to four steps). It shall be noted however that the decision on the ID of the depicted person is not necessarily obtained from the last position the robot has visited, due to the fact that the image with the maximum recognition confidence (FRC) is used for this purpose.

Table 3 Active Face Recognition Statistics (4 Steps, SD dataset): steps performed by the algorithm.

Image type	**Angle**	**# Images**	**Percentage**
I_r	$0°$	28	24.34%
I_r^+	$+10°$	5	4.347%
I_r^{++}	$+20°$	7	6.086%
I_r^{+++}	$+30°$	5	4.347%
I_r^{++++}	$+40°$	3	2.608%
I_r^-	$-10°$	52	45.217%
I_r^{--}	$-20°$	8	6.956%
I_r^{---}	$-30°$	4	3.478%
I_r^{----}	$-40°$	3	2.608%
Total	–	115	100%

The average number of movements that the algorithm instructs the robot to perform can be easily evaluated from statistics such as the ones presented in Table 3. The respective figures are presented in Table 4. Note that in case the robot decides to performs no movement (stay decision) the number of movements is obviously zero. As can be seen, when 4 steps are allowed, the algorithm instructs the robot to make, on average, from 0.76 to 1.17 movements, a fact that denotes that the time required for active recognition (time for the computations as well as the time for the robot to move) is relatively low and can be further lowered if only 2 steps are allowed.

Table 4 Active Face Recognition Statistics: average number of steps.

Method	**HPID [38]**	**QMUL [39]**	**SD [40]**
2 steps	0.82	0.6689	1.14
4 steps	0.89	0.7623	1.17

In addition, Table 5 presents statistics regarding the moves that the algorithm (equivalently the robot in a real situation) performs and whether these lead towards a frontal view, i.e., $0°$ in pan (which is something that might be expected since the recognised is trained on near frontal images) or away from

Table 5 Active Face Recognition Statistics: move type (4 steps)

Move Type	HPID [38]	QMUL [39]	SD [40]
towards frontal	**447**(59.67%)	57(4.9%)	**67**(58.26%)
stay	117(15.62%)	**573**(49.35%)	28(24.34%)
away from frontal	155(20.69%)	531(45.73%)	20(17.39%)
total	749	1161	115

such a view. The statistics for the HPID, show that in most cases (59.6%) the algorithm moves the robot towards a frontal view. However, in another 20.6% of the cases the robot moves away from the frontal position, which indicates that either the estimate for the view angle of the input (query) image provided by the view synthesis algorithm is rather inaccurate or that the generated synthetic views are in some cases of poor quality, causing the algorithm to err with respect to the direction it shall move the robot. A similar behavior can be observed in the SD dataset, whereas in QMUL in the majority of cases 49.35% the algorithm decides to stay in the initial position whereas it moves away from the frontal direction in 45.73% of the cases. Despite these issues, the algorithm manages to achieve good results in most cases.

5 Discussion and Conclusions

An active face recognition approach that utilizes facial views produced by facial image synthesis was presented in this paper. The camera-equipped robot that performs the recognition selects the best among a number of candidate physical movements around the person of interest by simulating their results through view synthesis. Experimental results show that the proposed method is superior to both its static version and face recognition that involves synthetically generated images. Moreover, it achieves significantly better results than the method in [7].

It shall be noted that certain assumptions were adopted in this paper and, furthermore, a number of issues were not fully addressed. First of all, the actual control of the robot in order to move in θ° increments around the person is not dealt with, being outside the scope of the paper. However, a rough estimate of the person position with respect to the robot would suffice to enable robot control. Also, it was assumed that the person being recognized remains relatively static during the recognition process, which can be a fair assumption if this process is brief. In case the person moves during this process, this shall be taken into account by the algorithm.

Moreover, it was assumed that there are no obstacles in the robot path. If this is not the case, these obstacles shall be detected (e.g. by a depth sensors) and taken into account when planning the next move. Furthermore, obstacles in the space between the robot and the person might occlude the person for certain robot-person relative positions. However, since the algorithm decides on the person's identity based on the acquired image where the recognizer obtained the largest recognition confidence, it is rather safe to assume that, in

Fig. 1 Samples from the database subset G of the HPID dataset, depicting real facial images of a subject with tilt angles $[-30°, -15°, 0°, 15°, 30°]$ and pans $[-15°, 0°, 15°]$.

Fig. 2 Samples from the query subset T depicting real facial images of a subject with tilts $[-30°, -15°, 0°, 15°, 30°]$ and pans $[-90°, -75°, -60°, -45°, -30°]$.

Fig. 3 Samples of synthetic facial images generated from the query subset T of the HPID dataset. Each row depicts the two synthetic images generated in pan angles $pan - 15°$, $pan + 15°$ from real images with pans $[-75°, -60°, -45°, -30°]$. Each row corresponds to a different tilt value of the real image, in the range $[-30°, -15°, 0°, 15°, 30°]$.

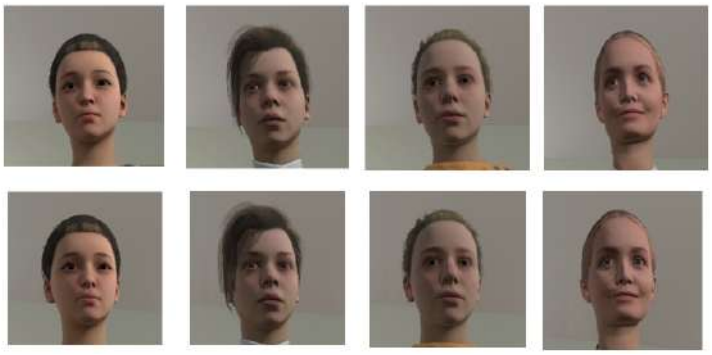


Fig. 4 Samples from the database subset G of the SD dataset, depicting facial images of four subjects with tilt angle $0°$ and pans $[0°, +10°]$.

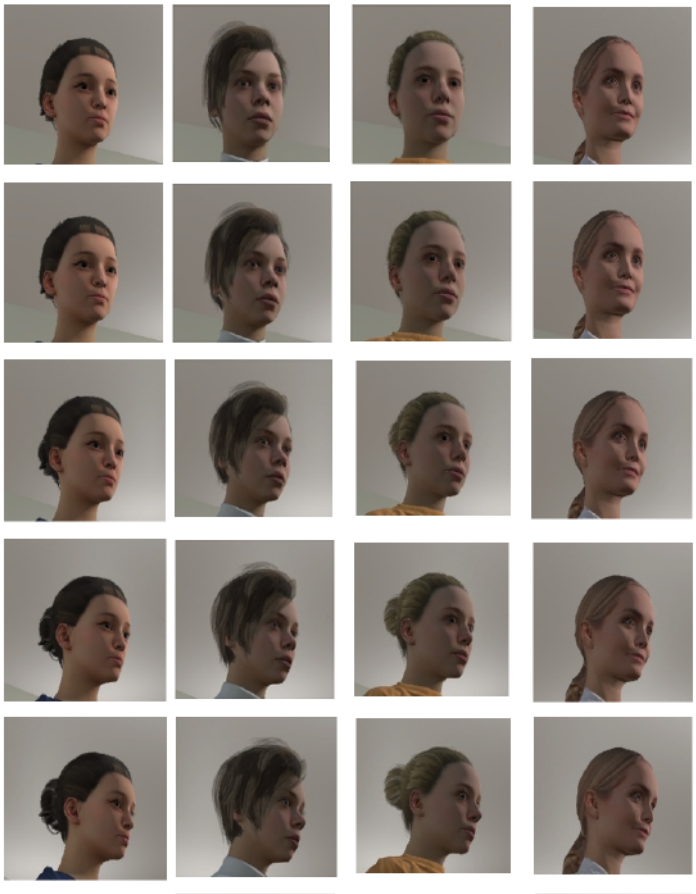


Fig. 5 Samples from the query subset T of SD Dataset depicting facial images of four subjects with tilt $0°$ and pans $[+20°, +30°, +40°, +50°, +60°]$.

most such cases, the algorithm might not face serious problems, even if it has instructed the robot to move in positions where occlusions occur.

One could also argue that, instead of using the synthesized views as proposed in this paper, it would suffice to estimate the view angle of the robot with respect to the person and instruct it to move directly (i.e., without intermediate steps) to the position that would allow it to obtain a frontal view ($0°$ in pan). However, there are certain difficulties that would make such an

approach hard to implement in practice. Indeed, we observed during the experiments that view angle estimates (at least those provided by the view synthesis algorithm used in this paper) although accurate enough for the purposes of view synthesis, are quite far from the ground truth values, thus deeming this approach rather problematic. Moreover, moving the robot from a view angle that differs significantly from frontal (e.g. 75°) to 0° is costly in time and energy and also unnecessary, since accurate enough recognitions might be obtained from other view positions, reachable by smaller movements.

Regarding algorithm performance, as mentioned in the previous section, there is a significant number of cases where the algorithm instructs the robot to move in a direction that is not towards a more frontal view. This might be attributed to errors of the view angle estimation and view synthesis algorithms. Using a better algorithm of this type might possibly lead to even bigger improvements with respect to the static approach. Another useful observation is that, giving the robot the freedom to move for additional steps (4 instead of 2) does, in two of the three datasets, significantly improve the recognition accuracy.

In the future, we plan to evaluate the proposed algorithm in larger datasets, and by creating a realistic simulation in an appropriate environment, e.g. in Unity, or Webots [41, 42], so as to investigate some of the issues mentioned above (occlusions, actual robot control, objects that hinder robot motion etc). Comparing our approach to additional methods is also planned.

Acknowledgments

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 871449 (OpenDR). This publication reflects only the authors views. The European Union is not liable for any use that may be made of the information contained therein.

References

[1] Miguel Mendoza, J Irving Vasquez-Gomez, Hind Taud, L Enrique Sucar, and Carolina Reta. Supervised learning of the next-best-view for 3D object reconstruction. *Pattern Recognition Letters*, 133:224–231, 2020.

[2] Jeffrey Delmerico, Stefan Isler, Reza Sabzevari, and Davide Scaramuzza. A comparison of volumetric information gain metrics for active 3D object reconstruction. *Autonomous Robots*, 42(2):197–208, 2018.

[3] Stefan Isler, Reza Sabzevari, Jeffrey Delmerico, and Davide Scaramuzza. An information gain formulation for active volumetric 3D reconstruction. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 3477–3484. IEEE, 2016.

[4] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Appearance-based active, monocular, dense reconstruction for micro aerial vehicles. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

[5] J Irving Vasquez-Gomez, David Troncoso, Israel Becerra, Enrique Sucar, and Rafael Murrieta-Cid. Next-best-view regression using a 3d convolutional neural network. *Machine Vision and Applications*, 32(2):1–14, 2021.

[6] Masaki Nakada, Han Wang, and Demetri 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.

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

[8] Rui Zeng, Yuhui Wen, Wang Zhao, and Yong-Jin Liu. View planning in robot active vision: A survey of systems, algorithms, and applications. *Computational Visual Media*, 6(3):225–245, 2020.

[9] Dorian Rohner and Dominik Henrich. Using active vision for enhancing an surface-based object recognition approach. In *Proceedings of Fourth IEEE International Conference on Robotic Computing (IRC)*, pages 375–382. IEEE, 2020.

[10] Kai Xu, Yifei Shi, Lintao Zheng, Junyu Zhang, Min Liu, Hui Huang, Hao Su, Daniel Cohen-Or, and Baoquan Chen. 3D attention-driven depth acquisition for object identification. *ACM Transactions on Graphics (TOG)*, 35(6):1–14, 2016.

[11] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015.

[12] Berk Calli, Wouter Caarls, Martijn Wisse, and Pieter P Jonker. Active vision via extremum seeking for robots in unstructured environments: Applications in object recognition and manipulation. *IEEE Transactions on Automation Science and Engineering*, 15(4):1810–1822, 2018.

[13] Edward Johns, Stefan Leutenegger, and Andrew J Davison. Pairwise decomposition of image sequences for active multi-view recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern*

Recognition (CVPR), pages 3813–3822, 2016.

[14] Mohsen Malmir, Karan Sikka, Deborah Forster, Ian Fasel, Javier R Movellan, and Garrison W Cottrell. Deep active object recognition by joint label and action prediction. *Computer Vision and Image Understanding*, 156:128–137, 2017.

[15] Shengyong Chen, Youfu Li, and Ngai Ming Kwok. Active vision in robotic systems: A survey of recent developments. *The International Journal of Robotics Research*, 30(11):1343–1377, 2011.

[16] Sumantra Dutta Roy, Santanu Chaudhury, and Subhashis Banerjee. Active recognition through next view planning: a survey. *Pattern Recognition*, 37(3):429–446, 2004.

[17] GCHE de Croon, Ida G Sprinkhuizen-Kuyper, and Eric O Postma. Comparing active vision models. *Image and Vision Computing*, 27(4):374–384, 2009.

[18] Weixing Peng, Yaonan Wang, Zhiqiang Miao, Mingtao Feng, and Yongpeng Tang. Viewpoints planning for active 3-d reconstruction of profiled blades using estimated occupancy probabilities (EOP). *IEEE Transactions on Industrial Electronics*, 68(5):4109–4119, 2020.

[19] Qingyan Duan and Lei Zhang. Look more into occlusion: Realistic face frontalization and recognition with boostgan. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):214 – 228, January 2021.

[20] Rui Huang, Shu Zhang, Tianyu Li, and Ran 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, (ICCV)*, pages 2439–2448, 2017.

[21] Jiashu Liao, Alex Kot, Tanaya Guha, and Victor Sanchez. Attention selective network for face synthesis and pose-invariant face recognition. In *Proceedings of IEEE International Conference on Image Processing (ICIP)*, pages 748–752. IEEE, 2020.

[22] Huan Tu, Gesang Duoji, Qijun Zhao, and Shuang Wu. Improved single sample per person face recognition via enriching intra-variation and invariant features. *Applied Sciences*, 10(2):601, 2020.

[23] Iacopo Masi, Tal Hassner, Anh Tuân Tran, and Gérard Medioni. Rapid synthesis of massive face sets for improved face recognition. In *Proceedings of 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, pages 604–611. IEEE, 2017.

[24] Hang Zhou, Jihao Liu, Ziwei Liu, Yu Liu, and Xiaogang 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 (CVPR)*, pages 5911–5920, 2020.

[25] Jianzhu Guo, Xiangyu Zhu, and Zhen Lei. 3DDFA. https://github.com/cleardusk/3DDFA, 2018.

[26] Jianzhu Guo, Xiangyu Zhu, Yang Yang, Fan Yang, Zhen Lei, and Stan Z Li. Towards fast, accurate and stable 3D dense face alignment. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 152–168. Springer International Publishing, 2020.

[27] Xiangyu Zhu, Xiaoming Liu, Zhen Lei, and Stan Z Li. Face alignment in full pose range: A 3d total solution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(1):78–92, 2017.

[28] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3D mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3907–3916, 2018.

[29] Qingzhong Wang, Pengfei Zhang, Haoyi Xiong, and Jian Zhao. Face. evolve: A high-performance face recognition library. *arXiv preprint arXiv:2107.08621*, 2021.

[30] Nikolaos Passalis, Stefania Pedrazzi, Robert Babuska, Wolfram Burgard, Daniel Dias, Francesco Ferro, Moncef Gabbouj, Ole Green, Alexandros Iosifidis, Erdal Kayacan, Jens Kober, Olivier Michel, Nikos Nikolaidis, Paraskevi Nousi, Roel Pieters, Maria Tzelepi, Abhinav Valada, and Anastasios Tefas. OpenDR: An Open Toolkit for Enabling High Performance, Low Footprint Deep Learning for Robotics. *arXiv preprint arXiv:2203.00403*, 2022.

[31] OpenDR: A modular, open and non-proprietary toolkit for core robotic functionalities by harnessing deep learning. https://github.com/opendr-eu/opendr. Accessed: 2022-06-27.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[33] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4690–4699, 2019.

[34] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE signal processing letters*, 23(10):1499–1503, 2016.

[35] Jiankang Deng, Jia Guo, Evangelos Ververas, Irene Kotsia, and Stefanos Zafeiriou. Retinaface: Single-shot multi-level face localisation in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5203–5212, 2020.

[36] OpenDR Face Detection module: RetinaFace. https://github.com/opendr-eu/opendr/blob/master/docs/reference/face-detection-2d-retinaface.md. Accessed: 2022-06-27.

[37] Gary B. Huang Erik Learned-Miller. Labeled faces in the wild: Updates and new reporting procedures. Technical Report UM-CS-2014-003, University of Massachusetts, Amherst, May 2014.

[38] Nicolas Gourier, Daniela Hall, and James L Crowley. Estimating face orientation from robust detection of salient facial structures. In *Proceedings of Workshop on Visual Observation of Deictic Cestures*, volume 6, page 7. FGnet (IST–2000–26434) Cambridge, UK, 2004.

[39] Jamie Sherrah and Shaogang Gong. Fusion of perceptual cues for robust tracking of head pose and position. *Pattern Recognition*, 34(8):1565–1572, 2001.

[40] Charalampos Georgiadis. Generation of a synthetic annotated dataset for training and evaluating active perception methods. *BSc Thesis, Aristotle University of Thessaloniki*, 2022, doi : 10.13140/RG.2.2.21002.34248.

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

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

8.10 Active Face Recognition through View Synthesis

The appended paper follows.

Active Face Recognition through View Synthesis

Efstratios Kakaletsis, Nikos Nikolaidis
Department of Informatics, Artificial Intelligence & Information Analysis Laboratory
Aristotle University of Thessaloniki
Thessaloniki, Greece, GR-54124
Email: {ekakalets, nnik }@csd.auth.gr

***Abstract*—Active vision exploits the ability of robots to interact with their environment, towards increasing the quantity / quality of information obtained through their sensors and, therefore, improving their performance in perception tasks. Active face recognition is largely understudied in recent literature. In this paper, we propose an active approach that utilizes facial views produced by facial image rendering. The robot that performs the recognition selects the best among a number of candidate movements around the person of interest by simulating their results through view synthesis. This is achieved by passing to the robot's face recognizer a real world facial image acquired in the current position, generating synthesized views that differ by $\pm\theta^\circ$ from the current view and deciding, on the basis of the confidence of the recognizer, whether to stay in place or move to the position that corresponds to one of the two synthesized views, so as to to acquire a new real image. Experimental results in two datasets verify the superior performance of the proposed method compared to the respective "static" approach and an approach based on the same face recognizer that involves face frontalization with synthesized views.**

I. Introduction

Recently the robotics and computer vision communities have started researching more thoroughly the field of active vision / perception and exploration [1]. Active perception methods try to obtain more, or better quality, information from the environment by actively choosing from where, when and how to observe it using a camera (or other sensors), in order to accomplish more effectively tasks such as 3D reconstruction [2], [3], [4], [5], [6] or object recognition [7], [8]. This could be achieved, for example, by moving a camera-equipped mobile robot, e.g. a wheeled robot or a UAV, in positions which provide different, hopefully better, views of the object of interest. Although active 3D object reconstruction has attracted considerable interest, mainly towards tackling the "next-best-view" problem (choosing the next viewing position so as to obtain a detailed and complete 3D object model), active approaches for recognition tasks, particularly for face recognition, are much less frequent in the literature. Deep Learning dominates face recognition research due to its superior performance. However the vast majority of recognition approaches adopt a static approach i.e., an approach that is based on an image acquired from a certain viewpoint, even in setups where an active approach could have been used. Indeed, face recognition can be combined with an active approach for directing the movement of a robot towards capturing the face from more informative views and thus obtain more robust results, at the expense of energy consumption and additional decision time. Synthesized views of faces, whose images were captured through a camera, can be used for robot movement guidance in an active face recognition setup. Instead of having the robot move in a physical way for capturing a novel view, one can use a synthesized view as an aid towards choosing a new viewpoint and improving recognition.

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 871449 (OpenDR). This publication reflects only the authors views. The European Union is not liable for any use that may be made of the information contained therein.

In this paper, we propose an active face recognition approach that utilizes facial views synthesized by photorealistic facial image rendering. Essentially, the camera-equipped robot that performs the recognition selects the best among a number of candidate physical 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, it provides the face recognizer with this image as well as with synthesized views that differ by $\pm\theta^\circ$ from the current view. Subsequently, it either stays in the current 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 the case of a "move" decision, it proceeds to acquire a "real" image from its new location. The procedure repeats in the same manner, for this location, for one or more steps. Using synthesized facial views facilitates the decision-making procedure by providing estimates of what is to be expected in a new robot position. The proposed method involves 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. This fact makes recognition challenging, but at the same time more easily applicable in a real-world scenario, since it does not require the existence of facial images acquired from different viewpoints in order to train a view-independent face recognizer.

The remainder of this paper is organized as follows. Section II presents related work, whereas in Section III we describe the details of the proposed method. In Section IV experiments conducted to measure the algorithm's performance are presented. Finally, Section V provides a discussion and conclusions.

II. Related Work

Despite the fact that active object recognition has attracted considerable interest in the computer vision and robotics communities [9], [10],[11], active face recognition has been scarcely studied. Such a simple method is described in [7] and comprises of a neural network-based face recognizer along with a decision making controller that decides for the viewpoint changes. More specifically, a pre-trained VGG-Face CNN is used by the recognition module in order to extract facial image features and it is combined with a nearest-neighbor identity recognition criterion. The simple controller module can make three different decisions based on the uncertainty of the current result (i.e., the distance d between the input image and the closest image in the database of known persons): a) recognize the individual, if d is below a threshold t_1 b) disregard the individual as unknown, if d is above a threshold t_2 or c) reassess the subject by moving to a different viewpoint, if $t_1 < d < t_2$. The direction towards which the movement shall be performed in order to increase the probability of correct recognition is not studied by the authors. The

authors in [8] propose a deep learning-based active perception method for embedding-based face recognition and examine its behavior on a real multi-view face image dataset. The proposed approach can simultaneously extract discriminative embeddings, as well as predict the action that the robot must take (stay in place, move left or right by a certain amount, on a circle centered at the person) in order to get a more discriminative view.

A significant number of techniques for synthesizing facial images in novel views appeared in the last years since such images can have a number of applications, e.g., in improving face recognition accuracy. For example, since profile faces usually provide inferior recognition results compared to frontal faces, generative adversarial networks (GANs) based methods for the frontalization of profile facial images [12] or generation of other facial views [13] have been proposed for improving face recognition results. In order to improve face recognition accuracy, a significant number of techniques for synthesizing facial images in novel views appeared in the last years since such images can have substantial impact on these applications. For example, since profile faces usually provide inferior recognition results compared to frontal faces, generative adversarial networks (GANs) based methods for the frontalization of profile facial images [12] or generation of other facial views [13] have been proposed for augmenting the face recognition results. A method for the generation of frontal views from any input view that utilizes a novel generative adversarial architecture called the Attention Selective Network (ASN) is described in [14]. Towards improving single-sample face recognition by both generating additional samples and eliminating the influence of external factors (illumination, pose), [15] presents an end-to-end network for the estimation of intrinsic properties of a facial image with recovery of albedo UV map and 3D facial shape. In [16], a facial image rendering technique is used both in the training and testing stages of a face recognition approach. A method that produces photorealistic facial image views is described in [17]. 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 [18], [19], [20]), 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 [21] along with 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 for the generation of the synthetic facial images required by the method proposed in this paper.

Although facial view synthesis can improve face recognition performance, active perception methods can be expected to provide better results, in cases where acquisition of additional "real' facial views is possible due to the existence of e.g. a wheeled robot.

III. Proposed Algorithm

A. Face Recognition

Let us denote as database subset G a set of training facial images for the persons that shall be recognized. Similarly, the facial images to feed the face recognizer are included in the query (test) set T. The face recognition library face.evoLVe [22] which contains many state of the art deep face recognition models, is used. More specifically, an implementation of a certain face recognition approach of face.evoLVe from the OpenDr Toolkit [23], [24] was used. IR-50 (50 layers) [25] trained on MS-CELEB-1M using an ArcFace [26] loss head was used as the 512-dimensional feature extraction backbone. For the database subset G, face detection, facial landmark extraction and face alignment was based on the face.evoLVe module that is based on MTCNN [27], whereas for the query images in T, these processing steps were based on RetinaFace [28]. Face recognition is performed by a nearest-neighbor classifier that uses Euclidean distance in the 512-dimensional feature space to find the database facial image that best matches the query image.

Face recognition confidence $FRC \in [0, 1]$, is also evaluated based on the distance between the input query image and the nearest image in the database G. The FRC is given by the following formula:

$$FRC = 1 - \frac{distance}{threshold} \tag{1}$$

where $distance$ is the euclidean distance of query facial image from the nearest neighbor image in the database G and $threshold$ is the optimal threshold found by running a pairwise matching experiment on LFW [29].

B. Active Face Recognition Through Synthesized Views

The proposed active face recognition algorithm uses the face recognition confidence FRC and facial images synthesized for view angles around the current robot view, in order to select the next robot movement, towards performing a successful recognition. Starting from an initial position, the robot can take one of the following three decisions: stay at the current position, move by θ° to the right or move by θ° 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 (subscript r stands for real), captured by the robot camera at the robot starting position, the face synthesis algorithm [17] is utilized to estimate the view angle and then render/generate facial views in 2 different view 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 (subscript s stands for synthetic). Then, the face recognizer is fed with these three images I_r, I_s^-, I_s^+ (one real, two synthetic ones). Depending on the image that obtained the biggest face recognition confidence FRC, the robot stays in its current position (if FRC was maximum in I_r) or physically moves -15° (or $+15^\circ$) (if FRC was maximum in I_s^- (or I_s^+)) and acquires through its camera a new real image I_r^- (or 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^- (or I_r^+) and the obtained FRC is compared to an experimentally evaluated threshold t. In case a high enough confidence was observed, the algorithm outputs the ID of the person it recognized in I_r^- (or 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, in this second step, it generates/synthesizes a facial view -15° (or $+15^\circ$) in pan from the current pan value (and the same tilt), denoted as I_s^{--} (or I_s^{++}), and evaluates (by calling the face recogniser) FRC on this synthetic image. If $FRC(I_r^-) > FRC(I_s^{--})$ (or $FRC(I_r^+) > FRC(I_s^{++})$) the algorithm decides that the robot shall stay in its current position, outputs the ID of the person it recognized in I_r^- (or I_r^+) and terminates. Otherwise, the robot physically moves -15° ($+15^\circ$) from its current position, acquires a new image I_r^{--} (I_r^{++}) and the algorithm outputs the ID of the person it recognized in this image.

The performance of the proposed procedure obviously depends on whether the synthesis algorithm [17] estimates with sufficient accuracy the view angle of the query image I_r and also on whether the synthesized views are of good quality. In order to limit the

Algorithm 1 Active Face Recognition Algorithm (2 steps) on Pseudocode

Input: I_r, $threshold$, θ°
Result: $Person_{ID}(I_r)$

$\alpha = Estimate\_View\_Angle(I_r)$
$\mathrm{I}_s^- = Render(\alpha - \theta^\circ, I_r)$
$\mathrm{I}_s^+ = Render(\alpha + \theta^\circ, I_r)$
$I = \underset{x \in \{I_r, I_s^-, I_s^+\}}{argmax}(FRC(x))$
if $I = I_r$ **then**
  $I_{ID} = I_r$
  go to 28
else
  if $I = I_s^+$ **then**
    $\theta_{incr} = +\theta^\circ$
  else
    $\theta_{incr} = -\theta^\circ$

$\mathrm{I}_r^{1step} = Move\_and\_Capture(\alpha + \theta_{incr})$
if $FRC(I_r^{1step}) > threshold$ **then**
  $\mathrm{I}_{ID} = \underset{x \in \{I_r, I_r^{1step}\}}{argmax}(FRC(x))$
  go to 28
else
  I_s^{2step}=$Render(\alpha + 2 * \theta_{incr}, I_r^{1step})$
  if $FRC(I_s^{2step}) < FRC(I_r^{1step})$ **then**
    $I_{ID} = \underset{x \in \{I_r, I_r^{1step}\}}{argmax}(FRC(x))$
    go to 28
  else
    $I_r^{2step} = Move\_and\_Capture(\alpha + 2 * \theta_{incr})$
    $I_{ID} = \underset{x \in \{I_r, I_r^{1step}, I_r^{2step}\}}{argmax}(FRC(x))$
    go to 28

$Person_{ID}(I_r) = Recognize(I_{ID})$

possibly negative effect of these factors on the performance of the algorithm (e.g. by leading it to move towards the wrong direction), the algorithm does not actually take a decision based on the last real image it has visited but does so based on the real image where it has obtained the maximum FRC value. In more detail, if the algorithm took one step of -15°, it takes a decision using the real image I given by:

$$I = \underset{x \in \{I_r^-, I_r\}}{argmax}\,(FRC(x)) \tag{2}$$

or the equivalent expression that involves I_r^+, I_r, if a step of $+15^\circ$ has been taken. Similarly, if two steps of -15° each have been performed, the algorithms decides on the person ID using the real image I given by:

$$I = \underset{x \in \{I_r^{--}, I_r^-, I_r\}}{argmax}\,(FRC(x)) \tag{3}$$

or the equivalent expression that involves I_r^{++}, I_r^+, I_r, if two steps, of $+15^\circ$ each, have been taken. The pseudocode is presented in algorithm 1.

It should be noted that the actual recognition is always performed on 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 behind the proposed approach is that in case the initial robot position is far from a frontal or nearly frontal one, the algorithm will hopefully direct it to move towards a position 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.

IV. Experimental Evaluation

For the evaluation of the proposed active approach experiments were conducted using the HPID dataset [30] and the Queen Mary University of London Multi-view Face Dataset (QMUL)[31]. In the two datasets, images of all subjects were divided into two non-overlapping subsets: a database subset G (images that the face recognizer uses to decide the ID of the query image through the nearest neighbor classifier) and a query (test) subset T (which includes the images captured by the robot camera in its initial position). Obviously G and T contained images from different pan ranges. This setup was adopted in order to simulate active recognition where the robot is moving only in the pan direction. Concise descriptions of the two datasets are provided below.

A. Datasets

The HPID dataset [30] is a head pose image dataset that consists of 2790 face images of 15 subjects captured by varying the pan and tilt from -90° to $+90^\circ$, in increments of $\theta = 15^\circ$. Two sets of images were captured for each person, (93 images in each set).

The database subset G (Figure 1) contains facial images with tilt in angles $[-30^\circ, -15^\circ, 0^\circ, +15^\circ, +30^\circ]$ and pans $[-15^\circ, 0^\circ]$, i.e., only nearly frontal images. The query subset contains face images with tilts $[-30^\circ, -15^\circ, 0^\circ, +15^\circ, +30^\circ]$ and pans $[-90^\circ, -75^\circ, -60^\circ, -45^\circ, -30^\circ]$. The selection of the range $[-90^\circ ... -30^\circ]$ in pan, instead of the full (i.e., $[-90^\circ ... -30^\circ]$ and $[+30^\circ ... +90^\circ]$) semi-circle, in this and the QMUL dataset, was just for simplicity. Similar results were obtained when the experiments involved the entire semi-circle.

Fig. 1. Samples from the database subset G of the HPID dataset, depicting real facial images of a subject with tilt angles $[-30^\circ, -15^\circ, 0^\circ, 15^\circ, 30^\circ]$ and pans $[-15^\circ, 0^\circ, 15^\circ]$.

Queen Mary University of London Multi-view Face Dataset (QMUL) [31] consists of automatically aligned, cropped and normalised face images of 48 persons. Images of 37 persons are in greyscale (100x100 pixels) whereas those of the remaining 11 persons are in colour and of dimensions 56x56 pixels. For each person 133 facial images exist, populating a viewsphere of $-90^\circ ... +90^\circ$ in pan and $-30^\circ ... +30^\circ$ in tilt in $\theta = 10^\circ$ increments. For the Database split G, images with pan in angles $[-10^\circ, 0^\circ]$ and tilt in the range $[-30^\circ, ..., +30^\circ]$ were used. The Query split T includes images with pan in angles $[-90^\circ, ..., -20^\circ]$ and tilt in the range $[-30^\circ, ..., +30^\circ]$.

B. Experimental Results

The results (in terms of recognition accuracy) are presented in Table I. The line marked "Static" 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. As can be seen, the proposed active method, implemented so as to perform up to 4 steps (line "Proposed (Active) (4 steps)") outperforms its static counterpart, increasing the recognition accuracy by 15.61% and 12.97% (absolute increase) in HPID and QMUL datasets, respectively. -

TABLE I
FACE RECOGNITION ACCURACY RESULTS AND COMPARISON WITH THE STATIC APPROACH AND OTHER VARIANTS

Method	HPID [30]	QMUL [31]
Static *(only Queries)*	72.49 %	69.88%
Proposed *(Active)* (4 steps)	**88.10%**	**82.85%**
Frontalization *(synthetic frontal views)*	80.75%	75.95%

The proposed approach was also compared to the frontalization approach that is often used in face recognition when the recognizer is trained only on frontal views. In this case, the facial view synthesis algorithm [17] is used in order to generate a frontal (0° in pan) view from the input (query) image. This image is then provided to the recognizer. The results (line "Frontalization (synthetic frontal views)") show that although frontalization achieves improved performance with respect to the static approach, it is clearly superseded by the proposed active approach.

TABLE II
ACTIVE FACE RECOGNITION STATISTICS (4 STEPS, HPID DATASET): STEPS PERFORMED BY THE ALGORITHM.

Image type	Angle	# Images	Percentage
I_r	0°	397	26.48%
I_r^+	+15°	368	24.54%
I_r^{++}	+30°	84	5.603%
I_r^{+++}	+45°	0	0%
I_r^{++++}	+60°	1	0.066%
I_r^-	-15°	515	34.35%
I_r^{--}	-30°	121	8.07%
I_r^{---}	-45°	9	0.600%
I_r^{----}	-60°	5	0.333%
Total	–	1500	100%

Statistics regarding the steps taken by the proposed approach were also evaluated and are presented in Table II for HPID dataset. These statistics show that in 26.48% of the cases the robot decided to stay in its initial position whereas in the remaining 73.64% it moved by $\pm 15^\circ, .., \pm 60^\circ$ (one to four steps). It shall be noted however that the decision on the ID of the depicted person is not necessarily obtained from the last position the robot has visited, due to the fact that the image with the maximum recognition confidence (FRC) is used for this purpose (equations (2) and (3)).

The average number of movements that the algorithm instructs the robot to perform can be easily evaluated from statistics such as the ones presented in Table II. Based on these calculations, the algorithm instructs the robot to make, on average, 0.76 (HPID) or 0.89 (QMUL) movements, a fact that signifies that the time required for active recognition (time for the computations as well as the time for the robot to move) is relatively low. Note that in case the robot decides to performs no movement (stay decision) the number of movements is obviously zero.

V. DISCUSSION AND CONCLUSIONS

An active approach for face recognition that utilizes facial views produced by facial image synthesis was presented in this paper. The robot that performs the recognition selects the best among a number of candidate physical movements around the person of interest by simulating their results through view synthesis. Experimental evaluation showed that the method supersedes both its static version and face recognition that involves frontalization through synthesis of frontal images.

It must be stressed that certain assumptions were adopted in this paper, whereas a number of issues were not fully addressed. First, the actual control of the robot so as to move in θ° increments around the person was not dealt with, since it falls outside the scope of the paper. However, a rough estimate of the person position with respect to the robot would suffice to enable robot control. Also, it was assumed that the person being recognized remains relatively static during the recognition process, which can be n acceptable assumption if the process is brief. If the person moves during this process, this shall be taken into account by the algorithm.

In addition, it was assumed that there are no obstacles in the robot's path. If this is not the case, these obstacles shall be detected (by e.g. depth sensors) and taken into account while planning the next movement. Furthermore, obstacles in the space between the robot and the person might occlude the person for certain robot positions. However, since the algorithm decides on the person's identity based on the acquired image where the recognizer obtained the largest recognition confidence, it is rather safe to assume that, in most such cases, the algorithm might not face serious problems, even if it has instructed the robot to move in positions where occlusions occur.

One could also consider, instead of using the synthesized views as proposed in this paper, to estimate the view angle of the robot camera with respect to the person and instruct it to move directly (namely, without intermediate steps) to the position that would allow it to capture a frontal view (0° in pan). However, there are certain difficulties that make this approach difficult in practice. Indeed, we observed in the experiments that view angle estimates (i.e., those provided by the utilized view synthesis algorithm used) although accurate enough for the purposes of view synthesis, are quite far from the ground truth values, thus rendering this approach problematic.

Future plans include evaluation of the algorithm in additional datasets and creation of a realistic simulation in an appropriate environment so as to investigate some of the issues mentioned above (occlusions, actual robot control, objects that hinder robot motion etc). Employing a more sophisticated face recognizer and comparing it to additional methods, are also planned.

REFERENCES

[1] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos, "Revisiting active perception," *Autonomous Robots*, vol. 42, no. 2, pp. 177–196, 2018.

[2] 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.

[3] J. Delmerico, S. Isler, R. Sabzevari, and D. Scaramuzza, "A comparison of volumetric information gain metrics for active 3d object reconstruction," *Autonomous Robots*, vol. 42, no. 2, pp. 197–208, 2018.

[4] 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)*. IEEE, 2016, pp. 3477–3484.

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

[6] 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*, vol. 32, no. 2, pp. 1–14, 2021.

[7] 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*, 2017, pp. 35–40.

[8] 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)*. IEEE, 2020, pp. 1–6.

[9] S. Chen, Y. Li, and N. M. Kwok, "Active vision in robotic systems: A survey of recent developments," *The International Journal of Robotics Research*, vol. 30, no. 11, pp. 1343–1377, 2011.

[10] B. Calli, W. Caarls, M. Wisse, and P. P. Jonker, "Active vision via extremum seeking for robots in unstructured environments: Applications in object recognition and manipulation," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 4, pp. 1810–1822, 2018.

[11] M. Malmir, K. Sikka, D. Forster, I. Fasel, J. R. Movellan, and G. W. Cottrell, "Deep active object recognition by joint label and action prediction," *Computer Vision and Image Understanding*, vol. 156, pp. 128–137, 2017.

[12] 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.

[13] 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*, 2017, pp. 2439–2448.

[14] 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)*. IEEE, 2020, pp. 748–752.

[15] 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*, vol. 10, no. 2, p. 601, 2020.

[16] 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)*. IEEE, 2017, pp. 604–611.

[17] 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*, 2020, pp. 5911–5920.

[18] J. Guo, X. Zhu, and Z. Lei, "3ddfa," https://github.com/cleardusk/3DDFA, 2018.

[19] J. Guo, X. Zhu, Y. Yang, F. Yang, Z. Lei, and S. Z. Li, "Towards fast, accurate and stable 3d dense face alignment," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[20] X. Zhu, X. Liu, Z. Lei, and S. Z. Li, "Face alignment in full pose range: A 3d total solution," *IEEE transactions on pattern analysis and machine intelligence*, 2017.

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

[22] Q. Wang, P. Zhang, H. Xiong, and J. Zhao, "Face. evolve: A high-performance face recognition library," *arXiv preprint arXiv:2107.08621*, 2021.

[23] N. Passalis, S. Pedrazzi, R. Babuska, W. Burgard, D. Dias, F. Ferro, M. Gabbouj, O. Green, A. Iosifidis, E. Kayacan, J. Kober, O. Michel, N. Nikolaidis, P. Nousi, R. Pieters, M. Tzelepi, A. Valada, and A. Tefas, "Opendr: An open toolkit for enabling high performance, low footprint deep learning for robotics," *arXiv preprint arXiv:2203.00403*, 2022.

[24] "OpenDR: A modular, open and non-proprietary toolkit for core robotic functionalities by harnessing deep learning, howpublished = https://github.com/opendr-eu/opendr, note = Accessed: 2022-06-27."

[25] 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.

[26] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4690–4699.

[27] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE signal processing letters*, vol. 23, no. 10, pp. 1499–1503, 2016.

[28] J. Deng, J. Guo, E. Ververas, I. Kotsia, and S. Zafeiriou, "Retinaface: Single-shot multi-level face localisation in the wild," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 5203–5212.

[29] G. B. H. E. Learned-Miller, "Labeled faces in the wild: Updates and new reporting procedures," University of Massachusetts, Amherst, Tech. Rep. UM-CS-2014-003, May 2014.

[30] 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*, vol. 6. FGnet (IST–2000–26434) Cambridge, UK, 2004, p. 7.

[31] J. Sherrah and S. Gong, "Fusion of perceptual cues for robust tracking of head pose and position," *Pattern Recognition*, vol. 34, no. 8, pp. 1565–1572, 2001.

8.11 Active Vision Control Policies for Face Recognition using Deep Reinforcement Learning

The appended paper follows.

Active Vision Control Policies for Face Recognition using Deep Reinforcement Learning

Pavlos Tosidis, Nikolaos Passalis, and Anastasios Tefas
Computational Intelligence and Deep Learning Group, AIIA Lab
School of Informatics, Faculty of Sciences
Aristotle University of Thessaloniki
Thessaloniki, Greece
{ptosidis, passalis, tefas}@csd.auth.gr

***Abstract*—Robotic systems are capable of interacting with their environment in order to better sense their surroundings. This key ability of robotic systems is often ignored when developing Deep Learning models, since the later are usually trained using static datasets. This limits the ability of robots to perceive the environment in challenging scenarios. On the other hand, integrating perception and action in tightly coupled systems while operating on-the-edge, holds the credentials for deploying DL-enabled robots in such scenarios; Thus leading to more robust agents that can solve challenging tasks more accurately. In this work, we investigate whether active perception approaches can be employed and integrated into robotic systems in order to improve face recognition accuracy, as well as, study the effect of such an approach on the computational requirements for edge applications. To this end, we propose a DRL-based control approach for training agents that are able to identify task-relevant objects, as well as, issue the appropriate control commands to acquire better results. Through the conducted experimental evaluation, we demonstrate that the proposed method leads to significant improvements in face recognition over the rest of the evaluated approaches by providing accurate control commands.**

***Index Terms*—Active Perception, Active Vision, Deep Reinforcement Learning**

I. Introduction

Recent advances in Deep Learning (DL) led to a number of spectacular applications, ranging from self-driving cars and robots that outperform humans in various tasks [1]. Despite the enormous success in these areas, DL methods operate in a static fashion, i.e., they do not typically provide means for interacting with the environment in order to better perceive it. This is in contrast, with the way many organisms, including humans, perceive their environment since perception and action usually form a tightly coupled system at various levels. For example, eyes can adjust to various illumination conditions while we tend to examine an object from different angles and/or distances when we are uncertain about it. This process is called *active perception* [2], and it is thought to be a critical component of robotic agents that can work in challenging real-world scenarios.

There have been several recent attempts to integrate active perception principles into DL models [3], [4]. Most of them focused on robotics tasks, where they attempt to appropriately manipulate a camera and/or a 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 [3], or use multi-view datasets to simulate some of the effects of active perception feedback [4]. 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 [4]. Another closely related line of work employs Deep Reinforcement Learning (DRL) algorithms to perform a specific control task [5], [6], [7], e.g., acquire a frontal view of a person [8]. 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 [9]. At the same time, the lengthy training time of DRL methods further limits their applications in robotics. As a result, despite their enormous potential for developing active perception approaches their 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 appropriate control commands accordingly to acquire better results. To train and evaluate the proposed method, we developed a simulation environment using the Webots simulator [10] and generated several 3D human models using the MakeHuman software [11]. 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 by issuing the appropriate control commands. 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.

The rest of the paper is structured as followed. First, the related work is briefly introduced in Section II. Then, the proposed method is introduced in Section III. The experimental evaluation is provided in Section IV, while conclusions are drawn in Section V.

II. Related Work

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 [12], to modern deep learning approaches that automatically learn the distinctive features of a face, when trained on massive datasets [13], [14], [15]. 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. This preprocessed image is then fed into a DL model which extracts a discriminative feature vector. Finally, this vector is compared to a set of feature vectors of people of interest [13], [14], [15], performing the final classification or verification task. The method proposed in this paper 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 Section IV, 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 an active vision pipeline.

This work is also closely related to active perception approaches. According to Bajscy [16], [17], an actively perceiving agent is one which can, among others, appropriately control its mechanical components in order to enable the best sensing of its 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 [3], [4]. 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 [3]. The controller module works as a rule-based controller that selects the most appropriate action according to the face recognition confidence and predefined thresholds for each action. A fully end-to-end trainable DL-based approach was also proposed in [4], 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. Moreover, this approach cannot fully exploit the potential of active perception, since it only considers 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, 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. A sample of the generated human models can be seen in Fig. 2. Furthermore, both 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.

III. Proposed method

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

$$\mathbf{x}_p = f_p(\mathbf{x}) \in \mathbb{R}^{W_p \times H_p \times C_p} \tag{1}$$

be the cropped face image, where the notation $f_p(\cdot)$ is used to refer to the face detector and preprocessing 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., [15], 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* [13], which is minimized when embeddings that belong to the same identity are close, while the representations of face images that do not belong to the same person are far. 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, during inference

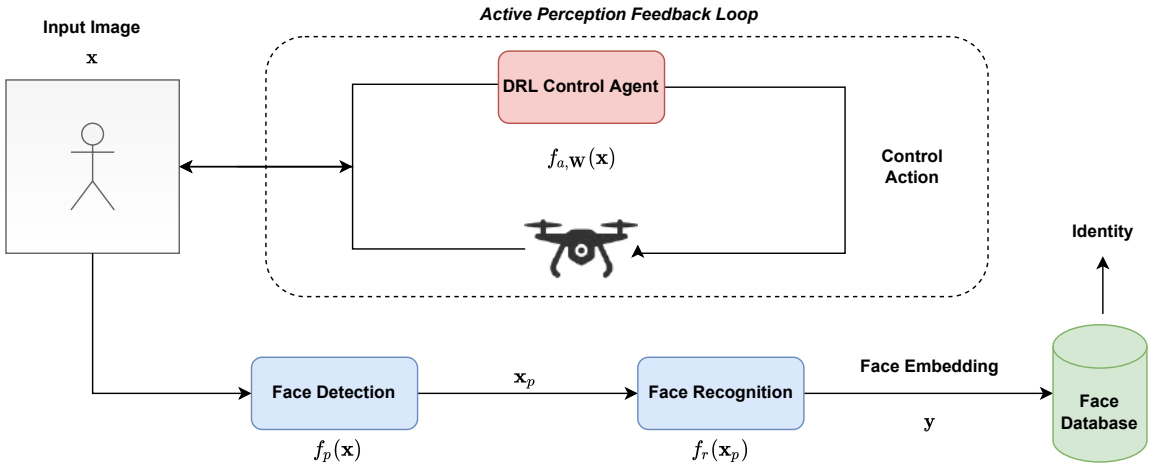


Fig. 1. Proposed Active Perception Approach: A DRL agent is employed to issue control commands in order to acquire the most appropriate view for improve face recognition accuracy.

Fig. 2. Realistic human models generated using MakeHuman.

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

$$k = \arg\min_i ||f(\mathbf{x}_i) - f(\mathbf{x})||_2 \ (\forall(\mathbf{x}_i, l_i) \in \mathcal{X}_d). \quad (2)$$

The proposed method aims to teach 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 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 computational 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., [4].

The proposed agent is trained using DRL. More specifically, the Proximal Policy Optimization (PPO) algorithm was used [6]. The reward used for training the DRL agent was defined based on the face recognition confidence of a pre-trained 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 (2), the reward at time-step t can be defined as:

$$r_t = \begin{cases} c & \text{if } \|\mathbf{y} - \mathbf{y}_k\|_2 < a \\ 0 & \text{otherwise} \end{cases}, \quad (3)$$

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:

$$c = 1 - \frac{\|\mathbf{y} - \mathbf{y}_l\|_2}{a}. \quad (4)$$

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, to 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 layer 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 constraint 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 faces at different directions in each episode. The simulation world consists of a square room, a human model at the center and a drone robot controlled by the DRL agent. 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 of 10,000,000 steps where performed during the training.

IV. Experimental evaluation

The proposed method was evaluated under two different setups. In the first setup, the agent was trained to select one of the first three actions ("stay", "move forward" and "move backward"). 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 other baselines. First, a face recognition pipeline was employed to evaluate the ability of existing approaches to detect and recognize humans at 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 [13] for the face recognition and RetinaFace [18] for the face detection. Furthermore, 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.

TABLE I
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	**0.68**
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.

The experimental results for the first setup are reported in Table I. In this setup, the drone was positioned at various distances in front of the human subject, ranging from 1m to 20m. Using a static setup, where the drone does not move, allows for recognizing persons only up to 4 meters. On the other hand, the rule-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 persons even up to 15m, while it can work correctly even for larger distances (up to 20m).

Similar conclusions can be also drawn for the evaluation results reported in Table II, 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. Additionally, 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 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.

TABLE II
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.

V. CONCLUSIONS

Despite its potential in a wide variety of robotics systems, active perception using DL models is a field not yet explored deeply. Indeed, it is expected that a robot should be able to interact with its environment to better understand it, improve situational awareness and make informed decisions. In this work we demonstrated that active perception approaches can indeed lead to improved face recognition accuracy in a wide variety of setups, including challenging ones, e.g., when images taken from the back side of humans or faces appear too small to be detected by traditional face detection models. At the same time, it was also shown that DRL can be efficiently integrated into such active perception pipelines, given that the appropriate reward function has been defined. Also, the proposed method can lead to performance improvements, apart from more accurate agents, since it can replace part of the existing DL pipelines. This work paves the way for more advanced DRL-based active perception approaches for human-centric perception.These approaches can be trained on more complex simulation environments, employ *sim2real* approaches [9], while also consider the trade off between the expected accuracy improvement and energy expenditure for each control action.

ACKNOWLEDGMENT

This work was supported by the European Union's Horizon 2020 Research and Innovation Program (OpenDR) under Grant 871449. This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

REFERENCES

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436, 2015.
[2] Ruzena Bajcsy, "Active perception," 1988.
[3] Masaki Nakada, Han Wang, and Demetri Terzopoulos, "Acfr: Active face recognition using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 35–40.
[4] Nikolaos Passalis and Anastasios Tefas, "Leveraging active perception for improving embedding-based deep face recognition," in *Proceedings of the International Workshop on Multimedia Signal Processing*, 2020, pp. 1–6.
[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller, "Playing atari with deep reinforcement learning," 2013.
[6] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov, "Proximal policy optimization algorithms," 2017.
[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.
[8] A Tzimas, Nikolaos Passalis, and Anastasios Tefas, "Leveraging deep reinforcement learning for active shooting under open-world setting," in *Proceedings of the IEEE International Conference on Multimedia and Expo*, 2020, pp. 1–6.
[9] Andrei A Rusu, Matej Večerík, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell, "Sim-to-real robot learning from pixels with progressive nets," in *Proceedings of the Conference on Robot Learning*, 2017, pp. 262–270.
[10] Olivier Michel, "Cyberbotics ltd. webots™: professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, pp. 5, 2004.
[11] Manuel Bastioni, Simone Re, and Shakti 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*, 2008, pp. 1–6.
[12] Jun Zhang, Yong Yan, and Martin Lades, "Face recognition: eigenface, elastic matching, and neural nets," *Proceedings of the IEEE*, vol. 85, no. 9, pp. 1423–1435, 1997.
[13] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4690–4699.
[14] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu, "Cosface: Large margin cosine loss for deep face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5265–5274.
[15] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 212–220.
[16] R. Bajcsy, "Active perception," *Proceedings of the IEEE*, vol. 76, no. 8, pp. 966–1005, 1988.
[17] Ruzena Bajcsy, Yiannis Aloimonos, and John K Tsotsos, "Revisiting active perception," *Autonomous Robots*, vol. 42, no. 2, pp. 177–196, 2018.
[18] Jiankang Deng, Jia Guo, Evangelos Ververas, Irene Kotsia, and Stefanos Zafeiriou, "Retinaface: Single-shot multi-level face localisation in the wild," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5203–5212.

8.12 Active Perception for Occlusion Removal in Face Recognition

The appended paper follows.

Active Perception for Occlusion Removal in Face Recognition

V. Dimaridou, N. Passalis and A. Tefas
Computational Intelligence and Deep Learning (CIDL) Group, AIIA Lab.,
Dept. of Informatics, Aristotle University of Thessaloniki

Abstract

Face recognition is an essential functionality in robotics, with a very high requested accuracy. While great success has been achieved in this task using all the latest advances of deep learning, it often relies on clear views of the query person. Partial or full occlusions in the face area greatly degrade the accuracy of available methods, setting them unsuitable for usage in robot agents. In this work we leverage active vision for getting a clear view of the person's face. For this, we carefully design a two-step pipeline, that initially predicts the target direction of the robot's movement. By moving towards to this direction using a predefined step, a second view of the face and the occluder object is captured. This view is later used for regressing the remaining movement the agent has to perform, as a ratio of the first step. The initial movement of the agent is essential for the network to take into consideration the distance of the occluder object. Using our framework we manage to clear out severe occlusions, above 50%, and reach verification accuracy of ***97%, 92% and 99% in agedb30, cplfw and lfw*** *benchmark datasets respectively.*

1. Introduction

The usage of facial recognition technology has increased dramatically the last couple of years, with new applications arising every day. Facial recognition is now so technologically mature, that it is being used in the most demanding fields, including border and access control, national security and banking identification.

In recent years face recognition is treated as a feature comparison problem. In the most common scenario, a detected and aligned image region of a person's face in given as an input to a deep convolutional network, which outputs a feature representation of it. Given the deep representation, a chosen similarity function is used to measure the degree of identity similarity between two images, known as *verification*, or a set of initially enrolled subjects, otherwise *identification*. Face recognition approaches follow the evolution of architecture and advanced training techniques of deep learning, thus common architectures such as ResNet, used in SphereFace [16] and VGG, proposed in VGGface2 [6] are utilized. Although backbone architectures significantly affect face recognition accuracy, the huge success rates are mostly based on carefully designed loss functions, specifically targeted to minimize the intra-class variation and maximize the inter-class variation. Large margin cosine [23] or additive angular margin losses [7] both target on obtaining highly discriminative face features. Face recognition poses as a zero-shot problem, and requires a massive amount of training data and training time to be deployed in real-world applications.

Person identification technology is also of great interest in the robotics field. The robot's ability to recognize individuals is a fundamental requirement for improving human-robot interaction. Face recognition techniques in robotics tend to follow the advances in literature, although they often use lightweight variations, due to limited computational ability. An example of hardware optimized deep learning face recognition module is presented in [22], where the authors optimize a topology based on SqueezeNet using advanced pruning and quantization, resulting on a well-fitted network that can be deployed on FPGA accelerator. Apart from hardware limitations, robotics face recognition also tries to tackle challenges introduced by the unconstrained environment. The authors of [12] propose a GAN schema that aims on predicting the neutral expression of the person in query, resulting in a face image that fits the distribution of the existing training face databases.

Face recognition challenges can be split into four main categories, the natural process of aging, which is uncontrollable in the sense that each person passes through different aging patterns; pose invariance, which can be potentially tackled with advanced alignment methods; severe illumination changes; and partial occlusions [1]. Further analyzing, partial occlusions refer to obstacles in the query image that block the face area by an occlusion less than 50%. The occluder object can be sunglasses, scarf, hair, masks, another person/object or even severe shadows. Partial FR is an active research topic that often uses face patches. More specifically, dynamic feature matching uses face patches

and face images of a subject for reducing the intra-class variation [9]. On the other hand, newer approaches modify the network architecture using attention modules [10] that drive the model to focus on relevant parts of the occludee face. All the aforementioned solutions try to tackle the partial FR problem using a single static image representation. However, a robot agent acts in dynamic environments and has the ability to further explore its surroundings to get a better understanding.

"We do not only see, we look", is written in an early active perception and exploratory robots work [3]. More recent works argue that one of the main reasons to use active control is to see a portion of the visual field otherwise hidden due to occlusion [4]. While object detection has undergone tremendous advancements, it is still rational that problems considered ill-posed for a passive observer can be simplified when managed by an active agent [2]. Applications on active perception that enforce deep learning often use reinforcement learning [[18], [20], [5]] and are being applied to various tasks, such as image classification or adversarial scenarios detection.

Enforcing the ability of robot agents to move around, we aim to tackle the occluded face recognition problem as an active perception vision task. We propose an active perception pipeline that intents on simultaneously moving towards a direction that gives a clear face view for the robot and performing face recognition. Our proposed pipeline is two-step, initially the robot captures the face image and decides the direction of the movement towards removing the object in the most efficient way. Later, given a new face view, the agent decides how much further it should move to fully clear out the occlusion, as a ratio of the initial movement. Both the direction decision network and the regression module are implemented using efficient deep learning networks.

One could easily question why two different networks are needed, when it is fairly easy to train a feature extraction network with two heads, one predicting the movement direction and the other the movement distance. This is a rational pipeline if the problem we aimed to solve would stay in the 2D image world or we had a 3D view of the environment as an extra input. However, we consider an agent that can only acquire color information of its surroundings, as a two dimensional image. Thus, given only an RGB image as input, there is no out-of-the-box way of knowing the actual distance between the agent and the object that causes the occlusion. We aim to solve this issue by first performing a pre-defined movement towards the direction that is picked by our network. This pre-defined movement will result in different image views based on three dimensional factors such as the robot's positioning with regards to the object and the human. Our second network is trained so that it recognizes the impact of the initial movement and regresses a factor showing how much more the network should move.

A second aspect of our work that needs to be noted is that our simulation modules operate in pixel values and not three-dimensional ones. We pick this formulation as we want to make full usage of the plethora of image datasets that exist and not restrict ourselves in three dimensional models and slow simulation modules. However, our pipeline is exactly constructed for this reason, the network's output is never expected to be in pixel or distance metrics. On the contrary, we force the network to predict the remaining movement as a *ratio of the first pre-defined movement*. We also conduct a detailed mathematical analysis on why the ratio that is predicted is analogous to distance metric values in the world. Our contributions can be summed up as follows:

- We consider a real-world problem, formally describe it, and construct an active perception pipeline to solve it.
- We design a two-step pipeline that initially predicts the direction of movement and then regresses towards the full object removal from robot's point of view. We manage to use the image data after the first predefine movement, in order to understand objects that are placed in various positions from the agent, without the need of any depth sensor.
- We follow a classic deep learning training procedure with two dimensional data, but specifically target into a pixel agnostic solution.
- We propose a pipeline which is currently applied in face recognition, but all the modules included can be plugged in a plethora of object identification/recognition problems.

The rest of the paper is structured as follows. The problem formulation along with formal definition of the main modules included in our pipeline are extensively described in Section 2. Our data creation and simulation pipeline are explained in Section 3, while our networks architecture and implementation details are reported in 2. All the required experiments that validate our methodology and assisted in picking the appropriate architecture for our final module are included in Section 5. Finally, Section 6 concludes the documents.

2. Methodology

2.1. Problem Formulation

Given a pair of face images and I_1 and I_2, we consider a function $Feat_m$ (implemented by a deep neural network) that predicts distinctive face features, F_1 and F_2 respectively. Given a threshold t_{th} and a distance function d, the

decision if the faces belong to the same or different individuals is produced by

$$Same = d(F_1, F_2) < t_{t_h}. \quad (1)$$

The verification accuracy calculated upon a number of fixed sets of verification pairs is acc. When obstacle images O_1 and O_2 are applied on arbitrary positions upon I_1 and I_2 respectively, new predicted features F_1^o and F_2^o are acquired. Repeating the application of eqn. (1) using the same functions $Feat_m$, d, a new acc^o is obtained, where generally, $acc^o < acc$.

Considering a robot agent that can move freely into 3D space, and capture new face images after each movement, we are looking for a set of movements that result in a viewpoint where the objects O_1 and O_2 are not visible in the new input images. The final target is to obtain $acc^o \simeq acc$.

2.2. Face Recognition Model

We consider a face recognition model that consists of a function $Feat_m$, responsible for predicting disciminative features and a face recognition head that compresses the features into a dimension where distance functions can predict the final classification match. Since a face recognition model should strictly separate faces in the feature space, specifically built classification heads and loss functions are enforced during the training procedure. During inference, the classification head is not used, on the contrary compressed face features are compared with candidate matches using various similarity functions.

2.3. Direction Decision Network

The first step towards getting a better viewpoint for the face image is to predict the direction towards obstacle removal. For this reason, we consider a direction classifier, D_m, which operates upon face embeddings F_n^o (input). D_m is implemented with a deep neural network, and is optimized with cross entropy loss,

$$cost(d, d'; \theta_1) \triangleq -\sum_{c=1}^{5} d'_{o,c} \log(D_m(F_n^o)_{o,c}), \quad (2)$$

where d is the predicted movement and d' is the ground truth, as defined by 2D occlusion generator. D_m network and its optimization procedure is depicted in Figure 1.

We argue and later show that the pre-trained face features predicted by $Feat_m$ preserve positional information regarding the obstacles that exist on face area. Direction Decision Network operates into 5 distinct categories: (1) up, (2) bottom, (3) right, (4) left, (5) none. None category is optimized using training samples that do not include obstacles in the face area, and is required so that the agent stays still and performs face recognition without further delay in this case. In our experimental analysis we operate on the before-mentioned 5 categories, however our pipeline can be easily extended to more complex agent movements.

2.4. Regress towards target

Given an occluded image I_n^o, *direction decision network* operates upon its features F_n^o and decides for the direction d that would efficiently remove the obstacle from the agent point of view. We employ a *3D emulator* module that synthesizes the occluded image from a viewpoint that follows movement direction d. The 3D emulator module is responsible for moving the occluder object in a way that simulates a constant robot movement towards direction d. The same module is in control of how the pre-defined agent movement would affect the new 2D image representation based on the distance of the agent and the occluder object, thus the name 3D emulator. Further information and formal definition about 3D emulator can be found in section 3.

Given the initial occluded image I_n^o and the new 2d representation captured by the modified viewpoint I'^o_n, we employ a function R_m that is responsible for predicting the remainder movement, so that the occlusion is fully removed. In detail, R_m is implemented by a deep neural network, which extracts features from I_n^o and I'^o_n, combines them, and regresses the rest of the movement. Our key contribution is that we do not regress either pixels or distance values. On the contrary, we predict a value that represents the proportion of the initial movement that needs to be repeated in order to fully clear out the robot's viewpoint from any occlusions. Finally, R_m network is optimized by

$$cost(r, r'; \theta_2) \triangleq \|R_m(I_n^o, I'^o_n) - r'\|_1, \quad (3)$$

where r' is the ground truth movement percentage produced by 3D simulator and r is the predicted movement, formulated by R_m network. R_m and its optimization procedure are illustrated in Figure 2.

2.5. Three dimensional analysis

Throughout the current document we defined all of our equations using two-dimensional representations in image plane. Although we consider pre-defined robot movement during stage 2 training and agent-object distance in 3D simulator module, all those are theoretically defined and are not represented by numerical values. 3DS module calculates the object movement and movement remainder (used as ground truth for stage 2 training), using image plane coordinates. Our end goal, as discussed, is a active perception pipeline that can be deployed in real life scenarios. That brings up a vital question: *How does the movement remainder, that is calculated using pixel values, correspond to real world distance metric?*

We consider a three-dimensional point $P_1(x, y, z)$ in world plane, pointing at the occluder object surface. We also employ a robot agent, that is capable of moving freely

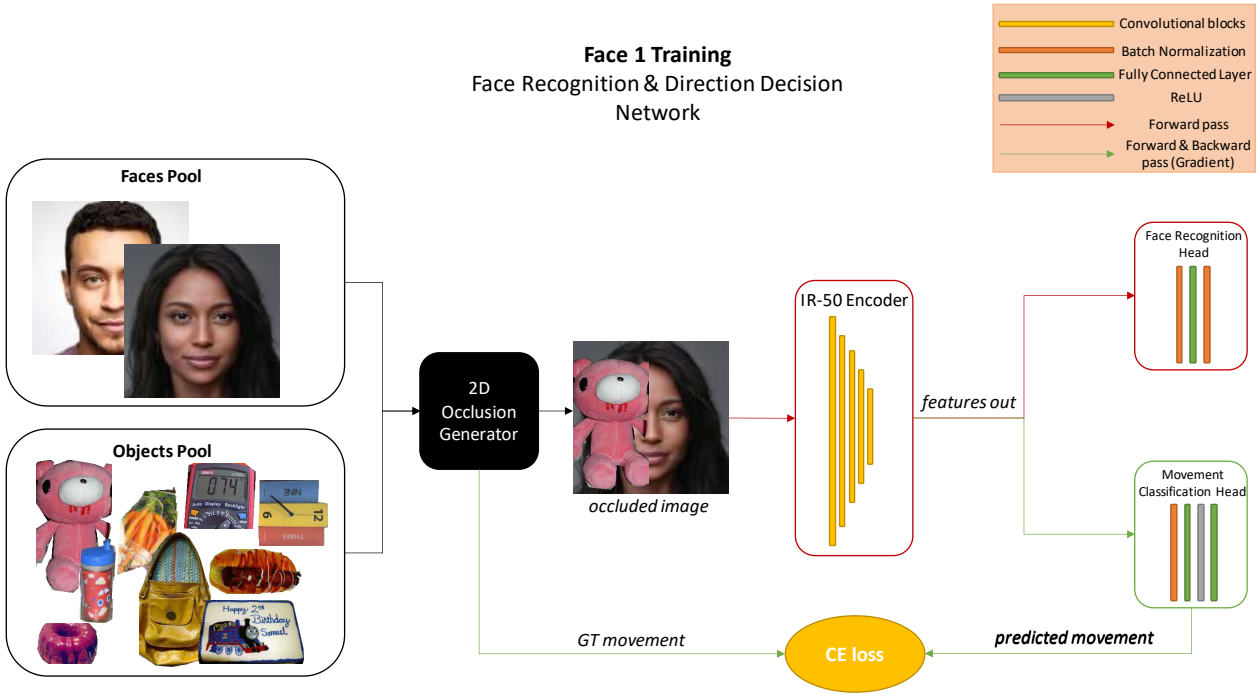


Figure 1. A detailed overview of our face 1 training procedure. *2D Occlusion Generator* module iterates through face (I_n) and object (o) pool, generating occluded face images I_n^o. Those are fed into *direction decision network*, D_m, which predicts the agent movement towards occlusion removal, d. Cross entropy loss drives the training procedure (eq. 2), with gradients flowing in the green arrows direction. Red outlined modules are kept frozen during training.

at the world plane. The agent, performs predefined steps in horizontal and vertical axis, denoted as m. Finally, we consider the robot agent to have a camera system attached, which can be described by a pinhole camera model.

Point $P_1(x, y, z)$ can be formatted as

$$p_1 = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} r_{11} & r_{12} & r13 & t_1 \\ r_{21} & r_{22} & r23 & t_2 \\ r_{31} & r_{32} & r33 & t_3 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{4}$$

in the image plane. We follow the pinhole camera model formulation, which converts a 3D point with cartesian coordinates (x, y, z) to camera space using $[R|t]$, where R is the rotation matrix and t is the translation vector that correspond points from 3D world to camera world. Final conversion to image plane comes from the first matrix product of 4, where f_x and f_y are the focal length values, and c_x and c_y is the camera center expressed in pixels. The same point $P_1(x, y, z)$ is calculated by

$$p_2 = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} r_{11} & r_{12} & r13 & t_1 + \mathbf{m} \\ r_{21} & r_{22} & r23 & t_2 \\ r_{31} & r_{32} & r33 & t_3 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{5}$$

using a translated $[R|(t, m)]$ matrix. The translation comes from the robot movement in the predicted direction d, by distance m. In eq. 5 we consider robot moving in horizontal direction, however our conclusions can be extended in vertical direction too. In our pipeline, R_m network tries to regress the remaining movement the robot has to perform, as a proportion of the first pre-defined step. The predefined step which is given in pixel coordinates by 3DS module does not yet correspond to real world robot movement. However, given the previous analysis, the remainder of the robot's movement can be extended as

$$r_{out} * Dp = r_{out} * (p_2 - p_1) = r_{out} * m * \begin{bmatrix} f_x \\ 0 \\ 0 \end{bmatrix}. \tag{6}$$

Eq. 6 shows that the predicted remaining movement can be expressed as proportion of the agent's actual movement

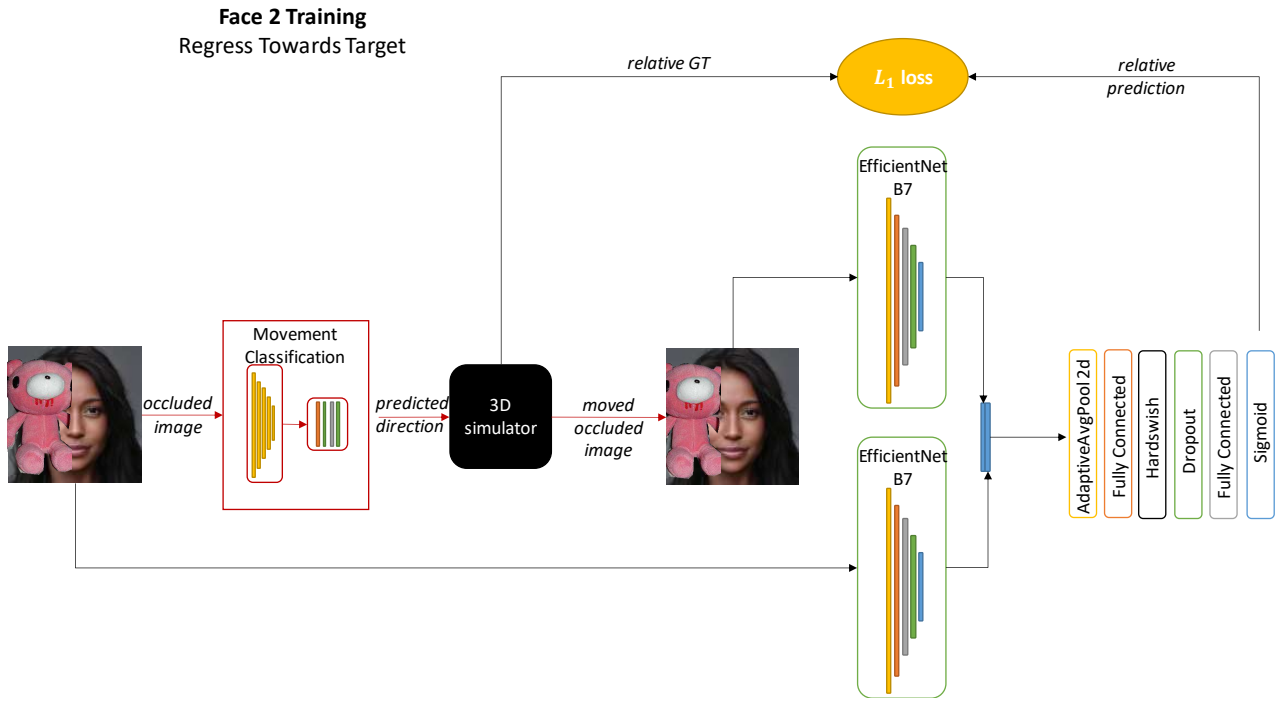


Figure 2. The second step of our training pipeline. Initial occluded image I_n^o is used as input in D_m network, and the predicted direction d is fed into *3D simulator* module. A new viewpoint of occluded image, referred as ${I'_n}^o$, along with the initial I_n^o view are fed into regression prediction network R_m. The training is led by l_1 loss, between R_m output and ground truth relative movement produced by *3D simulator*. During this step, D_m network isn't optimized.

and focal length, which is constant for a given camera system. The above mathematical analysis is illustrated in Figure 3.

3. Occluded Faces Dataset

In order to initially test our assumption that face recognition is severely affected by partial occlusions and later construct an active perception method that solves the above problem, we need to create a pipeline that can generate partially occluded face images. Our main requirement for the data creation pipeline is that it can be easily parameterized, in order to conduct different experiment types. Towards this end, we consider a set of 2D RGB face images, $\mathcal{F}$, and a set of 2D RGBA object images, $\mathcal{O}$. Overlaying set $\mathcal{O}$ to $\mathcal{F}$ set, results to a huge amount of occlusion data, that is easy to handle and fast to generate. We first describe the baseline datasets used as $\mathcal{F}$ and $\mathcal{O}$ sets, and then describe the preprocessing steps along with our 2D Occlusion Generation and 3D Simulation modules.

3.1. Baseline Datasets

MS-Celeb-1M MS-Celeb dataset [8] is one of the largest publicly available databases used for face recognition and verification. It consists of a total of 10M images, from 1M individuals. The individuals depicted in this database are people that have received public attention, mostly due to their profession. The dataset includes diversity both in terms of age and race. The subset that is used in the current pipeline, as the main training dataset $\mathcal{F}$, is constructed by selecting the top 100K celebrities, based on the frequency of their appearance. The final training dataset consists of 5084127 unique images.

COCO Common Objects in Context [15] dataset is a publicly available database that consists of 328K images. It includes highly detailed annotations for object detection, human pose estimation and instance segmentation. COCO dataset comes with 2.5M labelled instances, depicting 91 categories of commonly used objects (e.g. toothbrush, oven, apple). The dataset is split into training, validation and testing subsets. Training set consists of 164K images, while validation and testing set share 82K images each. In the current document, objects acquired from training set are used in the training process and objects from validation set during architecture search. COCO dataset consists of objects with different scaling, type and viewpoint, thus using it as the overlay dataset $\mathcal{O}$ offers high diversity to the data generation pipeline.

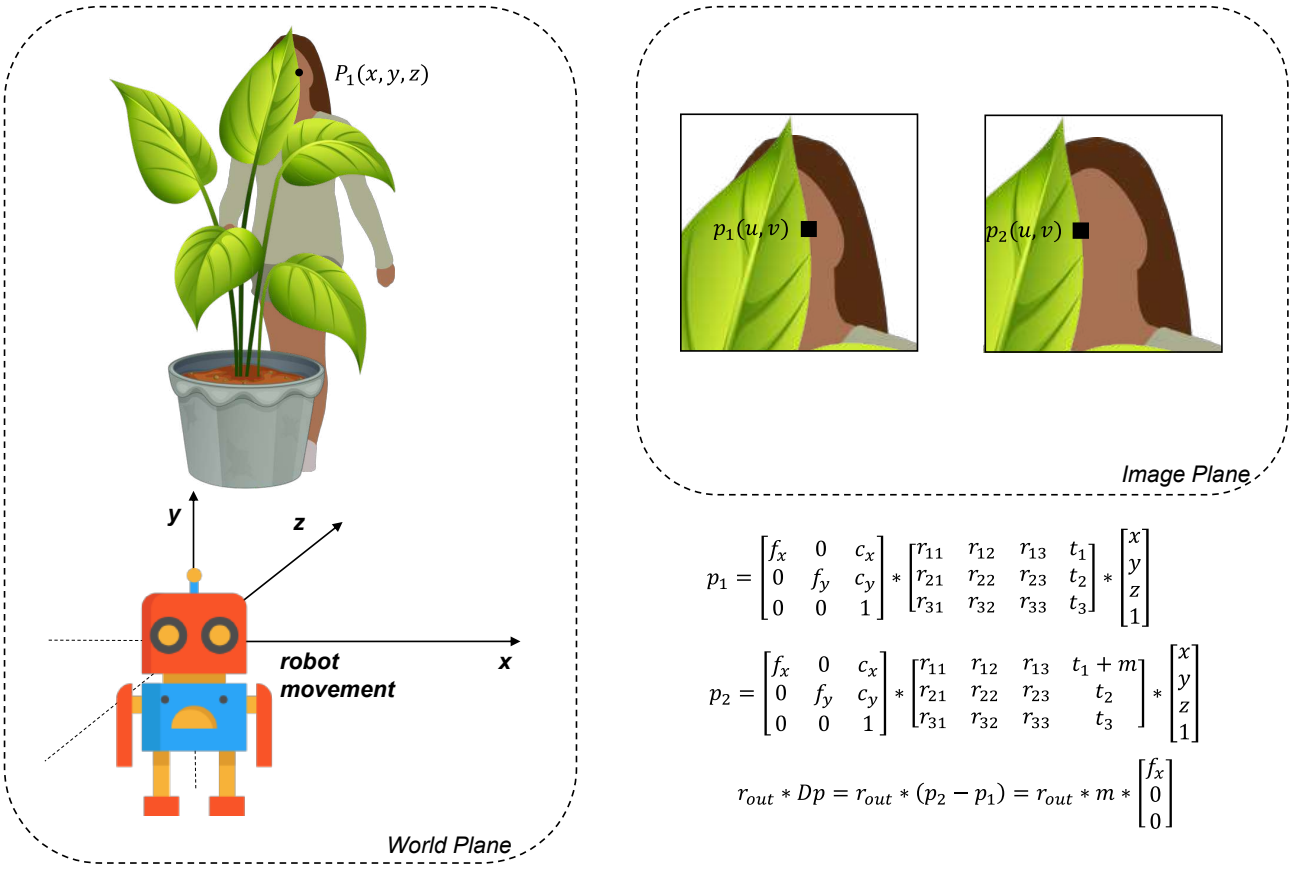


Figure 3. We consider a robot agent that can move freely into world plane and a partly occluded person (left side of figure). A point $P_1(x, y, z)$ of the world plane is expressed as $p_1(u, v)$ on image plane (right side of figure). When the robot agent moves towards the predicted direction expressed by D_m, a new edge occlude point is created, denoted as $p_2(u, v)$. R_m model's training target is created by predicting a *ratio of occlude change in pixel values*, which can be expressed as a *ratio of the agent's movement*.

3.2. Occlusion Generator & 3D Simulator

COCO dataset includes highly accurate annotations for instance segmentation task, which makes it ideal for object extraction. More specifically, we use the instance segmentation masks to accurately crop all the annotated objects included in each sample image. We save the cropped objects as RGBA samples, in order to seamlessly apply the occlusions on face images. We later apply 3 post processing steps to further improve the quality and realism of the occlusions: (1) we remove images that their axis aligned bounding box is smaller than 50x50 pixels; (2) we apply contour detection and exclude the images that contain more than one blob (this happens due to object split, caused by occlusion with a different instance); (3) we further crop the objects, so that the object coverage in the RGBA image is higher than 70%. We specifically perform the last two steps, because we want the object to occlude as much as possible and not leave face features visible in the occlusion areas. Our final object dataset includes 129K and 65K objects split into training and validation sets respectively.

Our face occlusion dataset is formulated by two separate components: the **2D Occlusion Generator**(2DOG) and the **3D simulator**(3DS) module. **2D Occlusion Generator** picks an image and an object from sets $\mathcal{F}$ and $\mathcal{O}$, a direction of occlusion and an occlusion percentage. Given a face image of size $w_f \times h_f$, an object with size $w_o \times f_o$ and an occlusion percentage O_p we resize the object to $w_f \times (O_p * h_f)$ size, apply the appropriate padding and overlay the two images. For training, we choose to place the model up, down, right and left (object rotation is performed for right and left occlusions). Other experimentation during testing phase includes objects placed on the center of the image or in random positions across the image plane. **3D simulator** module accepts an occluded image produced by 3DOG and the direction decided by the direction decision network. It first simulates different object distance by setting the movement impact to be variant across samples. Considering a movement towards the direc-

tion of the classifier's decision, the object padding changes and it is reapplied on top of the initial face image. The same module then calculates the remaining movement needed for complete object removal, as a ratio of the movement impact. This ratio is the target for the regression model optimization process. Implementation-wise, our data generation pipeline applies on-the-fly modifications to the training/validation samples, while a random seed on both 2DOG and 3DS modules is used for result reproducibility.

4. Architecture

The pipeline described in Section 2 includes three networks: (1) Face Recognition Model; (2) Direction Decision Network; (3) Ratio Regression Module. In this section we describe the architecture of each one, as well as the proposed two-step training procedure.

4.1. Models Architecture

Face Recognition Model, $Feat_m$, is implemented using a deep convolutional network provided by the high performance face recognition library Face.evolve [24]. The backbone used is a ResNet inspired network with 50 layers; the head used during training is ArcFace [7]; and focal loss [14] drives the training procedure. We select the pretrained variant that is optimized on MS-Celeb-1M dataset. Throughout all the training procedure for our D_m and R_m models we do not back-propagate gradients on $Feat_m$ model, thus keeping its weights frozen. We choose the above model since we want a well fitted and accurate baseline to conduct our experiments, however our proposed modules can be plugged in to every face recognition network that produces face features.

Direction Decision Network, D_m, is designed as an extra classification head that predicts direction data parallel to $Feat_m$ final classification head. It operates on the $512 \times 7 \times 7$ feature maps produced by face recognition model, using two linear layers, where the last linear module outputs five probabilities, one for each possible direction. More details on D_m architecture can be found on Figure 1, in the Movement Classification Head component. While it would be possible to enforce a separate backbone and head to serve as direction decision network, we choose to operate upon pre-trained face features. The main reason for this choice is that we can acquire both the face representation and the proposed direction of movement on a single forward pass. Given the above, the recognition process can continue with no further delay in the case that the predicted direction in none (which in real scenarios will be a very common one).

Ratio Regression Module, R_m, expects an input of two face images. A logical continuation of D_m design process would be to use $Feat_m$ predicted features for each image, fuse them, and attach them into a regression head. While this method often results to accurate predictions, it has a severe drawback: the features predicted by $Feat_m$ are robust to small object displacements. This results to feature maps with zero mean absolute difference. To overcome this issue, we use a separate lightweight backbone network, which accepts the raw images as input and outputs the regression value. We consider two different variations of our regression module: a backbone model that accepts six channel input and predicts one class; and a Siamese inspired network architecture, where the same backbone branch predicts features that are concatenated and used as input in a two linear layer regressor. The regressor architecture is depicted on 2. The backbone used is EfficientNet B7 variant [21] variant. Both of the variations use Sigmoid activation, so that we can limit the robot's movement into non-extreme values.

4.2. Implementation Details

We enforce a two-step pipeline to train our direction decision and ratio regression modules. The first part of the training pipeline includes the fine-tuning of the movement classification head, as visualized in Figure 1. We load pretrained weights both for the face feature extractor and face recognition head (from [24]) and keep them frozen in the optimization step. The second part of the training pipeline includes the training of ratio regression module, as depicted in Figure 2. The previously trained (from face 1) movement classification head remains frozen in this step, and EfficientNet B7 backbone along with its classification head are being optimized. We initialize the parameters of the backbone with ImageNet pre-trained weights and normalize the input images accordingly. We use PyTorch deep learning library [19] and enforce CE and MSE for stage 1 and 2 respectively. All the experiments mentioned in this manuscript are trained for 250K iterations, and the latest model is used. Adam optimizer is used [13] with an initial learning rate of $1e{-}3$.

5. Experiments

5.1. Validation Datasets

LFW Labeled Faces in the Wild [11] is a publicly available dataset used in face verification. The dataset contains 13233 labeled images, collected from the web. The total number of distinct people depicted in this collection is 5749. From those, 1680 people have two or more distinct images picturing them. The dataset is separated into 10 non-repeating subsets of verification pairs, where each subset contains 300 positive and 300 negative matches. LFW dataset is mostly used in academic research, since it lacks of diversity in terms of depicting people from different age groups or various ethnicity. Some examples of LFW images are depicted in the first row of Figure 4.

CPLFW Cross-Pose Labeled Faces in the Wild [25] is

Dataset	agedb-30	cplfw	lfw
Verification Accuracy	97.63%	92.23%	99.81%

Table 1. Baseline accuracy of [24] across different datasets.

a refactored version of LFW dataset. It contains 11652 images, with the same individuals as LFW, however it is enriched with additional images for each person, so that none of them has less than two images. The additional images and the verification pairs are closely selected, so that positive samples include images with as large pose difference as possible. Additionally, negative samples include cherry-picked images with people with the same race and gender. Second row of Figure 4 illustrates three hard-examples of same person pair (first six images) and different person pair (last six images).

AgeDB-30 Age Database [17] is a database that contains a large number of human faces, annotated with their ages, at the time the photo was taken. It includes 16488 unique images, with annotations of identity, age and gender. The number of distinct subjects is 568, with the average number of photos per person being 29. Since it was first constructed to be used in age-specific tasks, it includes subjects with age ranges from 1 to 101. AgeDB-30 is an use-case of AgeDB, where a verification dataset is created, exploiting the identity labels. Following LFW protocol, AgeDB dataset consists of 10 folds of images, where each fold includes 300 positive and 300 negative pairs. The variation of AgeDB used in the current document, is the one where the age difference of each verification pair is set to a fixed range of 30 years. The last row of 4 shows some sample images from AgeDB-30 dataset.

The validation protocol followed in the current document uses all the three available datasets to verify the generalization ability of the trained models. LFW poses as the easiest target, while CPLFW and AgeDB-30 both include different additional challenges. It is important to note here that the maximum expected accuracy is limited by the frozen face recognition network used in all of the experiments.

5.2. Partial Occlusion Effect on Verification

We consider the face features predicted by [24] as our baseline face recognition accuracy. We choose a stable and well-fitted deep learning module to serve as our baseline model, however it has to be noted that our active perception modules can be plugged into any model that produces face features. Initial face verification accuracy across different datasets is given in Table 1.

Our target is to inspect how verification accuracy drops when random objects are placed upon face images, using our dataset generation pipeline. For this reason, we maintain the feature extraction network unchanged, while the verification pairs also remain the same. We gradually increase the occlusion percentage and place the objects on our four pre-selected locations: top, borrom, right & left. The verification accuracy gradually drops, as depicted in Table 2.

Dataset	agedb-30	cplfw	lfw
Baseline Accuracy	97.63%	92.23%	99.81%
30% object coverage	96.46%	90.28%	99.49%
40% object coverage	92.03%	85.64%	97.68%
50% object coverage	81.50%	72.46%	89.75%
60% object coverage	69.76%	63.50%	81.68%

Table 2. Face verification accuracy when occlusion objects are applied on face images. Object coverage percentage shows the occlusion object coverage upon the face image. The task is verification, thus random classifier percentage is 50% (binary task).

Dataset	agedb-30	cplfw	lfw
DDN Classification Accuracy	89.46%	84.13%	90.775%

Table 3. Classification Accuracy of DDN. Our validation split of COCO objects is applied on verification datasets, with occlusions ranging from 20% to 60%. The target used to compute the classification result is produced by 2D Occlusion Generator module.

When applying a 30% object occlusion, our selected baseline model manages to extract discriminatory features (in the majority of the cases), proving its robustness on mild occlusions. However, as the occlusion coverage rises, the model cannot keep-up, and gradually drops its accuracy to near random classifier levels. Please note that the task we consider here is verification, which is a binary task, therefore the random classifier percentage is 50%. Another observation worth noticing is that accuracy drop reflects on dataset difficulty level. LFW verification accuracy doesn't manage to drop bellow 80%, proving that it is a relatively easy dataset to solve, while cplfw suffered the most severe accuracy drop. Finally, for the majority of our experimentation we will not increase the object coverage above 60%, as we are interested in partial occlusion cases.

5.3. Iterative Direction Decision Network

The first step of our pipeline is to train the decision classifier. Given the architecture and implementation details mentioned in subsection 4, we present the classification accuracy of our Direction Decision Network in Table 3. In order to validate our network's accuracy, we used our 2D Occlusion Generator module, where the faces pool consists of the images present in LFW, CPLFW and AgeDB-30 verification pairs and the objects pool is a 20% subset of our COCO objects, all unknown during training. 2D Occlusion Generator applies occlusions in range 20%-60%.

A rational first approach to our obstacle removal problem would be to iteratively use the direction decision network. To this end, a pre-defined step that the agent will perform is set, and the input image is fed into D_m network. The agent will perform the movement as defined by the step and the

Figure 4. Examples of LFW (row 1), CPLFW (row 2) and AgeDB-30 (row 3). First three pairs of each row illustrate actual verification samples of the same person, while last three pairs depict verification samples of different people. Notice how CPLFW and AgeDB-30 include more challenging pairs compared to LFW.

network's decision. Newly captured images are again fed to the network, until the network decides that there is no obstacle in the face view, or a maximum step has reached. We apply the last condition to verify that the procedure will end in the case the network makes circle decisions. We use the iterative pipeline to report verification accuracy on multiple datasets on Table 4. Note that we report multiple occlusion positions hard cases, such as constant high occlusion or even center position on the image. Even though the baseline verification accuracy in severe occlusion cases drops down to random classifier, the iterative solution manages to reach high verification scores after 10 position fixes, proving the generalization ability of our D_m module, that mostly makes usage of pre-trained face features.

5.4. Direction & Regression Pipeline

Although our iterative solution boosts the verification accuracy near baseline face verification metrics, it lacks in terms of efficiency. More specifically, it is unsure how many iterations should be set as a maximum value. For example center occlusion position in Table 4 could not be removed even after 10 iterations of the direction network. For this reason, we employ our regression model, that given two different views of the face image, regresses the percentage of movement the network should perform for the occlusion to be fully removed. In this section we aim to discuss the design choices made for the final regression model.

As discussed, a natural architecture choice would be to take advantage of the pre-trained face features produced by $Feat_m$, fuse the predicted featured coming from the two sequential images, and attach a regression head for the final regression output. This has been implemented and tested in our pipeline, with feature fusion being implemented using concatenation and regression head defined as a two linear layer network that accepts flattened features. The regression l_1 loss perceived by the network, when validated on unknown data, is given in the second line of Table 5. To overcome the issues caused by the Features Regressor when in the two sequential the object has slight movement, we design EfficientNet Dual Regressor, which instead of using the pre-trained $Feat_m$ model, uses the EfficientNet backbone to produce features for each of the two images, fuses them, and uses the same regreesion head as in Features Regressor case. The results acquired by this training procedure are given on the second row of Table 5. We have to mention here, that for fair comparison, we train the two before-mentioned networks for the same number of iterations. However, this is towards Feature Regressor favor, since it has a lot more parameters to fine-tune. Lastly, we design a simpler network, that includes an Effiecient Regressor backbone, that accepts 6 channels on its input layer and uses the same regression head to output the regression value. The l_1 loss of this model (see third row of Table 5) shows that it converges quickly, even for a small number of iterations. Thus, this model architecture is included in our final pipeline.

In our stage-2 training, we use our D_m model in order to create the second occlusion image. This exact pipeline scheme, including both D_m and R_m modules is also being using during inference. However, we experiment with replacing the trained movement classifier with a random classifier, which randomly picks a direction from choices $top, bottom, right, left$. The intuition behind this choice is that the resulting model would be more robust and could handle wrong classifications from D_m model during inference. Our design choice is validated by cplfw column in Table 5, where our model trained with random occlusion model (denoted with †), improves upon baseline, in the dataset where D_m model has the poorest accuracy.

Occlusion Percentage	20%-60%			60%			60% center		
Dataset	agedb-30	cplfw	lfw	agedb-30	cplfw	lfw	agedb-30	cplfw	lfw
Baseline Verification Accuracy ↑	89.03%	81.61%	95.04%	69.76%	63.5%	81.68%	52.43%	51.56%	63.65
Verification Accuracy Iterative Fix (max 1)↑	94.51%	87.43%	98.23%	83.01%	71.88%	91.36%	55.28%	54.23%	72.28%
Verification Accuracy Iterative Fix (max 2)↑	96.50%	89.93%	99.41%	91.7%	79.95%	96.96%	65.88%	61.23%	82.06%
Verification Accuracy Iterative Fix (max 3) ↑	97.23%	90.91%	99.59%	95.73%	85.35%	99.18%	77.81%	71.91%	87.98%
Verification Accuracy Iterative Fix (max 10) ↑	97.38%	91.68%	99.66%	97.28%	90.83%	99.56%	88.85%	86.74%	91.46%

Table 4. Verification accuracy after iterative usage of direction decision network. Baseline Verification Accuracy denotes the accuracy when no correction is performed in the occluded image. Note that our iterative solution manages to boost the accuracy even on extreme occlusions - however, this comes on a high computational cost.

Dataset	agedb-30	cplfw	lfw
Features Regressor $l_1 \downarrow$	8.93	15.78	8.46
EfficientNet Dual Regressor $l_1 \downarrow$	6.83	34.54	11.56
Efficient 6 channel Regressor $l_1 \downarrow$	**3.89**	8.01	2.70
Efficient 6 channel Regressor†$l_1 \downarrow$	5.50	**6.00**	**2.17**

Table 5. Regression accuracy of different models, measured on our three validations sets, using unknown occlude objects (from validation set). When trained on the same number of iterations, the 6 channel solution clearly outperforms the other two.

Dataset	agedb-30	cplfw	lfw
Verification Accuracy 20-60% ↑	97.63 %	92.05%	99.8%
Verification Accuracy 60% ↑	97.60	91.79	99.76
Verification Accuracy 60% center ↑	90.66	88.96 %	93.34 %

Table 6. Verification accuracy of Efficient 6 channel Regressor † model, measured on our three validations sets, using unknown occlude objects (from validation set). When validated on the same occlusion schema as in training, the verification accuracy reaches baseline face model levels. Even when validated on severe occlusions, our pipeline manages to restore a large percentage of verification accuracy loss.

6. Conclusion

Object recognition tasks are often tackled using a static inference schema. However, robot agents act in a dynamic environment and are able to alter the viewpoint of their camera system. For this reason, we consider a common problem, namely the face verification under partial occlusion, and aim to tackle it using an active perception pipeline. Our proposed two-step pipeline can be trained using classic deep learning procedures, while being able to advantage from all the available image data present in face recognition literature. However, we systematically prove that our pipeline is specifically created to be pixel-agnostic in environments that only include image color sensors. Our experimental results prove the decrease of face verification accuracy into random classifier values, under partial occlusion settings. We later manage to correct the accuracy drop near percentage rates before the addition of occluder objects using our proposed methodology. We hope that our pipeline leads the way for more active perception methodologies that are the only way to solve ill-posed problems in real world.

Acknowledgment

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

References

[1] Face recognition techniques: A survey. *CoRR*, abs/1803.07288, 2018. Withdrawn. 1

[2] Alexander Andreopoulos and John K Tsotsos. 50 years of object recognition: Directions forward. *Computer vision and image understanding*, 117(8):827–891, 2013. 2

[3] Ruzena Bajcsy. Active perception and exploratory robotics. 1989. 2

[4] Ruzena Bajcsy, Yiannis Aloimonos, and John K Tsotsos. Revisiting active perception. *Autonomous Robots*, 42(2):177–196, 2018. 2

[5] Luca Bartolomei, Lucas Teixeira, and Margarita Chli. Semantic-aware active perception for uavs using deep reinforcement learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3101–3108. IEEE, 2021. 2

[6] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*, pages 67–74. IEEE, 2018. 1

[7] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1, 7

[8] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *European conference on computer vision*, pages 87–102. Springer, 2016. 5

[9] Lingxiao He, Haiqing Li, Qi Zhang, and Zhenan Sun. Dynamic feature learning for partial face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7054–7063, 2018. 2

[10] Stefan Hörmann, Zeyuan Zhang, Martin Knoche, Torben Teepe, and Gerhard Rigoll. Attention-based partial face recognition. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 2978–2982. IEEE, 2021. 2

[11] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik 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. 7

[12] Tianfu Jiang, Tao Wang, Boyan Ding, and Han Wu. Degan: De-expression generative adversarial network for expression-invariant face recognition by robot vision. In *2019 WRC Symposium on Advanced Robotics and Automation (WRC SARA)*, pages 209–214. IEEE, 2019. 1

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 7

[14] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 7

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 5

[16] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 212–220, 2017. 1

[17] Stylianos Moschoglou, Athanasios Papaioannou, Christos Sagonas, Jiankang Deng, Irene Kotsia, and Stefanos Zafeiriou. Agedb: the first manually collected, in-the-wild age database. In *proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 51–59, 2017. 8

[18] Hossein K Mousavi, Guangyi Liu, Weihang Yuan, Martin Takáč, Héctor Muñoz-Avila, and Nader Motee. A layered architecture for active perception: Image classification using deep reinforcement learning. *arXiv preprint arXiv:1909.09705*, 2019. 2

[19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 7

[20] Macheng Shen and Jonathan P How. Active perception in adversarial scenarios using maximum entropy deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3384–3390. IEEE, 2019. 2

[21] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 7

[22] Iris Walter, Jonas Ney, Tim Hotfilter, Vladimir Rybalkin, Julian Hoefer, Norbert Wehn, and Jürgen Becker. Embedded face recognition for personalized services in the assistive robotics. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 339–350. Springer, 2021. 1

[23] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1

[24] Qingzhong Wang, Pengfei Zhang, Haoyi Xiong, and Jian Zhao. Face. evolve: A high-performance face recognition library. *arXiv preprint arXiv:2107.08621*, 2021. 7, 8

[25] Tianyue Zheng and Weihong Deng. Cross-pose lfw: A database for studying cross-pose face recognition in unconstrained environments. *Beijing University of Posts and Telecommunications, Tech. Rep*, 5:7, 2018. 7

8.13 AUTH-Persons: A Dataset for Detecting Humans in Crowds from Aerial Views

The appended paper follows.

AUTH-PERSONS: A DATASET FOR DETECTING HUMANS IN CROWDS FROM AERIAL VIEWS

Charalampos Symeonidis, Ioannis Mademlis, Ioannis Pitas and Nikos Nikolaidis

Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece
Email: {charsyme, imademlis, pitas, nnik}@csd.auth.gr

ABSTRACT

Recent advances in artificial intelligence, control and sensing technologies have facilitated the development of autonomous Unmanned Aerial Vehicles (UAVs). Detecting humans from video input captured on-the-fly from UAVs is a critical task for ensuring flight safety, mostly handled with lightweight Deep Neural Networks (DNNs). However the detection of individual people in the case of dense crowds and/or distribution shifts (i.e., significant visual differences between the training and the test sets) is still very challenging. This paper presents AUTH-Persons, a new, annotated, publicly available video dataset, that consists of both real and synthetic footage, suitable for training and evaluating aerial-view person detection algorithms. The synthetic data were collected from 8 visually distinct photorealistic outdoor environments and they mostly contain scenes with crowded areas, where heavy occlusions and high person densities pose challenges to common detectors. This dataset is employed to evaluate the generalization performance of various state-of-the-art detection frameworks, by testing them on environments that are visually distinct from those they have been trained on. Finally, given that Non-Maximum Suppression (NMS) methods at the end of person detection pipelines typically suffer in crowded scenes, the performance of various NMS algorithms is also compared in AUTH-Persons.

Index Terms— person detection, Unmanned Aerial Vehicles, synthetic data generation, Non-Maximum Suppression

1. INTRODUCTION

Recent advances have led to an unprecedented popularization of Unmanned Aerial Vehicles (UAVs, or "drones") during the last decade. Drones have proven useful for many civilian and military applications, such as search and rescue operations, surveillance, inspection, mapping [1], wildlife monitoring, crowd monitoring/management, precision agriculture, or aerial media production [2] [3] [4] [5]. Gradual increases in UAV cognitive autonomy have made flight safety a critical issue, due to the hazard drones potentially pose in case of malfunction. Safety during UAV interactions with the environment must be particularly ensured when the vehicle is operating near humans. Autonomous UAVs should be able to visually detect people with a high-level of precision from various aerial views [6]. This task poses challenges due to the small size of objects/persons (especially in high flight altitudes), as well as due to unforeseen and wide-ranging variations in illumination, camera orientation, etc. Problems are exacerbated when UAVs must detect the presence of individuals within crowded areas.

A typical object detector, employed in real-life conditions, must be trained on multiple datasets in order to improve its generalization abilities and ensure its robustness during actual deployment. In recent years, several real-world and synthetic datasets have been proposed to tackle the problem of detecting humans from aerial images/video frames. *Stanford Drone Dataset* [7] is a large-scale video dataset consisting of 60 annotated videos for detection and tracking of various object classes, including pedestrians. The *Okutama-Action* [8] dataset, mainly developed for concurrent human action detection, can also be employed for person detection. It consists of 43 minute-long fully-annotated sequences with the corresponding bounding-boxes/Regions-of-Interest (ROIs) and 12 action classes. The *VisDrone dataset* [9] consists of 263 video clips with 179,264 frames and additional 10,209 static images. The videos/images were acquired by various drone platforms, including the DJI Mavic Phantom series, and depict various scenarios across 14 cities in China. The dataset is suitable for image and video object detection, as well as for single/multi-object detection tracking.

In the general person detection task, the use of synthetic datasets has recently gathered pace since they can viably replace or augment real-world training data. In [10], 3D human models rendered on random backgrounds are employed to train a pedestrian detector. In a similar fashion, [11] inserts realistic DNN-generated 3D human models into existing natural background images, while trying to select appropriate scale and insertion locations. In [12], a graphical simulator is proposed which can automatically generate datasets for pedestrian and crowd analysis. Experimental evaluation

The AUTH-Persons dataset is available at: `https://aiia.csd.auth.gr/open-multidrone-datasets/`. This research has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements No 731667 (MULTIDRONE) and No 871449 (OpenDR).

on crowd estimation showed that DNN models which were pretrained on a synthetic dataset and later finetuned with the real-world dataset, outperformed models trained exclusively with real-world data.

A typical case where person detection methods may drastically fail to perform is when they operate on images depicting dense crowds [13]. Non-Maximum Suppression (NMS), which is a common post-processing step typically placed at the end of the overall person detection pipeline, suffers especially in crowded scenes. NMS methods prune the number of overlapping detected raw candidate ROIs generated by a detector, in order to assign a single and spatially accurate detection to each object. The de facto standard in NMS for object detection is Greedy-NMS [14]. 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. Modern alternatives include Soft-NMS [15], where a rescoring function decreases the score of neighboring less confident detections, instead of completely eliminating them, achieving better precision and recall rates. GossipNet [16] 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. GossipNet was modified in [17] for the specific case of person detection from aerial views, so as to jointly process visual appearance and geometric properties of candidate ROIs. More recently, [18] proposed *Distance-IoU* (DIoU), a new metric which can replace the typical IoU metric in Greedy-NMS, by also considering the distance between the centers of two neighboring detections. Alternatively, Cluster-NMS was proposed in [19], i.e., a method where NMS is performed by implicitly clustering candidate detections, achieving very fast inference runtimes. Finally, a DNN-based NMS method called *Seq2Seq-NMS* [20] achieved top person detection results by assuming a sequence-to-sequence formulation of the NMS problem, exploiting the Multihead Scale-Dot Product Attention mechanism and jointly processing both geometric and visual properties of the input candidate detections.

This paper introduces *AUTH-Persons*, a large-scale dataset containing videos that depict human crowds from an aerial point of view and is suitable for training/evaluating relevant person detection methods. The paper makes the following contributions:

- AUTH-Persons is presented and made publicly available. It contains both synthetic videos, from diverse and realistic landscape environments, and real-world videos collected at the campus of the Aristotle University of Thessaloniki, Greece. In both cases, the footage was collected using UAVs performing flights at various altitudes, while crowds of people were present on the ground. All video frames are annotated with 2D bounding boxes.
- AUTH-Persons was constructed with the explicit aim to study person detection performance in the presence of distribution shifts between the training and the test set in crowded scenes. This has barely been explored in the literature. Thus, experimental evaluation of multiple DNN-based person detectors is conducted on AUTH-Persons in a manner that allows us to assess their ability to generalize to environments that are visually distinct from those they have been trained with.
- To complement this study, a lightweight version of YOLOv4 [21] is employed to evaluate the performance of several NMS methods at the end of the person detection pipeline, in an attempt to examine the negative impact of this data distribution shift on them. Such a study is of particular interest, because NMS algorithms are particularly susceptible to performance degradation in crowded scenes.

The dataset is available at: `https://aiia.csd.auth.gr/open-multidrone-datasets/`.

2. DATASET DESCRIPTION

AUTH-Persons is a UAV video dataset containing 53 videos, summing to footage with a total duration of 37.31 minutes. It is suitable for training and evaluating methods related to the person detection task. Overall, 4 of those videos were collected from a *DJI Phantom 4* while performing flights in the campus of the Aristotle University of Thessaloniki, Greece. The resolution of real-world video frames is 3840×2160 pixels. The remaining videos were collected in virtual environments, using AirSim [22] and a set of environments we designed on Unreal Engine 4[1]. We designed 8 environments, aiming to realistically simulate various environmental conditions (e.g., snow, fog, etc.) in rural landscapes. All environments were populated with a large number of humans and obstacles (e.g., trees, structures, etc.), in order to achieve a high level of occlusions. The footage was collected from a virtual UAV, while orbiting around in various altitudes. The resolution of these synthetic video frames is 1280×720 pixels. Video frames from *AUTH-Persons* are depicted in Figure 1. The frame-rate of all videos is set to 30 fps. The average number of people depicted in each frame is 14.79. 2D bounding boxes of humans are provided as annotations for each frame. Details about the structure of the dataset are provided in Table 1.

3. EXPERIMENTAL EVALUATION

The evaluation conducted on AUTH-Persons, using recent DNN-based object detectors and NMS methods, is presented here. Exploiting the diversity of the dataset, AUTH-Persons was split so that the test set contains environments that are visually distinct from those included in the training set. Thus, environments I-VII were selected for training and the rest for testing. This setup simulates the case where a person detector,

[1] `https://www.unrealengine.com/en-US/`

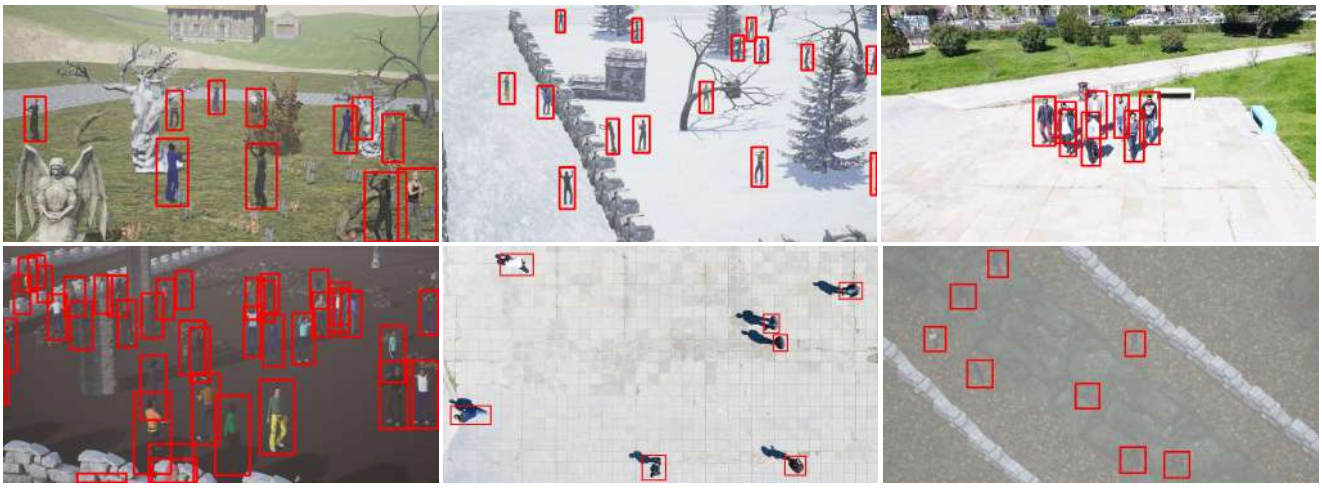


Fig. 1: Video frames from the *AUTH-Persons* dataset, depicting both real and synthetic environments. The ground-truth bounding boxes surrounding visible humans are depicted in red.

Table 1: Structure of the AUTH-Persons Dataset.

Env. ID	Synthetic/ Real-World	Num. of Videos	Duration [mm:ss]	Resolution	Aver. Num. of Persons per Frame
I	S	7	03:33	1280×720	13.50
II	S	6	03:34	1280×720	13.95
III	S	6	04:29	1280×720	16.14
IV	S	4	03:14	1280×720	18.15
V	S	6	03:27	1280×720	31.34
VI	S	10	05:19	1280×720	10.75
VII	R	1	02:22	3840×2160	8.69
VIII	S	3	02:12	1280×720	16.93
IX	S	7	05:41	1280×720	15.67
X	R	3	03:40	3840×2160	3.9
–	–	53	37:31	–	14.79

embedded in an autonomous UAV, is deployed on an unseen environment. The goal was to measure the impact of such a data distribution shift on the detector's precision, as well as on the performance of its integrated DNN-based NMS method.

Evaluation of Person Detection Methods: First, we report results of three person detectors: the Single Shot Detector (SSD) [23], YOLOv3 [24] and YOLOv4-tiny [21]. Various Convolutional Neural Networks (CNNs) are employed as backbone feature extractors. Both the training and the test set were constructed by sampling 1 out of 10 consecutive video frames. All detectors were trained for 15 epochs. Their learning rate was initially set to 5×10^{-4} and it was decreased twice by multiplying it with 0.1 at epochs 10 and 13, respectively. The remaining training hyperparameters were adjusted in the best possible manner, so as to achieve a fair comparison. Greedy-NMS with a 0.6 IoU threshold was applied as the last step on all detectors. The results on both sets are reported in Table 2.

All detectors exhibit a significant precision drop of at least 8% in $AP_{0.5}$ in the test set, compared to the training set. The obvious explanation is the distribution shift between the training and the test data, due to the visually different environments depicted in those two sets. The best precision rates in the test set were achieved by YOLOv4-tiny, although it did not achieve top precision in the training set. Overall, YOLOv4-tiny seems to be the most robust method.

Table 2: Person Detection Evaluation.

Detector	Training Set		Test Set	
	$AP_{0.5}$	$AP^{0.95}_{0.5}$	$AP_{0.5}$	$AP^{0.95}_{0.5}$
SSD-512 (VGG16_atrous)	88.6%	52.3%	76.7%	40.4%
SSD-512 (ResNet50)	85.8%	47.8%	73.8%	36.7%
SSD-512 (MobileNetV1-1.0)	84.3%	43.7%	68.4%	31.1%
YOLOv3-512 (DarkNet53)	**94.6%**	61.5%	81.9%	**48.4%**
YOLOv3-512 (MobileNetV1-1.0)	92.3%	55.9%	77.4%	41.5%
YOLOv3-800 (MobileNetV1-1.0)	**94.6%**	**61.9%**	79.8%	47.2%
YOLOv4-tiny-608 (CSPDarknet53-tiny)	93.7%	56.3%	**85.0%**	47.3%

Evaluation of NMS Methods: YOLOv4-tiny was selected as the main person detector for the evaluation of the NMS methods. In this setup we compare the performance of the recently proposed Seq2Seq-NMS [20] and a wealth of other state-of-the-art NMS methods. The second competing method is a baseline Greedy-NMS approach running on CPU. The third is TorchVision's [2] Greedy-NMS implemented to run very fast on GPUs. Additionally, the non-neural approach Soft-NMS [15] was tested, using both the proposed linear and the Gaussian weighting functions (referred to as Soft-NMS_L and Soft-NMS_G, respectively). The method was executed on CPU. Furthermore, several variants of the more recent Cluster-NMS [19] non-neural approach were selected for comparison purposes. In what follows, the term Cluster-NMS_S is used to imply the case where the score penalty mechanism is used, and Cluster-NMS_D the scenario where the normalized central point distance is added. In the latter case, the method is equivalent to DIoU-NMS [18]. Moreover, the term Cluster-NMS_{S+D} is used when both of these mechanisms are utilized. Finally, Cluster-NMS_{S+D+W} indicates the case where a weighted strategy is applied. The last selected approach is

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

GossipNet [16], a neural NMS method achieving good precision.

The vast majority of NMS methods operate by only analyzing geometric relations between raw candidate detection ROIs in an image. Very few, such as Seq2Seq-NMS, exploit also visual appearance information. However, the visual data distribution shift between the training and the test samples may disproportionately affect in a negative manner those DNN-based NMS methods which do exploit appearance-based features in comparison to those who do not. Thus, Seq2Seq-NMS was evaluated in two variants: a) the vanilla Seq2Seq-NMS [20], and b) a trivial variant Seq2Seq-NMS$_{geom}$ which only exploits geometry-based features without considering visual appearance. Seq2Seq-NMS$_{geom}$ was implemented by simply feeding the DNN a zero vector for each ROI, as a dummy appearance-based feature.

Table 3: Comparison of different NMS methods on AUTH-Persons dataset using detections from YOLOv4-tiny [21].

Method	**Device**	**Test set**		**Average Inference Time (ms)**
		AP$_{0.5}$	**AP**$^{0.95}_{0.5}$	
Greedy-NMS IoU>0.5	CPU	84.7%	46.8%	1.8
Greedy-NMS IoU>0.6	CPU	85.0%	47.3%	1.9
Greedy-NMS IoU>0.7	GPU	83.9%	47.5%	2.1
Original NMS IoU>0.5 TorchVision	GPU	84.7%	46.9%	0.4
Original NMS IoU>0.6 TorchVision	GPU	84.9%	47.3%	0.4
Original NMS IoU>0.7 TorchVision	GPU	83.8%	47.5%	0.4
Soft-NMS$_{\text{L}}$	CPU	84.5%	**48.0%**	2.5
Soft-NMS$_{\text{G}}$	CPU	84.6%	47.9%	1.9
Cluster-NMS	GPU	84.9%	47.4%	1.3
Cluster-NMS$_{\text{S}}$	GPU	84.3%	47.8%	1.6
Cluster-NMS$_{\text{D}}$	GPU	85.0%	47.2%	1.9
Cluster-NMS$_{\text{S+D}}$	GPU	84.5%	47.9%	2.0
Cluster-NMS$_{\text{S+D+W}}$	GPU	84.4%	47.4%	15.6
GossipNet	GPU	85.4%	47.4%	5.6
Seq2Seq-NMS	GPU	85.2%	46.9%	15.8
Seq2Seq-NMS$_{geom}$	GPU	**85.5%**	**48.0%**	15.8

The hyperparameters of all competing NMS methods were set according to the original respective papers. In order to account for the extremely large number of raw detections in several frames, we first apply TorchVision NMS with the relaxed 0.8 IoU threshold on all deployed methods as a typical preprocessing step. A similar scheme was also employed in the training and the testing phases of GossipNet and Seq2Seq-NMS. All experiments were performed on a PC using an Intel Core i7-9700 CPU and an NVIDIA GeForce RTX 2070 GPU with 8GB of memory, both for training and inference.

NMS evaluation results on AUTH-Persons are provided in Table 3. Vanilla Seq2Seq-NMS, which exploits both appearance-based and geometry-based features, achieves an $AP_{0.5}$ of 85.2%, which is the second highest after GossipNet. Seq2Seq-NMS$_{geom}$ improves upon vanilla Seq2Seq-NMS in $AP_{0.5}$ and $AP^{0.95}_{0.5}$ by +0.3% and +1.1%, respectively, thus rendering it the overall best NMS method for AUTH-Persons. Notably, Soft-NMS$_L$ also achieves a top $AP^{0.95}_{0.5}$ score, equal to that of Seq2Seq-NMS$_{geom}$. The non-negligible improvement of Seq2Seq-NMS$_{geom}$ over vanilla Seq2Seq-NMS confirms our intuition that NMS methods exploiting appearance-based features suffer more in the presence of distribution shifts, compared to methods that only exploit geometry-based features.

4. CONCLUSIONS

Detecting humans on video footage captured on-the-fly by UAVs is a challenging, yet critical task for ensuring flight safety in the case of autonomous drones. However, aerial detection of individual people in crowds under the presence of distribution shifts is still very challenging. This paper presented AUTH-Persons in order to facilitate relevant research. It is a new, publicly available video dataset that consists of both real and synthetic footage, summing approximately to a duration of 37 minutes. The dataset is suitable for training and evaluating aerial-view person detection algorithms. The synthetic data were collected from 8 visually distinct photorealistic outdoor environments and they mostly contain scenes with crowded areas, where heavy occlusions and high person densities pose challenges to common detectors. The generalization performance of various state-of-the-art DNN-based object detectors was evaluated on AUTH-Persons, by testing them on environments that are visually distinct from those they have been trained on. YOLOv4-tiny was empirically shown to be the most robust person detector. Finally, given that Non-Maximum Suppression (NMS) methods at the end of the person detection pipeline suffer in crowded scenes, they were compared with respect to the degree that train-to-test distribution shifts affect them in such settings. Among DNN-based free-rescoring NMS algorithms, Seq2Seq-NMS achieved top precision, while appearance-based features are more likely to lead to NMS performance degradation under distribution shift, compared to purely geometry-based ones.

5. REFERENCES

[1] F. Nex and F. Remondino, "UAV for 3D mapping applications: a review," *Applied Geomatics*, vol. 6, no. 1, pp. 1–15, 2014.

[2] I. Mademlis, V. Mygdalis, N. Nikolaidis, M. Montagnuolo, F. Negro, A. Messina, and I. Pitas, "High-level multiple-UAV cinematography tools for covering outdoor events," *IEEE Transactions on Broadcasting*, vol. 65, no. 3, pp. 627–635, 2019.

[3] I. Karakostas, I. Mademlis, N. Nikolaidis, and I. Pitas, "Shot type constraints in UAV cinematography for au-

tonomous target tracking," *Elsevier Information Sciences*, vol. 506, pp. 273–294, 2020.

[4] I. Karakostas, I. Mademlis, N. Nikolaidis, and I. Pitas, "Shot type feasibility in autonomous UAV cinematography," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[5] I. Mademlis et al., "A multiple-UAV architecture for autonomous media production," *Springer Multimedia Tools and Applications*, pp. 1–30, 2022.

[6] 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.

[7] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, "Learning social etiquette: Human trajectory understanding in crowded scenes," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[8] M. Barekatain, M. Martí, H.F. Shih, S. Murray, K. Nakayama, Y. Matsuo, and H. Prendinger, "Okutama-action: An aerial view video dataset for concurrent human action detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017, pp. 2153–2160.

[9] P. Zhu, L. Wen, D. Du, X. Bian, H. Fan, Q. Hu, and H. Ling, "Detection and tracking meet drones challenge," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021.

[10] L. Pishchulin, A. Jain, C. Wojek, M. Andriluka, T. Thormählen, and B. Schiele, "Learning people detection models from few training samples," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[11] 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*, 2021.

[12] A. Khadka, P. Remagnino, and V. Argyriou, "Synthetic crowd and pedestrian generator for deep learning problems," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[13] L. Songtao, H. Di, and W. Yunhong, "Adaptive nms: Refining pedestrian detection in a crowd," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-IoU loss: Faster and better learning for bounding box regression," *Association for the Advancement of Artificial Intelligence (AAAI)*, vol. 34, no. 7, pp. 12993–13000, 2020.

[19] 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.

[20] C. Symeonidis, I. Mademlis, I. Pitas, and N. Nikolaidis, "Neural attention-driven non-maximum suppression for person detection," *TechRxiv*, vol. 10.36227/techrxiv.16940275.v1, 2021.

[21] J. Zicong, Z. Liquan, L. Shuaiyang, and J. Yanfei, "Real-time object detection method based on improved yolov4-tiny," *ArXiv*, vol. abs/2011.04244, 2020.

[22] S. Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics (FSR)*, 2017.

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

[24] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

8.14 Efficient Feature Extraction for Non-Maximum Suppression in Visual Person Detection

The appended paper follows.

EFFICIENT FEATURE EXTRACTION FOR NON-MAXIMUM SUPPRESSION IN VISUAL PERSON DETECTION

Charalampos Symeonidis, Ioannis Mademlis, Ioannis Pitas and Nikos Nikolaidis

Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece
E-mail: {charsyme, imademlis, pitas, nnik}@csd.auth.gr

ABSTRACT

Non-Maximum Suppression (NMS) is a post-processing step in almost every visual object detector, tasked with rapidly pruning the number of overlapping detected candidate rectangular Regions-of-Interest (RoIs) and replacing them with a single, more spatially accurate detection (in pixel coordinates). The common Greedy NMS algorithm suffers from drawbacks, due to the need for careful manual tuning. In visual person detection, most NMS methods typically suffer when analyzing crowded scenes with high levels of in-between occlusions. This paper proposes a modification on a deep neural architecture for NMS, suitable for such cases and capable of efficiently cooperating with recent neural object detectors. The method approaches the NMS problem as a rescoring task, aiming to ideally assign precisely one detection per object. The proposed modification exploits the extraction of RoI representations, semantically capturing the region's visual appearance, from information-rich feature maps computed by the detector's intermediate layers. Experimental evaluation on two common public person detection datasets shows improved accuracy against competing methods, with acceptable inference speed.

Index Terms— Non-Maximum Suppression, Object Detection, Scaled-Dot Product Attention, Person Detection

1. INTRODUCTION

Non-Maximum Suppression (NMS) is a final refinement step incorporated to almost every visual object detection framework, where any detected rectangular Regions-of-Interest (RoIs, defined in pixel coordinates) that spatially overlap are merged/filtered. The problem it attempts to solve arises from the tendency of many detectors to output multiple, neighbouring candidate object RoIs for a single given visible object, due to their implicit sliding-window nature. NMS methods typically rescore the raw candidate detections/RoIs outputted by the detector, before thresholding these modified scores so that, ideally, only a single RoI is finally retained for each visible object.

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

The de facto standard in NMS for object detection is GreedyNMS [1]. It selects high-scoring detections and deletes less confident neighbours, since they most likely cover the same object. Its simplicity and speed make it competitive against proposed alternatives, given that rapid execution is of utmost importance in NMS. An Intersection-over-Union (IOU) threshold determines which less-confident neighbors are suppressed by a detection. Most NMS algorithms, including GreedyNMS, do not make any extra effort to jointly process the RoIs and assign one detection per object. In addition, this fixed IOU threshold leads GreedyNMS to failure in certain cases. For instance, wide suppression may remove detections that cover objects with lower scores, while too low a threshold is unable to suppress duplicate detections.

A typical case where most NMS methods struggle to perform is when they operate on images depicting objects in complex scenes, where several in-between occlusions appear. This occurs frequently when detecting persons/pedestrians within human crowds. This is a very important scenario for security- or safety-critical applications [2] [3] [4].

Over the years, several methods have been proposed as alternatives to GreedyNMS, achieving faster inference times or improved accuracy. Both non-neural algorithms and, more recently, deep learning (DL) have been employed to this end (see Section 2).

However, the vast majority of existing methods only exploit geometric properties/interrelations between the candidate RoIs, in the form of geometric features. An exception specifically for the case of person detection is Seq2Seq-NMS [5], a deep neural architecture which approaches NMS as a sequence-to-sequence problem. Seq2Seq-NMS extracts RoI representations based on geometric *and* visual appearance properties of the input candidate RoIs. An efficient implementation of FMoD [6] is employed for visual RoI description. These RoI representations are then refined by the Seq2Seq-NMS, by capturing relations of neighboring RoIs and aiming to ideally assign precisely one detection per person. A more recent variant of Seq2Seq-NMS, which used as input only geometric RoI representations, was presented in [7] showing that it can achieve improved accuracy rates when deployed on distribution shift scenarios.

Motivated by the relative lack of NMS methods exploit-

ing high-level, semantically meaningful representations of the candidate RoI/detections' visual appearance, this paper proposes a novel variant of Seq2Seq-NMS, named FSeq^2-NMS, which is able to harness the information-rich intermediate feature maps of DL-based object detectors. These are used to derive learned, high-level, semantically meaningful RoI representations, which are then exploited *instead of* handcrafted visual descriptors (such as [6]). FSeq^2-NMS can be easily plugged on top of any DL-based detector, and trained as a separate submodule. The efficacy achieved by the internal/latent image representations of state-of-the-art detectors allows the method to discriminate duplicate RoIs from a set of densely sampled and heavily occluded candidate detections, a problem commonly encountered when detecting humans in crowded scenes. Experiments conducted on two public person detection datasets, widely used for detecting humans in crowded scenes, confirms that FSeq^2-NMS is highly suitable for this scenario, achieving top accuracy.

2. RELATED WORK

Modern attempts to replace GreedyNMS and improve upon it were initially non-neural. Thus, Soft-NMS [8] employs a rescoring function aiming to decrease the score of neighboring less-confident detections, instead of completely eliminating them, achieving better precision and recall rates compared to GreedyNMS. Gaussian and linear weighting functions are utilized, which both require a hyper-parameter tuning similar to GreedyNMS. In [9], the authors replaced the classification scores of candidate detections, used in GreedyNMS, with learned localization confidences to guide NMS towards preserving more accurately localized bounding boxes.

A number of more advanced methods rely on *Distance-IoU* (DIoU) [10], a new metric which can replace the typical IoU metric in GreedyNMS. [10] 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 [11], 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. In [12], the authors presented Representative Region NMS, an approach to effectively remove the duplicate candidate detections in human crowded scenes. The method uses the IoU between the visible parts of two RoIs to determine whether the two full-body boxes are overlapped. In the pedestrian detection task, the novel attribute-aware MMS [13] was proposed, in order to distinguish the pedestrian from a high overlapped group. The proposed method adaptively rejects the false-positive results in the crowded settings.

Due to the prevalence of DL, neural NMS methods started to appear during the late 2010s. In [14], an attention module exploited relations between the input detections, in order to classify them as duplicate or not. Adaptive-NMS [15] is a dynamic thresholding version of GreedyNMS, specifically for detecting humans within crowds. A relatively shallow neural network was designed for predicting a density map and then the proposed method set an adaptive IoU thresholds in NMS for different detections according to the predicted density. In [16], the authors presented GossispNet, a DNN-based NMS method, which jointly analyzes the scores and coordinates of candidates detections in the image, so as not to directly prune them, but to rescore them. GossipNet was modified in [17], for the specific case of person detection from aerial views, by exploiting the handcrafted FMoD descriptor vectors [6] for representing the visual appearance of the candidate RoIs. Seq2Seq-NMS [5], upon which the method presented in this paper relies, also exploited FMoD descriptors for visual RoI representation and incorporated them into a sequence-to-sequence DL neural architecture for candidate RoI rescoring, operating via the Scaled-Dot Product Attention mechanism. Finally, NMS-Loss [18] can be incorporated to almost any single class DL-based object detector, allowing it to be trained with NMS end-to-end and pay attention to the false predictions caused by NMS.

3. PROPOSED METHOD

Seq2Seq-NMS [5] is a DL-based NMS method, aiming to classify an input candidate detection as "correct" or as "potentially suppressed". The label of each candidate detection is formed based on evaluation criteria established in object detection [19] [20]. The method mainly relies on the Scaled Dot-Product Attention mechanism, for exploring relations between neighboring candidate RoIs and finally build discriminative RoI representations for the classification task. As input, Seq2Seq-NMS receives appearance-based and geometric representations for each candidate RoI. In the original version of the method, the authors proposed the use a fast and parallel GPU-bound implementation of FMoD [6], as an optional step, for extracting appearance-based RoI representations. In this implementation, FMoD computed an edge map of the corresponding image, and then extracted an appearance-based representation for each RoI based on statistical properties (e.g., mean, skew, etc.,) of the spatial distribution of the enclosed edges. Based on the corresponding experimental evaluation, Seq2Seq-NMS, along with FMoD, achieved top results against competing methods, proving to be a suitable solution for Non-Maximum Suppression on the person detection task.

Aiming to exploit the representational efficacy of modern DL-based detectors, we propose FSeq^2-NMS, which incorporates an appearance-based RoI representations extraction module, capable to utilize feature-maps, already computed from the in-between layers of the deployed detector. To our intuition, the proposed module will enable FSeq^2-NMS to provide improved results on the challenging task of detecting humans within crowded scenes, by feeding it with more qualitative RoI representations regarding their visual appearance.

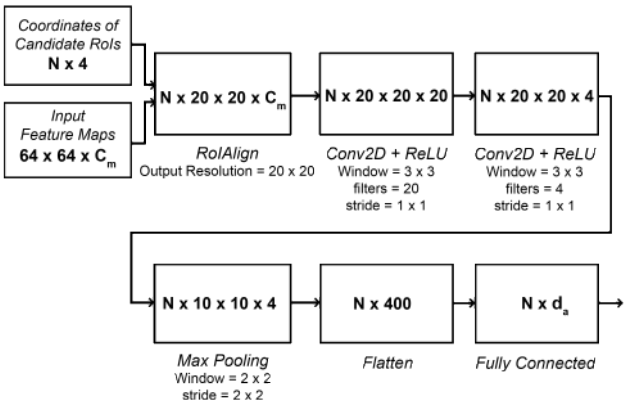


Fig. 1: Appearance-based RoI representations extraction module, capable to utilize feature-maps of the corresponding detector.

The proposed architecture of the appearance-based Roi representations extraction module is depicted on Fig. 1. As input, the module receives $\mathbf{B} = [\mathbf{b}_0, \mathbf{b}_1, .., \mathbf{b}_N] \in \mathbb{R}^{N\times 4}$, which correspond to the coordinates of N candidate RoIs, as well as $\mathbf{M} \in \mathbb{R}^{64\times 64\times C_m}$, which correspond to a set of features maps, extracted from an in-between layer of the deployed detector and resized to a fixed 64×64 resolution. C_m corresponds to the number of the channels of the corresponding feature maps. Using the *RoIAlign* [21] operator, initial RoI maps can be in-parallel extracted in a fixed 20×20 spatial resolution. Then two convolutional layers, with the Rectified Linear Unit (ReLU) as activation function, followed by a max-pooling layer are applied on the extracted RoI maps. The final RoI representations $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, .., \mathbf{a}_N] \in \mathbb{R}^{N\times d_r}$ are computed by flattening the RoI maps and applying a fully connected layer using ReLU as activation function. d_a corresponds to the dimension of the final appearance-based RoI representations.

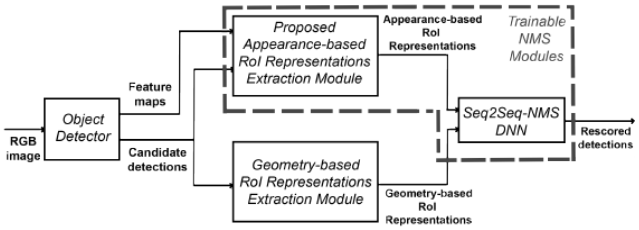


Fig. 2: Pipeline of the overall object detection framework, in which FSeq^2-NMS in employed.

The proposed appearance-based RoI representations extraction module should be trained along with core attention-based Seq2Seq-NMS, as it consists part of the overall structur. However, similar to [5], the training procedure of the proposed architecture of FSeq^2-NMS must be carried out after the training of the deployed detector. The overall detection pipeline, which incorporates FSeq^2-NMS, is depicted in Fig. 2.

4. EXPERIMENTAL EVALUATION

The performance of the proposed Seq2Seq-NMS variant was evaluated on two separate datasets, suitable for detecting humans in crowded scenes. In both datasets, candidate RoIs from the *Single Shot Detector* (SSD) [22] were provided as input to the corresponding NMS methods. In the implemented version of the detector, VGG16 with atrous convolutions was selected as the backbone CNN. The input images were resized to a resolution of 512×512 pixels, while the detector was trained from scratch for each dataset[1].

The core attention-based architecture of Seq2Seq-NMS and the training setup were similar to [5] and any deviations are reported separately on each dataset. Feature-maps from the initial layer of VGG16 were selected as input to the employed appearance-based RoI representations extraction module. Based on this selection, $C_m = 512$. In addition, we set $d_r = 315$.

In both datasets, FSeq^2-NMS was compared against neural and non-neural NMS algorithms. 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. Soft-NMS [8], i.e., a non-neural NMS method widely used as a more accurate replacement for Greedy NMS, was also tested. Additionally, several variants of Cluster-NMS [11], a more recent non-neural method, were also used for comparisons. The last approach selected for comparison purposes is GossipNet [16], a neural NMS method achieving state-of-the-art accuracy. More details regarding these variations can be found in [11]. Additional information about the variant of each competing method can be found in [5].

The hyperparameters of all non-neural methods were tuned so as to report the best achieved results on 0.5 IoU matching threshold. Evaluation was performed on a PC using an Intel Core i7-7700 CPU and an NVIDIA GeForce RTX 2080 GPU with 11GB of memory, both for training and inference. The employed evaluation metrics are $AP_{0.5}$, $AP^{0.95}_{0.5}$ and inference times. $AP_{0.5}$ corresponds to the average precision for 0.5 IoU, while $AP^{0.95}_{0.5}$ to the mean average precision for IoU ranging from 0.5 to 0.95 with a step size of 0.05. Finally, all RoIs outputted by the NMS algorithms were utilized for evaluation, without any thresholding.

4.1. PETS

PETS [23] is a relatively small dataset, whose images were collected from static surveillance cameras and provide diverse levels of occlusion.

[1]The employed SSD implementation was adopted from `https://github.com/opendr-eu/opendr/tree/master/src/opendr/perception/object_detection_2d/ssd`

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

The average number of people depicted in an image is approximately 14. The proposed NMS architecture was trained for 6 epochs. The learning rate was set to $10^{-4}/10^{-5}/10^{-6}$ for epochs 1-3/4-5/6, respectively. GossipNet's architecture and training followed [16]. Final parameters of all methods were selected according to the best achieved accuracy in the validation set.

Table 1 reports the results using cadidate detections from [22]. FSeq^2-NMS achieved an $AP_{0.5}$ of 87.4% and an $AP_{0.5}^{0.95}$ of 38.0% in the validation set, thus attaining gains of $+0.4\%$ $+1.0\%$ over GossipNet in the corresponding metrics. In the testing set, the proposed method achieved an $AP_{0.5}$ of 91.2% and an $AP_{0.5}^{0.95}$ of 38.9% surpassing GossipNet by $+0.5\%$ and $+0.1\%$ in the corresponding metrics.

Table 1: Comparison of different NMS methods on PETS dataset.

Method	Device	Val set		Test set		Average Inference Time (ms)
		$\mathbf{AP}_{0.5}$	$\mathbf{AP}_{0.5}^{0.95}$	$\mathbf{AP}_{0.5}$	$\mathbf{AP}_{0.5}^{0.95}$	
Greedy-NMS	CPU	84.3%	34.7%	89.9%	36.3%	13.1
TorchVision NMS	GPU	84.3%	34.7%	90.0%	36.4%	0.2
Soft-NMS_L	CPU	85.3%	35.9%	90.0%	38.2%	108.8
Soft-NMS_G	CPU	83.9%	36.2%	89.6%	38.6%	134.4
Cluster-NMS	GPU	84.5%	34.3%	90.2%	36.9%	13.4
Cluster-NMS_S	GPU	84.7%	36.0%	90.1%	38.0%	13.8
Cluster-NMS_D	GPU	84.5%	34.5%	90.2%	36.6%	17.9
Cluster-NMS_{S+D}	GPU	85.7%	36.0%	90.6%	38.3%	22.4
Cluster-NMS_{S+D+W}	GPU	85.7%	36.0%	90.6%	38.3%	38.2
GossipNet	GPU	87.1%	37.0%	90.7%	38.8%	24.5
FSeq^2-NMS	GPU	**87.4%**	**38.0%**	**91.2%**	**38.9%**	7.8
Gains		+0.3%	+1.0%	+0.5%	+0.1%	-

4.2. CrowdHuman

The CrowdHuman dataset has been released specifically to target human detection in crowded areas, and has been proved to be a challenging for person detectors, due to heavy visual occlusions of individual humans. The average number of persons in an image is 22.64.

FSeq^2-NMS was trained for 14 epochs. The learning rate was set to $10^{-4}/10^{-5}/10^{-6}$ for epochs 1-8/9-12/13-14, respectively. GossipNet was trained for 10^6 iterations, with a learning rate set to 10^{-4} and decreased by 0.1 at the 6×10^5-th and the 8×10^5-th iterations.

Table 2 shows that the proposed method achieves gains, both in terms of $AP_{0.5}$ and $AP_{0.5}^{0.95}$. FSeq^2-NMS, achieved an $AP_{0.5}$ of 75.3% and an $AP_{0.5}^{0.95}$ of 36.9%, which are $+2.9\%$ and $+1.9\%$ improvements against GossipNet, respectively.

4.3. Discussion

Overall, FSeq^2-NMS incorporating the appearance-based RoI representations extraction module, achieved top accuracy rates on $AP_{0.5}$ and $AP_{0.5}^{0.95}$ metrics in both datasets. The results demonstrate that the proposed module can push the core attention-based Seq2Seq-NMS DNN towards achieving top results in the challenging task of detecting humans in crowded scenes. This is done by exploiting the representational efficacy of modern-DL based detectors towards providing FSeq^2-NMS with enriched RoI representations regarding their visual appearance.

Table 2: Comparison of different NMS methods on CrowdHuman dataset.

Method	Device	Test set		Average Inference Time (ms)
		$\mathbf{AP}_{0.5}$	$\mathbf{AP}_{0.5}^{0.95}$	
Greedy-NMS	CPU	67.0%	32.4%	9.8
TorchVision NMS	GPU	66.9%	32.4%	0.4
Soft-NMS_L	CPU	66.5%	32.3%	54.2
Soft-NMS_G	CPU	67.1%	33.0%	58.1
Cluster-NMS	GPU	67.1%	32.1%	5.0
Cluster-NMS_S	GPU	64.0%	31.0%	5.2
Cluster-NMS_D	GPU	67.1%	32.1%	6.5
Cluster-NMS_{S+D}	GPU	65.7%	31.8%	8.0
Cluster-NMS_{S+D+W}	GPU	65.7%	31.9%	32.3
GossipNet	GPU	72.4%	35.0%	10.0
FSeq^2-NMS	GPU	**75.3%**	**36.9%**	4.8
Gains		+2.9%	+1.9%	-

5. CONCLUSIONS

Successful NMS is challenging when detecting humans in crowded areas with high levels of in-between occlusions. This paper proposed a modification to a DL-based NMS architecture, capable of harnessing the representational efficacy of state-of-the-art neural detectors. The proposed approach, called FSeq^2-NMS, is able to utilize feature maps, extracted from the intermediate detector layers, in order to build semantically rich representations of the candidate RoIs' visual appearance. These are then employed by the NMS Deep Neural Network which this paper improves (Seq2Seq-NMS), for better discriminating whether a candidate detection is duplicate or not. Experiments on two person detection datasets, whose images mostly depict humans in crowded scenes, showed that the proposed method is indeed suitable for such a scenario, achieving top accuracy rates among the competing methods. The results confirm that exploiting semantic visual appearance descriptions of the candidate RoIs is indeed the best option for NMS in person detection within dense crowd images, compared either to geometry-only RoI representations, or to using lower-level statistical visual appearance descriptors (e.g., FMoD).

6. REFERENCES

[1] P. F. Felzenszwalb, R. B. Girshick, D. A. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 1627–1645, 2009.

[2] C. Papaioannidis, I. Mademlis, and I. Pitas, "Fast CNN-based single-person 2D human pose estimation for autonomous systems," *IEEE Transactions on Circuits and Systems for Video Technology*, 2022.

[3] E. Kakaletsis, C. Symeonidis, M. Tzelepi, I. Mademlis, A. Tefas, 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 (CSUR)*, vol. 54, no. 9, pp. 1–37, 2021.

[4] E. Kakaletsis, I. Mademlis, N. Nikolaidis, and I. Pitas, "Multiview vision-based human crowd localization for UAV fleet flight safety," *Signal Processing: Image Communication*, vol. 99, pp. 116484, 2021.

[5] C. Symeonidis, I. Mademlis, I. Pitas, and N. Nikolaidis, "Neural attention-driven non-maximum suppression for person detection," *TechRxiv*, vol. 10.36227/techrxiv.16940275.v1, 2021.

[6] 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)*, 2015.

[7] C. Symeonidis, I. Mademlis, I. Pitas, and N. Nikolaidis, "AUTH-Persons: A dataset for detecting humans in crowds from aerial views," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2022, pp. 596–600.

[8] 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.

[9] 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)*. 2018, Springer.

[10] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-IoU loss: Faster and better learning for bounding box regression," *Association for the Advancement of Artificial Intelligence (AAAI)*, vol. 34, no. 07, pp. 12993–13000, 2020.

[11] Z. Zheng, P. Wang, W. Ren, D.and Liu, R. Ye, Q. Hu, and W. Zuo, "Enhancing geometric factors in model learning and inference for object detection and instance segmentation," *IEEE Transactions on Cybernetics*, vol. 52, no. 8, pp. 8574–8586, 2022.

[12] 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, pp. 10747–10756.

[13] J. Zhang, L. Lin, J. Zhu, Y. Li, Y.-C. Chen, Y. Hu, and S. C. H. Hoi, "Attribute-aware pedestrian detection in a crowd," *IEEE Transactions on Multimedia*, vol. 23, pp. 3085–3097, 2021.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] Z. Luo, Z. Fang, S. Zheng, Y. Wang, and Y. Fu, "NMS-Loss: Learning with non-maximum suppression for crowded pedestrian detection," in *Proceedings of the 2021 International Conference on Multimedia Retrieval*, 2021, p. 481–485.

[19] 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)*, 2014.

[20] 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.

[21] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 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)*, 2016.

[23] 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.

8.15 Real-time synthetic-to-real human detection for robotics applications

: The appended paper follows.

Real-time synthetic-to-real human detection for robotics applications

Maria Tzelepi, Charalampos Symeonidis, Nikos Nikolaidis, and Anastasios Tefas
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
email: {mtzelepi, charsyme, nnik, tefas}@csd.auth.gr.

***Abstract*—During the recent years, Deep Learning achieved exceptional performance in various computer vision tasks, paving auspicious research directions for its application in robotics. A key component for its exceptional performance is the availability of sufficient training data. However obtaining such amount of training data constitutes a challenging task, especially considering robotics applications. Thus, synthetic data have recently been regarded as a promising tool to overcoming the data availability problem. In this work we first build a synthetic human dataset, and then we train a lightweight model, capable of operating in real-time for high-resolution input on low-power GPUs, for discriminating between humans and non-humans. The target of this work is to assess the generalization of the model trained on synthetic data, to real data, and also to explore the effect of using (few) real images in the training phase. As it is shown through quantitative and qualitative results the use of only few real images can beneficially affect of the performance of the synthetic-to-real real-time model.**

***Index Terms*—Synthetic-to-real, human detection, real-time, heatmaps, robotics.**

I. Introduction

During the recent years, Deep Learning (DL) attained widespread popularity due to its exceptional performance on various computer vision tasks [1]–[4]. Its impressive performance on computer vision, paved auspicious research directions for its application in robotics [5]–[8]. A key component for the successful performance of DL algorithms is the availability of sufficient training data. State-of-the-art DL models require millions of training examples [9]. However, obtaining such amount of training data, especially considering robotics applications, constitutes a challenging task. Thus, synthetic data, i.e., data generated artificially rather than by actual events, have recently been regarded as a very promising tool to circumvent the data availability problem [10].

The use of synthetic data is accompanied, in general, by various benefits linked with their low-cost nature and ability to meet specific requirements imposed by the application, which may not be feasible in real data. Thus, synthetic data have been utilized in a wide range of robotics applications, e.g., [11]–[14]. Their application on robotics applications is associated with a series of specific advantages. A few of those follow below: 1) synthetic data provide detailed annotations, since these are automatically produced, without containing errors usually occurring in the manual annotation process; 2) they are usually large in scale, since they are procedurally generated; 3) they minimize the risk of DL methods deployed in simulation environments in robotics to exhibit unstable behaviours or complete failures, due to not having been adapted to the visual differences between the virtual and the real world data.

A key issue associated with the successful use of synthetic data in robotics is the gap between the generated data and their deployment considering real data (that is, synthetic-real gap). The need for bridging this gap has fueled a new research area [15]–[17].

In this work, we first build a synthetic dataset for discriminating between humans and non humans, and use it to train a lightweight fully convolutional model that is capable of operating in real-time (about 25 Frames Per Second - FPS) utilizing a low-power GPU for high resolution input [18]. The target is to use the model to provide semantic heatmaps of human presence on real data. That is, we train the real-time model on the synthetic data, and we test the model on unseen images that contain real humans, producing semantic heatmaps, as explained in [18]. A main objective of this work is to assess the generalization of the model to real data, and investigate the effect of using real images in the training phase. As it is demonstrated in the experimental evaluation the use of even few real training examples can considerably improve the performance of training merely with synthetic data, while this is also reflected in the qualitative evaluation through the produced heatmaps.

The remainder of the manuscript is organized as follows. Section II presents in detail the proposed synthetic-to-real real-time human detection model, including the real-time model and the constructed synthetic dataset. Subsequently, in Section III the experiments conducted to assess the performance the synthetic-to-real real-time model, both quantitatively and qualitatively, are provided. Finally the conclusions are drawn in Section IV.

II. Real-time Synthetic-to-Real Human Detection

In this work we propose a *synthetic-to-real real-time model* for discriminating between humans and non humans. The core objective of this work is to assess the generalization of the

978-1-6654-6390-4/22/$31.00 ©2022 IEEE

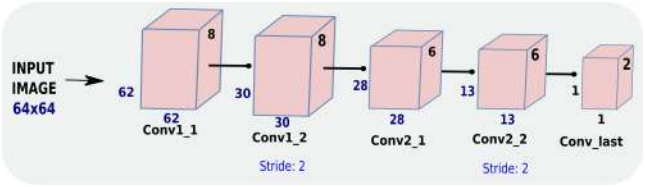


Fig. 1. Architecture of the real-time VGG-1080p model.

Fig. 2. Sample images of Synthetic Human dataset.

model trained on synthetic data, to real data, and also to explore the effect of using (few) real images in the training phase. In the following Sections we describe the real-time model architecture and the generation of the *synthetic human* dataset.

A. Real-Time Model

In this work, we train a fully convolutional lightweight on synthetic data, that is able to operate in real-time for detecting humans in real-images, considering high-resolution input on a low-power GPU. That is, the VGG-1080p model [18] is used, consisting of five convolutional layers 11K parameters. The model's architecture is illustrated in Fig. 1. The model runs in real-time (i.e., 25.6 FPS) on a Jetson TX-2 for 1080p input. More specifically, the network is trained on synthetic images of size 64×64, and then in the test phase, real images of size 1920×1080 are propagated to the network, and for every window 64×64 the output of the network at the output layer is computed, in order to generate the heatmaps of human presence.

B. Synthetic Human Dataset

The *synthetic human* dataset consists of real background images populated with 3D human models in various poses. PIFu [19], a state-of-the-art deep learning method for generating realistic 3D human models from single-view images, is used to generate the human models. The dataset consists in 133 human models, generated using full-body images of people from the Clothing Co-Parsing [20] dataset as PIFu's input. The Cityscapes [21] dataset which is composed of video sequences depicting street scenes in various cities, was used to take background images. The 3D human models are placed on potential 2D image locations (e.g., pavements, roads), based on coarse annotations for semantic image segmentation provided by Cityscapes, so as to manage a higher level of realism.

Since, the target is to train models that can run in real-time on high-resolution input for producing heatmaps of human presence [18], the generated images are cropped, and a train set of 20,000 synthetic cropped images containing humans is constructed. The train set also contains 20,000 non human images, cropped from images of the Cityscapes dataset. The test set consists of 4,000 real images containing humans and 4,000 real images without humans, cropped from video frames that were gathered by querying YouTube video search engine with random keywords. The cropped images are of size 64×64. Since a main objective of this work is to evaluate the effect of real-human images on the train set, we also construct four additional versions of the train set where 100, 200, 500, and 1000 out of 20,000 images are real-human images, while the rest are synthetic. The real human images are derived from the CUHK Person Re-identification datasets [22], [23]. Sample images of the constructed dataset are provided in Fig. 2.

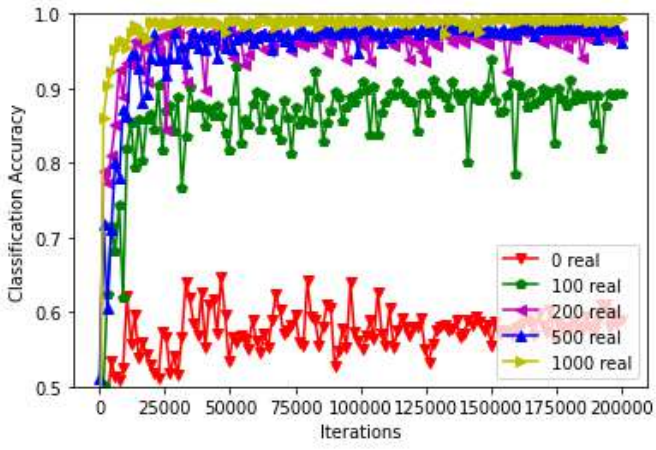


Fig. 3. Classification accuracy using the synthetic-to-real real-time model trained with 0, 100, 200, 500, and 1000 real images throughout the training iterations.

III. Experimental Evaluation

A. Evaluation Metrics and Implementation Details

Two sets of experiments were conducted. First, the performance of the synthetic-to-real real-time human detection model is evaluated using classification accuracy (test accuracy) as evaluation metric. Furthermore, the training curves of classification accuracy throughout the training iterations. Second, qualitative results are provided using the proposed synthetic-to-real real-time model. The model is used to produce heatmaps of human presence on real unseen high-resolution test images. The models are trained for 200,000 iterations (i.e., 320 epochs) using the mini-batch gradient descent with mini-batch of 64 samples, and we set the learning rate to 10^{-3}.

B. Experimental Results

In Table I we provide the classification accuracy of the synthetic-to-real real-time model trained merely with synthetic data, and with 100, 200, 500, and 1000 real images. As it is demonstrated, the model trained only with synthetic humans achieves sufficient performance, while as we include real

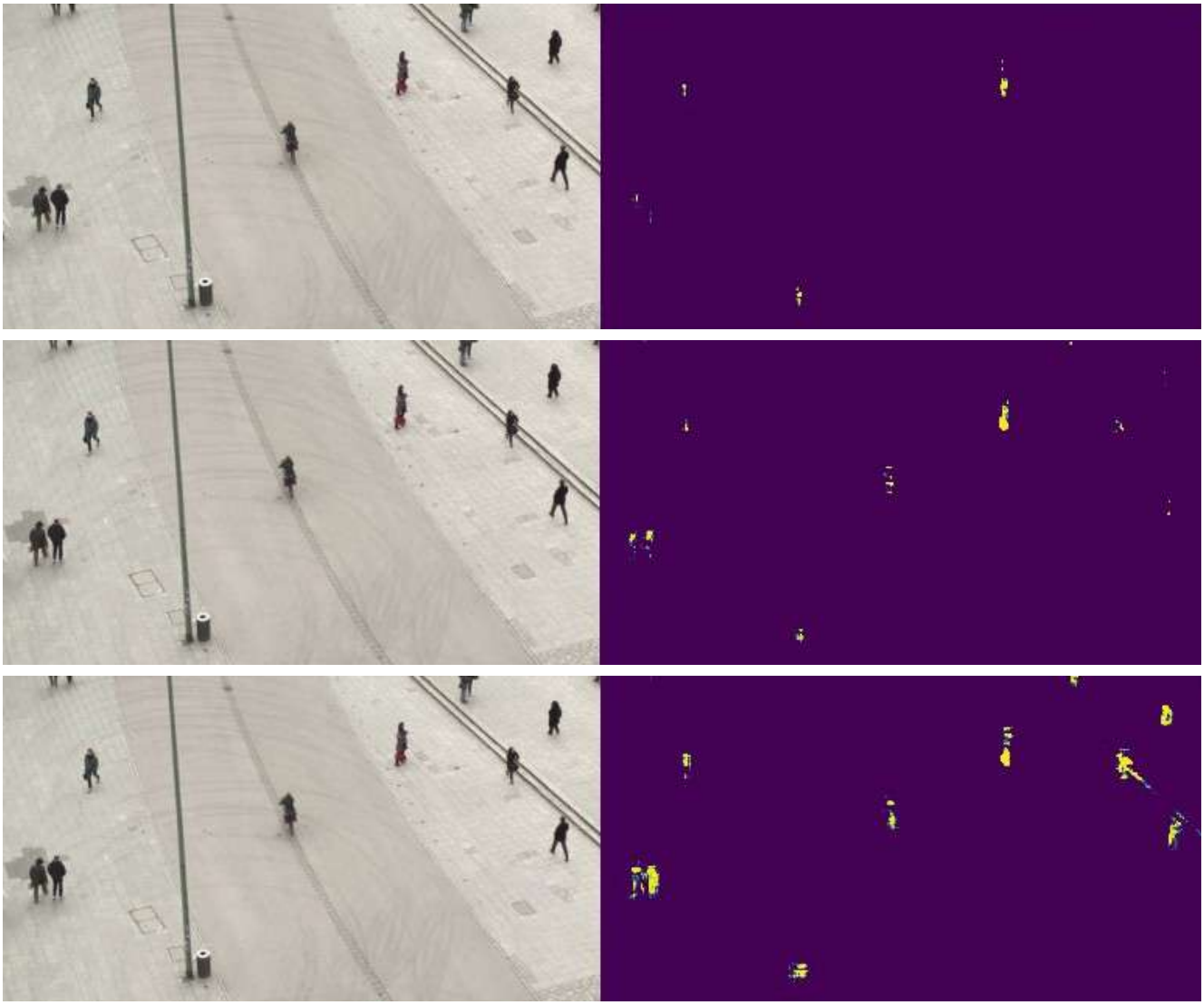


Fig. 4. Heatmaps on real-image containing humans using the synthetic-to-real real-time model trained with 0, 500, and 1000 real images respectively.

human images, we can accomplish progressively increased performance. We can notice that even by adding only 100 real images the performance is remarkably improved. Furthermore, the same remarks are drawn in Fig. 3, where the training curves of the synthetic-to-real real-time model trained with 0, 100, 200, 500, and 1000 real images throughout the training iterations, are illustrated. Furthermore, another important remark is that the more real images we include in the training procedure, the more stable the performance is. That is, we notice that when training only with synthetic data, apart from the poorer performance in terms of classification accuracy, the model also exhibits unstable performance. This is also occurs in the case of training with only 100 real images, while when training with 200, and especially with 500 and 1000 real images a more stable performance is managed.

Finally, in the second set of experiments, we use the proposed trained model on the synthetic human dataset to generate heatmaps on unseen high-resolution images that contain real humans. That is, as previously mentioned, unseen images of size 1920×1080 are fed to the network, and for every window 64×64 we compute the output of the network at the output layer. First, in Fig. 4 we provide the heatmaps on an unseen high-resolution image, with the model trained with none, 500, and 1000 real images respectively. As it is

TABLE I
CLASSIFICATION ACCURACY USING THE SYNTHETIC-TO-REAL REAL-TIME MODEL TRAINED WITH 0, 100, 200, 500, AND 1000 REAL IMAGES.

N. of real images	Classification accuracy
0	0.7725
100	0.9546
200	0.9848
500	0.9871
1000	0.9958

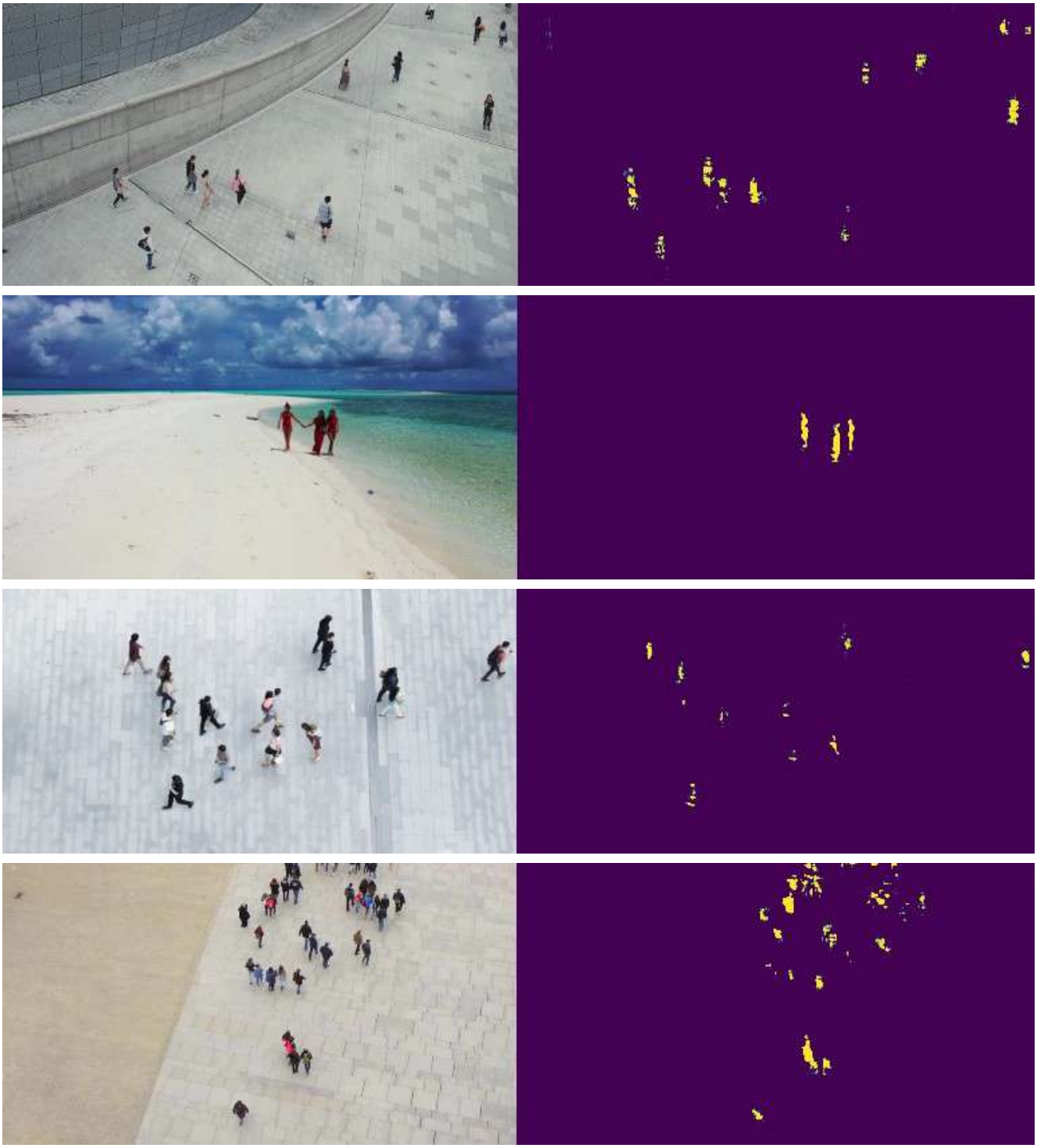


Fig. 5. Heatmaps on real-images containing humans using the synthetic-to-real real-time model trained with 1000 real images.

shown, the beneficial effect of including a few real images in the training , demonstrated in the first set of experiments, is also reflected in the qualitative results. That is, as it is shown in the produced heatmaps, while using only synthetic data, only a few humans can be detected, when using 1000 real images in the training, all of them can be detected. Finally, in Fig. 5, we provide some heatmaps using the synthetic-to-real real-time model, trained with only 1000 real images. As it is demonstrated, the model achieves remarkable performance on detecting real humans.

IV. Conclusions

In this paper, we dealt with synthetic data considering robotics applications. More specifically, we first built a synthetic human dataset, and then we trained a lightweight model, capable of running in real-time for high-resolution input, for discriminating between humans and non-humans. The objective of this work is to assess the generalization of the model trained on synthetic data, to real data, and also to investigate the effect of using (few) real images in the training phase. As it is demonstrated in the experimental evaluation, the use of only few real images can beneficially affect of the performance of the synthetic-to-real real-time model.

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] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational intelligence and neuroscience*, vol. 2018, 2018.

[2] M. Tzelepi and A. Tefas, "Deep convolutional learning for content based image retrieval," *Neurocomputing*, vol. 275, pp. 2467 – 2478, 2018.

[3] C. Nasioutzikis, M. Tzelepi, and A. Tefas, "Deep hashing regularization towards hamming space retrieval," in *11th Hellenic Conference on Artificial Intelligence*, 2020, pp. 74–77.

[4] M. Tzelepi and A. Tefas, "Quadratic mutual information regularization in real-time deep cnn models," in *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2020, pp. 1–6.

[5] H. A. Pierson and M. S. Gashler, "Deep learning in robotics: a review of recent research," *Advanced Robotics*, vol. 31, no. 16, pp. 821–835, 2017.

[6] A. I. Károly, P. Galambos, J. Kuti, and I. J. Rudas, "Deep learning in robotics: Survey on model structures and training strategies," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 1, pp. 266–279, 2020.

[7] N. Passalis, S. Pedrazzi, R. Babuska, W. Burgard, D. Dias, F. Ferro, M. Gabbouj, O. Green, A. Iosifidis, E. Kayacan *et al.*, "Opendr: An open toolkit for enabling high performance, low footprint deep learning for robotics," *arXiv preprint arXiv:2203.00403*, 2022.

[8] M. Tzelepi and A. Tefas, "Semantic scene segmentation for robotics applications," in *2021 12th International Conference on Information, Intelligence, Systems & Applications (IISA)*. IEEE, 2021, pp. 1–4.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[10] S. I. Nikolenko, "Synthetic data for deep learning," *arXiv preprint arXiv:1909.11512*, 2019.

[11] D. Ward, P. Moghadam, and N. Hudson, "Deep leaf segmentation using synthetic data," *arXiv preprint arXiv:1807.10931*, 2018.

[12] Z. Tang, M. Naphade, S. Birchfield, J. Tremblay, W. Hodge, R. Kumar, S. Wang, and X. Yang, "Pamtri: Pose-aware multi-task learning for vehicle re-identification using highly randomized synthetic data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 211–220.

[13] Y. Lin, C. Tang, F.-J. Chu, and P. A. Vela, "Using synthetic data and deep networks to recognize primitive shapes for object grasping," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 10 494–10 501.

[14] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020, pp. 737–744.

[15] M. Yan, Q. Sun, I. Frosio, S. Tyree, and J. Kautz, "How to close sim-real gap? transfer with segmentation!" *arXiv preprint arXiv:2005.07695*, 2020.

[16] W. Chen, Z. Yu, Z. Wang, and A. Anandkumar, "Automated synthetic-to-real generalization," in *International Conference on Machine Learning*. PMLR, 2020, pp. 1746–1756.

[17] W. Zhao, J. P. Queralta, L. Qingqing, and T. Westerlund, "Towards closing the sim-to-real gap in collaborative multi-robot deep reinforcement learning," in *2020 5th International Conference on Robotics and Automation Engineering (ICRAE)*. IEEE, 2020, pp. 7–12.

[18] M. Tzelepi and A. Tefas, "Improving the performance of lightweight cnns for binary classification using quadratic mutual information regularization," *Pattern Recognition*, vol. 106, p. 107407, 2020.

[19] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li, "Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[20] W. Yang, P. Luo, and L. Lin, "Clothing co-parsing by joint image segmentation and labeling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[21] 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 *PProceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[22] W. Li, R. Zhao, and X. Wang, "Human reidentification with transferred metric learning," in *ACCV*, 2012.

[23] W. Li and X. Wang, "Locally aligned feature transforms across views," in *CVPR*, 2013.