University of Modena and Reggio Emilia
XXXIV cycle of the International Doctorate School in
Information and Communication Technologies
Doctor of Philosophy dissertation in
Computer Engineering and Science

Deep video understanding for human actions, multimedia and computing

Video understanding, action detection and a computational investigation
Grant by Metaliquid

Matteo Tomei

Supervisor: Prof. Rita Cucchiara
PhD Course Coordinator: Prof. Sonia Bergamaschi

Modena, 2022

Review committee composed of:

Prof. Vittorio Murino, *University of Verona*
Dr. Pascal Mettes, *University of Amsterdam*

Contents

Chapter 1

Introduction

Videos represent the most spread information medium nowadays, mainly converging to social networks and streaming services. This is not surprising, given their ability to gather different messages in a single container (spatial composition of the scene, temporal sequence of events, audio information). Video automatic understanding would be of extreme importance in many fields, ranging from surveillance in security scenarios, to indexing and retrieval for broadcast service providers, and to healthcare monitoring applications. This broad spectrum of possibilities pushed the video analysis research community to propose a number of video recognition tasks, based on the specific field of application.

With the successful spread of deep learning, data has recently become the foundation for developing impressive models, in terms of their ability to solve a task. Data has a major impact on the task itself, on the size of the model, and on its performance. However, close attention should be paid when handling large datasets, and even more when this data could potentially contain sensitive information: video datasets often exhibit both of these features.

This thesis focuses on the research line of human actions in the context of video understanding, with deep learning as a foundation. It is the result of a doctorate granted by Metaliquid S.R.L. and it follows several research lines dictated by the company's needs. Not only common video-related tasks are addressed, but also computation-specific aspects are tackled, given the limitations and constraints experienced when handling video datasets. Moreover, a specific analysis regarding privacy issues is carried out. All the presented solutions have been developed taking into account two major goals: to follow the needs of the research community,

requiring novel architectures breaking the state of the art on publicly available data, and to develop production-oriented applications necessary for the multimedia industry, given the continuous collaboration with Metaliquid.

1.1 Organization and summary

After providing in Chapter 2 an overview of the main existing solutions in different fields of video recognition, in Chapter 3 we tackle the spatio-temporal action detection task, aiming to detect people in the scene and to classify their actions. It requires a fine-grained representation of the video, resulting in a variety of possible applications, like video categorization, retrieval, and sensitive content detection. In our formulation, we heavily rely on the role of context, exploiting the standard approach in literature of separating the detection task from that of classification: if we are able to obtain representative features for actors and objects in a scene, we can build a graph between these entities and learn their relationships. In addition, the relation optimization can be performed independently of the feature extraction process, resulting in a stand-alone and high-level module.

In Chapter 4, we explore a more production-oriented application of video activity understanding, still without disregarding the research needs. Specifically, we tackle the soccer event spotting task, representing the first step towards automatic highlight generation from broadcast soccer games. Event spotting requires localizing the exact timestamp in which an interesting action occurs, and to classify it. Differently from spatio-temporal action detection, spotting events in a soccer match can be achieved by a coarse-grained representation of the video: interesting events are quite scattered throughout the video, and high-level visual cues are usually enough to suggest the presence of a highlight. Inspired by other works in the field, we propose a lightweight convolutional network focusing on the frames *after* the event itself, since they usually contain the majority of visual cues (for instance, celebrations or replays are common after a goal). Moreover, the resulting model placed third in a CVPR 2021 ActivityNet challenge.

After starting to work with videos, we soon noticed two inherent and common challenges of video handling, namely privacy and computation. Publicly available action recognition datasets usually ignore the sensitive information contained in their videos (like actors' faces). This has limited consequences in research, when working on public data collected on purpose. In a production environment, where the action recognition service is usually run by a third-party service provider, the user might want to not transmit the original video, which could potentially

compromise his privacy. In Chapter 5, we propose a general teacher-student architecture that exploits sensitive information only during training, while working on privacy-preserving videos (*e.g.* with blurred faces) during inference, with negligible loss in accuracy.

As a second drawback, video understanding pipelines usually require huge resources for both training and inference, limiting access to this research area and increasing the costs in production environments. We propose an iterative algorithm able to reduce the size (in terms of both parameters and floating-point operations) of existing state-of-the-art video understanding models, as detailed in Chapter 6. The core idea relies on the shrinkage of one dimension at a time, where each dimension partially determines the size of the model. The shrinkage iterations are performed by maximizing the accuracy of the reduced model, which should always mimic the behavior of the original bigger one.

Sintesi in lingua italiana

I video rappresentano oggi il mezzo più diffuso di condivisione dell'informazione, trovando nei social network e nei servizi di streaming le principali vie di trasmissione. La diffusione dei video è dovuta principalmente alla loro capacità di mettere insieme messaggi di tipo diverso (la vicinanza spaziale tra le entità rappresentate, la sequenza temporale degli eventi, le informazioni audio). La comprensione automatica dei video è quindi un obiettivo di vitale importanza in diverse aree applicative, dalla sorveglianza in scenari di sicurezza, all'indicizzazione per i fornitori di servizi broadcast, alle applicazioni di monitoraggio. Questo ampio spettro di possibilità ha spinto la comunità scientifica a concentrarsi su diverse attività di riconoscimento attraverso la collezione di diversi dataset e lo sviluppo di modelli predittivi. In alcuni casi è richiesto un riconoscimento dettagliato degli attori e degli oggetti nella scena, mentre in altre circostanze è sufficiente una rappresentazione ad alto livello del video stesso.

Questa tesi propone algoritmi e metodi per l'analisi automatica delle azioni umane nei video, attraverso l'uso di tecniche di deep learning e visione artificiale. Non solo vengono affrontati alcuni dei principali problemi relativi al riconoscimento di azioni, ma vengono anche analizzati e discussi i loro aspetti computazionali. Vengono inoltre sviluppati temi in materia di privacy, argomento ancora poco considerato dalla comunità scientifica. Tutte le soluzioni presentate sono state sviluppate tenendo in considerazione due obiettivi principali: far fronte alle esigenze della ricerca, che richiede nuove architetture in grado di ottenere performance sempre migliori su dati pubblicamente disponibili, e sviluppare ap-

plicazioni orientate alla produzione e necessarie all'industria multimediale, dato il ruolo e la collaborazione con Metaliquid SRL nel corso del mio dottorato.

Dopo una sintesi delle principali soluzioni presenti in letteratura (Capitolo 2), nel Capitolo 3 viene affrontato il problema della localizzazione spazio-temporale delle azioni, con l'obiettivo di localizzare ogni attore nella scena e di classificare le sue azioni. Sfruttando il ruolo del contesto e la separazione tra la parte di *detection* e quella di *classification*, viene proposta una rappresentazione degli attori e degli oggetti nella scena basata su un grafo e una tecnica attentiva di apprendimento delle loro relazioni. Inoltre, l'apprendimento di queste relazioni può essere eseguito indipendentemente dal processo di estrazione delle feature spazio-temporali: questa osservazione ci ha permesso di ottenere un modello che può essere inserito in coda a qualsiasi rete convoluzionale. Il modulo proposto permette di arricchire le rappresentazioni delle entità sulla base di quelle circostanti, risultando in un incremento delle prestazioni per diverse reti e su diversi dataset.

Nel Capitolo 4 viene esplorata un'applicazione del riconoscimento video più orientata alla produzione, pur senza trascurare le esigenze della ricerca. Nello specifico, viene affrontata l'individuazione automatica di eventi salienti nelle partite di calcio, che rappresenta un primo passo verso la generazione automatica di highlights. L'individuazione di questi eventi richiede di localizzare il momento esatto in cui si verifica un'azione interessante e di classificarla. Ispirati da altri lavori presenti in letteratura, viene proposta una rete convolutiva addestrata a focalizzarsi sui frame che seguono l'evento stesso, e che contengono la maggior parte degli indizi visivi (ad esempio esultanze o replay sono comuni dopo un goal). Inoltre, il modello risultante è stato sottomesso nella challenge SoccerNet-v2 del ActivityNet Workshop di CVPR 2021, classificandosi terzo.

Nel Capitolo 5 viene sottolineato come i dataset pubblicamente disponibili solitamente ignorano le informazioni sensibili contenute nei loro video (come i volti degli attori). La ricerca risente in modo limitato di questo problema, dal momento che i dati vengono raccolti di proposito e con lo scopo di spingere la comunità scientifica a risolvere un determinato problema. In ambiente produttivo, in cui il servizio di riconoscimento delle azioni è solitamente gestito da un fornitore terzo, l'utente potrebbe non voler trasmettere il video originale, in modo da non compromettere la propria privacy. Viene quindi proposta un'architettura basata su due reti convoluzionali identiche, un teacher ed uno student, che sfrutta le informazioni sensibili solo durante l'apprendimento, mentre lavora su video anonimi (ad esempio con volti sfocati) durante la fase di inferenza, con una diminuzione trascurabile delle performance. Vengono analizzate due modalità di anonimizzazione delle informazioni personali, attraverso il mascheramento del

volto o dell'intero corpo degli attori coinvolti.

Come attività aggiuntiva nell'ambito di questa tesi viene studiata la complessità computazionale delle architetture di riconoscimento delle azioni (e dei modelli di analisi video in generale). Questi di solito richiedono enormi risorse, sia in fase di apprendimento che di inferenza, limitando l'accesso a quest'area di ricerca e aumentando i costi negli ambienti di produzione. Nel Capitolo 6 viene proposto un algoritmo iterativo in grado di ridurre la dimensione (in termini di numero di parametri e di operazioni in virgola mobile) dei modelli di comprensione video esistenti. L'idea di base consiste nella riduzione di una dimensione alla volta, dove ogni dimensione rappresenta un iper-parametro della rete ed ha un ruolo nella complessità del modello. Le iterazioni di riduzione vengono eseguite massimizzando l'accuratezza di ciascuno dei modelli ridotti, tramite un processo che permette di imitare il comportamento del modello originale, limitando gli effetti sulle performance.

Activities carried out during the Ph.D.

Besides the research activities described in this thesis, and those briefly summarized in Appendix A, I took part in several conferences and journals as a reviewer and as a presenter. A list of the main additional activities is reported below, while the complete list of my publications is reported in Appendix B.

Collaborations with companies and participation to programs

- Continuous collaboration with Metaliquid S.R.L. to meet the needs of both company and research
- Collaboration with Maticad S.R.L. for the development of state-of-the-art semantic segmentation models
- Participation in the NVIDIA AI Technology Center program
- Direct interaction with CINECA for the adoption of HPC systems

Conferences and journals reviewing

- ACM Transactions on Multimedia Computing, Communications, and Applications
- ACM International Conference on Multimedia

- International Conference on Pattern Recognition
- European Conference on Computer Vision Workshops

Conferences and schools attended

- IEEE Conference on Computer Vision and Pattern Recognition Workshop on Fair, Data Efficient and Trusted Computer Vision (Virtual), 2021
- NVIDIA GPU Technology Conference (Virtual), 2021
- 25TH International Conference on Pattern Recognition (Virtual), 2021
- 3rd Advanced Course on Data Science & Machine Learning (Siena, Italy), 2020
- 20th International Conference on Image Analysis and Processing (Trento, Italy), 2019
- International Computer Vision Summer School (Sicily, Italy), 2019
- International Workshop on Computer Vision (Modena, Italy), 2018

Master Thesis co-advising

- Davide Morelli, "Attentive Networks for video-level feature extraction and action recognition", 2020
- Lorenzo Silvio Del Rossi "Rep-Transformer: Attentive Networks for Repetition Counting and Boundary Detection in Videos", 2021

Awards and other academic services

- Third place in the SoccerNet-v2 challenge on the action spotting task, CVPR Workshops (Virtual), 2021
- Invited Presentation "RMS-Net: Regression and Masking for Soccer Event Spotting" for the SoccerNet-v2 challenge, in conjunction with the ActivityNet challenge, CVPR Workshops (Virtual), 2021

Seminars attended

- "Research in videogames: use of deep learning for saliency estimation and cheating prevention" - Dr. Iuri Frosio (NVIDIA) - June 30, 2021
- "About Time" - Prof. Arnold Smeulders (University of Amsterdam) - September 30, 2020
- "Brain-Inspired Computing: From Neuroscience to Artificial Intelligence" - October 11, 2019
- "Computational Aspects of Deep Reinforcement Learning" - Dr. Iuri Frosio (NVIDIA) - July 15, 2019
- "ICT Technology Commercialization and Business Development for Engineers" - Steven A. Gedeon - May 22-June 5, 2019
- "Deep Learning-based Automatic Speech Recognition" - Dr. Leonardo Badino - March 14, 2019
- "Computer Graphics for Cultural Heritage Applications" - Dr. Roberto Scopigno (CNR) - March 13, 2019
- "Academic English Workshop I" – Dott. Silvia Cavalieri – February 18-March 1, 2019
- "Academic English Workshop I" – Dott. Silvia Cavalieri – February 4-12, 2019

Chapter 2

Literature Survey

In this chapter, we provide an overview of the literature from which our research has been inspired. We first review the most important works related to deep learning-based spatio-temporal feature extraction (Sec. 2.1) and we present general alternatives devised for modeling the additional temporal information, together with related works tackling the problem of privacy in video modeling. We then discuss the specific task of action detection in video clips in Sec. 2.2 and the domain of sports and soccer (Sec. 2.3) together with proposed solutions for action spotting, representing a production application useful for broadcast service providers. Finally, we describe different ways to address the high computational demand of video models in Sec. 2.4.

2.1 Deep automatic video understanding

In this section, we provide an overview of the most successful deep models and algorithms devised for automatic video understanding. Specifically, we first describe general approaches for extracting spatio-temporal features, which is essential for solving video-related tasks. We analyze convolutional approaches, two-stream networks, and attentive alternatives. We discuss knowledge distillation for video modeling, as it inspired much of our research, and privacy-related problems that arise when handling videos containing sensitive information. We refer the reader to the survey from Zhu *et al*. [172], which provides a comprehensive timeline and a detailed description of the last efforts in action recognition.

Deep spatio-temporal backbones

Video understanding is one of the most challenging and broad areas of computer vision, and many research efforts have been dedicated to this field in the last few years. Convolutional neural networks are currently the state-of-the-art approach to extract spatio-temporal features for video processing and understanding. Video clips can be viewed as ordered sequences of images, which should be treated as a whole to extract temporal patterns. The most straightforward solution to adapt convolutions to videos consists in expanding 2D kernels to 3D ones in order to handle both time and appearance simultaneously, as proposed by Tran *et al*. [131] and Varol *et al*. [138]. However, such a solution suffers from the large number of parameters and floating point operations. To limit this drawback, other works [37, 110, 133] proposed to decompose 3D kernels in 2D spatial filters capturing frame appearance, followed by 1D temporal ones capturing motion.

Another strategy for separately modeling spatio-temporal features consists in the adoption of two different networks, one focusing on space and the other looking at the temporal evolution. Simonyan *et al*. [121] first proposed to give a single frame as input to the first network and the clip optical flow as input to the second one. Other works followed in order to increase the overall performance or efficiency [38, 145]. The clip's optical flow is usually adopted as input for the second stream, but it requires not negligible time resources to be computed. Among the most successful two-stream solutions, the one proposed by Carreira *et al*. [18] expands filters and pooling kernels of deep image classification CNNs into 3D, leveraging both successful ImageNet [28] architecture designs and their pre-trained parameters. Feichtenhofer *et al*. [36] introduced SlowFast networks, consisting of two pathways, a Slow path operating at low frame-rate which focuses on appearance, and a Fast one operating at higher frame-rate and employing fewer channels in convolutional layers, hence focusing on temporal modeling. The final predictions from the two independently trained networks are usually averaged at test time, but lateral connections can also be employed to fuse intermediate features during both training and inference.

Alternatives for time modeling

Besides adopting convolutional operators, other approaches devised more sophisticated techniques to equip a deep neural network with temporal reasoning. The temporal shift module [84], for instance, shifts a portion of the channels in a spatio-temporal volume along the temporal dimension, for exchanging informa-

tion between neighboring frames without the need for expensive 3D convolutions. Xie *et al*. [157] explored the best trade-off between 2D and 3D convolutions for high performance and efficiency, finding that temporal modeling is more useful at the bottom of a deep network. Hussein *et al*. [62] developed a multi-scale temporal convolution approach which uses different kernel sizes and dilation rates to better capture temporal dependencies, while Li *et al*. [80] employed 2D convolutional kernels sliding over the three 2D projections of a spatio-temporal volume. Recently, 4D kernels were introduced [168] by splitting a video clip containing a complete action instance into sections, each of which representing an action unit, and treating them as an additional axis on which convolution can be performed. This helps capturing inter-unit interactions, *i.e*. temporal interactions.

Attention-based networks

Recently, in order to overcome the limited receptive field issue of the convolution operator, new alternatives to fully convolutional networks arose, mainly based on attention operations. Non-local neural networks [147] introduced an attentive operator to update each pixel with information coming from all the other pixels of the input spatio-temporal volume. With the introduction of the Transformer architectures for both natural language processing [140] and vision [31], new architectures based on self-attention applied to input tokens have been proposed even for action recognition. These tokens usually correspond to square patches in vision tasks. However, the number of tokens in videos becomes prohibitive to handle from memory and computation perspectives, making the simple application of a Transformer-style encoder unfeasible. Several efficient alternatives have been proposed, by encoding frames separately before temporal feature extraction [6, 165], or using self-supervised pre-training [153]. Bertasius *et al*. [7] devised a convolution-free architecture, named TimeSformer, demonstrating the performance of separating spatial and temporal attention between 2D patches in a 3D input. Arnab *et al*. [3] proposed different ways of factorizing spatial and temporal dimensions of the input, exploring efficient solutions to handle the long sequences of tokens encountered in videos. Sharir *et al*. [118] exploited temporal attention to reduce the number of frames required for inference. A standard 2D convolutional backbone for extracting spatial information has been adopted by Neimark *et al*. [103], while aggregating temporal information with an attention-based encoder working on frame-level spatial features and attending to the entire video sequence.

Knowledge Distillation for video modeling

Knowledge Distillation (KD) has been first proposed in [56], and many alternatives followed [146]. The idea is to train a small network, the student, using the knowledge from a pre-trained bigger model, the teacher. While many approaches have been proposed for applying KD in image-related tasks [53, 71, 101, 162, 163], by transferring logits or intermediate features, the same does not hold for video-related tasks, where only a few methods have been introduced. A student looking only at a small fraction of the input frames has been proposed in [8]. Other works have exploited KD for multimodal action recognition [40, 41, 129], transferring knowledge between networks trained on different modalities. Other approaches have proposed to transfer knowledge indirectly, employing mutual relations between sample representations [39, 106, 130], in place of or together with the representations themselves. In this thesis, we propose two different strategies exploiting knowledge distillation for solving sensible data preservation and computational requirements reduction in video networks.

Privacy-preserving action recognition

In this thesis we also address the problem of privacy issues related to the adoption of videos disclosing people identities. Privacy issues are often ignored when building video pipelines. Most visual understanding models are trained to work on unaltered images and videos which could contain private information, such as people faces from which identities can be revealed. Although the preservation of sensible data is still not being tackled enough by the community, some approaches have been proposed in this direction. Yang *et al*. [159] studied the effect of image obfuscation on ImageNet [28], observing that the impact of privacy problems is quite limited in the case of image classification datasets. Scene understanding usually determines more severe privacy issues: efforts in removing pedestrians from street-view imagery [135] have been made, and action videos with anonymized faces have been collected [108]. Some technical approaches have been proposed for privacy-preserving action recognition. Wang *et al*. [149] adopted motion features between pairs of frames for masking identities, while Dai *et al*. [25] investigated the possibility of reducing camera resolution (thus making the actors unrecognizable) while keeping accuracy. An adversarial approach has been proposed in [113], consisting of a generator aiming to modify faces in videos and a discriminator trained to recognize modified faces, while action detection task is solved on anonymized faces.

2.2 Action detection

Solutions described so far have been mainly applied to general action recognition, with the goal of extracting meaningful spatio-temporal features to correctly classify a video clip with the label of the action performed in the clip itself. More specific tasks aim not only to classify the correct action, but also to localize it, both in space and time. Towards this goal, two problems have been analyzed in the last few years: temporal action detection and spatio-temporal action detection. The former task aims to segment the temporal interval in which the action takes place [14, 119]. Xu *et al*. [158] proposed a 3D CNN followed by a temporal proposal subnet for generating candidate activities of variable length, which are then filtered. Activity labels and refined segment boundaries are finally predicted. The boundary sensitive network [86] and the boundary matching network [85] predict starting and ending probabilities for each input frame, and build proposals between two starting and ending peaks, with different solutions to determine the proposal's confidence scores.

The second task is intended to detect people in space and time and to classify their actions [48, 65, 123, 124]. The role of context and interactions between entities becomes vital for the classification of actions performed by each actor in the scene. Seminal works in action detection and recognition have already investigated the role of context and that of modelling object interactions [33, 49, 50, 98, 109] to improve recognition. Recent approaches have tackled the spatio-temporal action detection task by exploiting human proposals coming from pre-trained image detectors [36, 151] and replicating boxes in time to build straight spatio-temporal tubes. Others have extended image detection architectures to infer more precise spatio-temporal tubelets. The approach from Gkioxari *et al*. [47] starts from image region proposals and selects those that are motion salient based on the magnitude of the optical flow, while Kalogeiton *et al*. [69] adopts an SSD detector [91] and proposes an algorithm for linking tubelets in time. Saha *et al*. [116] proposed an end-to-end 3D CNN to predict 'micro-tubes' spanning two successive frames, which are linked up into complete action tubes, and Hou *et al*. [57] devised Tube-of-Interest pooling for obtaining fixed 3D output size. Weakly-supervised approaches have also been explored [32, 99, 100]. Gu *et al*. [48] presented the new large scale AVA dataset and a baseline exploiting an I3D network encoding RGB and flow data separately, along with a Faster R-CNN [111] to jointly learn action proposals and labels. Ulutan *et al*. [136] suggested combining actor features with every spatio-temporal region in the scene to produce attention maps between the actor and the context. Girdhar *et al*. [46], instead, proposed a Transformer-style

architecture [140] to weight actors with features from the context around him. The ACAR-Net [104] computes second-order relations between every two actors based on their interaction with the context, *i.e*. based on first-order actor-context relations. Finally, graph-based representations have been used in action recognition [13, 64, 148, 169] to model spatio-temporal relationships, although the use of graph learning and graph convolutional neural networks [26, 73, 142] in video action detection is still under-investigated. Wang *et al*. [148] proposed to model a video clip as a combination of the whole clip features and weighted proposal features, computed by a graph convolutional network based on similarities in the feature space and spatio-temporal distances between detections. Zhang *et al*. [169], instead, defined the strength of a relation between two nodes in the graph as the inverse of the Euclidean distance between entities in the video.

2.3 Soccer action spotting

This thesis retraces a continuous collaboration with a company (*i.e*. Metaliquid S.R.L.), leading to solutions useful in production scenarios, and following the needs of both research and industry. Sports, and soccer in particular, represent a production-oriented domain in which all the previously described techniques could ideally find application. Soccer videos are used by professionals for statistics generation, for analyzing and developing strategies and for understanding failures. Broadcast video providers, on the other hand, often need to automatically generate summaries and highlights of soccer matches, currently still achieved via manual annotation in many cases. In this sections, we present some large-scale soccer datasets available in literature, and the recently defined soccer event spotting task. We also describe solutions tackling this task, and how they can contribute to the development of real multimedia applications.

Large-scale Soccer datasets

Gathering a large number of realistic soccer videos is not easy, because of the limited public availability of broadcast content. Nevertheless, a number of datasets for soccer analysis have been recently proposed. Yu *et al*. [161] collected 222 soccer matches with shot transitions, event boundaries, and players bounding box annotations. SoccerDB [144] includes 346 soccer games with event segments and players, ball, goalposts bounding box annotations. Pappalardo *et al*. [105] released an open collection of soccer-logs containing information about position, time,

outcome, player and other features, however without releasing any video. Recently Giancola *et al*. defined the action spotting task and released SoccerNet [44], a collection of 500 broadcast soccer games coming from the Big Five European leagues (EPL, La Liga, Ligue 1, Bundesliga and Serie A) with one second resolution event annotations. Three event classes are annotated in SoccerNet, namely Goal, Yellow/Red Card, and Substitution. SoccerNet-v2 [27] further extends the pool of available annotations, with 17 event classes, and defines two new tasks, namely camera shot segmentation and replay grounding.

Action spotting methods

The action spotting task has been proposed for soccer videos along with the SoccerNet dataset in [44], with the aim of finding the exact anchor time (or *spot*) of an event and to recognize it. The final objective of action spotting approaches is primarily automatic highlights identification and generation. With this purpose, Bag-of-Words and SIFT features have been adopted by [5], together with an LSTM for soccer video classification, while deep convolutional features have been used in [66]. Tsagkatakis *et al*. [134] adopted an optical flow and appearance feature fusion strategy for *goal* and *no goal* event detection. Giancola *et al*. [44] defined a first baseline based on a watershed method to compute segment proposals using the center time within the segment to define the spotting candidate. Several approaches tackling the action spotting task followed: audio stream integration has been explored in [137], showing promising spotting results. Rongved *et al*. [114] applied a 3D ResNet [52] directly to the video frames in a 5-seconds sliding window fashion. Vats *et al*. [141] introduced a multi-tower temporal convolutional network that accounts for the uncertainty of the action locations, while a context-aware loss function has been defined in [22], observing that frames *just after* an event contain most of the visual cues for event recognition.

2.4 Efficient video modeling

Recently, there has been a growing interest in efficient video processing [74]. Speed-accuracy trade-off was analyzed by Xie *et al*. [157], where early 3D convolutions were replaced by 2D ones in the network design. A policy network to decide per-frame input resolution has been proposed by Meng *et al*. [97], improving efficiency when handling less informative frames. Channel-separated convolutions for video classification have been explored by Tran *et al*. [132],

which limited the number of network parameters. Luo *et al*. [94] decomposes the feature channels into two separate groups handled in parallel, one for spatial modeling and the other for temporal modeling, which are then concatenated together. To capture temporal evolution without convolutions along the time axis, Temporal shift module [84] shifts channels along the temporal dimension and can be inserted into 2D CNN for temporal modeling without extra computation. X3D networks [35] progressively expand a 2D CNN through a form of coordinate descent in the space defined by some expansion axes, achieving impressive computation/accuracy trade-offs. Specifically, starting from a simple 2D CNN, the model size or the input size is expanded at each iteration, and the operation that gives the highest improvement in accuracy is kept. Bilen *et al*. [10] have learned a representation of video clips in the form of an image which summarizes the video dynamics in addition to appearance. A Multi-Fiber architecture that slices a complex neural network into an ensemble of lightweight networks have been proposed in [21], reducing the computational cost of 3D networks by an order of magnitude while increasing recognition performance. Wu *et al*. [154] observed that superfluous information can be reduced by up to two orders of magnitude by training a network directly on the compressed video. Wu *et al*. [152] varied mini-batch shapes and spatial-temporal resolutions according to a schedule, accelerating training by scaling up the mini-batch size and learning rate when shrinking the other dimensions. The importance of efficiency in video-related tasks is also highlighted by many other huge efforts in this direction [17, 30, 173].

Chapter 3

Spatio-temporal action detection

Understanding people actions in video clips is an open problem in computer vision, which has been addressed for more than twenty years [11, 54], given the wide range of its possible applications. In the past, this task was tackled through handcrafted features designed for specific actions [77, 143]. Recently, the video action detection task [126, 136, 160] (also referred to as spatio-temporal action localization) has been introduced, aiming to not only predict correct action labels from an input video, but also to detect each individual actor in subsequent frames and to recognize their (possibly multiple) actions. In this scenario, deep architectures able to extract fine-grained and discriminative spatio-temporal features are necessary, in order to represent video chunks in a compact and manageable form. This has motivated recent efforts to design novel backbones for video feature extraction [36, 133, 151]. On the other hand, higher-level reasoning is vital for detecting and understanding human actions.

Interestingly, the performances of video action detection networks that take inspiration from object detection architectures are still far from being satisfactory, especially when they only focus on the context included in the boxes around the actors themselves. For example, it would be difficult to recognize whether a person is *watching* something just by looking at a bounding box around him,

This chapter is related to publication "M. Tomei et al, Video action detection by learning graph-based spatio-temporal interactions, CVIU 2021". See Appendix B for details.

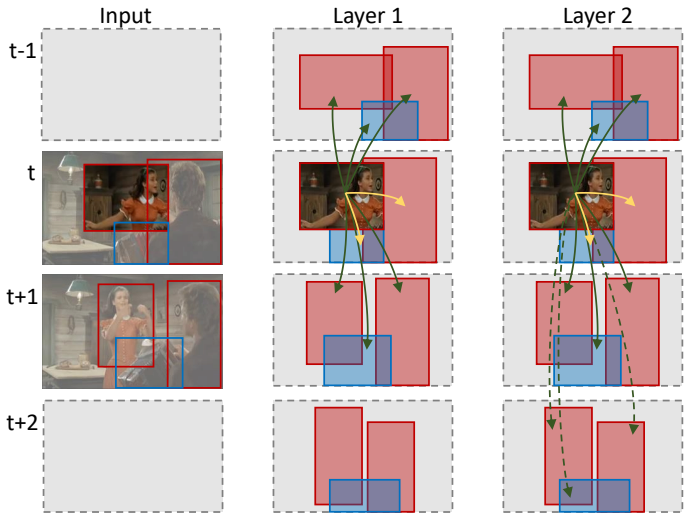


Figure 3.1: We propose a graph-based module for video action detection which encodes relationships between actors and objects in a spatio-temporal neighborhood. Multiple layers of the module generate indirect edges between temporally distant entities, increasing the temporal receptive field.

without considering a wider context. This can be partly explained by the lack of proper context understanding of the previous works, as they cannot model the relationships between actors and surrounding elements [136]. Indeed, the presence of objects and other people in the scene, together with their behaviors, influences the understanding of the actor at hand. High-level reasoning is necessary not only at the spatial level, to model relations between close entities, but also in time: most of the existing backbones can handle small temporal variations, without modeling long-term temporal relationships.

Following these premises, in this chapter we propose a high-level module for video action detection which considers interactions between different people in the scene and interactions between actors and objects, represented by a graph structure. Further, it can take into account temporal dependencies by connecting consecutive clips during learning and inference. The same module can be stacked multiple times to form a multi-layer structure (Fig. 3.1). In this way, the overall temporal receptive field can be arbitrarily increased to model long-range dependencies.

Since our method works at the feature level, it can expand its temporal receptive field without dramatically increasing computation. The proposed solution can exploit existing backbones for feature extraction and can achieve state-of-the-art results without an end-to-end finetuning of the backbone.

Contributions. Previous works in action analysis have already tried to exploit graph-based representations [148, 169], to model relationships with the context [46, 136] and to exploit long-term temporal relations [151]: in this chapter, we propose to merge all these insights in a single module, which is independent of the feature extraction layers and works on pre-computed representations. To the best of our knowledge, our model is among the first to employ a learning-based approach also on graph edges. The main novelty of this work consists in the design of a single graph spanning both space and time, where connections between nodes are both learned and affected by entities' proximity. We experimentally validate our approach on different publicly available benchmark datasets, namely the Atomic Visual Actions (AVA) dataset [48], which represents a challenging test-bed for recognizing human actions and exploiting the role of context, J-HMDB-21 [65] and UCF101-24 [124]. We demonstrate that our approach increases the performance of three different video backbones, reaching state-of-the-art results.

The rest of the chapter is organized as follows: in Sec. 3.1 we introduce the details and the architecture of the proposed action detection model. Adopted datasets, metrics and backbones' setup together with implementation and training details are presented in Sec. 3.2. Finally experiments, quantitative results, and qualitative evaluation are reported in Sec. 3.3. Our code is publicly available at `https://github.com/aimagelab/STAGE_action_detection`.

3.1 Graph-based interaction learning

Given a video clip, the goal of our approach is to localize each actor and classify his actions. As actions performed in a clip depend on actor and object relationships through both space and time, we define a graph representation in which actor and object detections are treated as nodes, and edges hold relationships between them. Further, we link graphs from subsequent clips in time, to encode relations between clips belonging to the same longer video. We name our approach STAGE, as an acronym of *Spatio-Temporal Attention on Graph Entities*.

In this section, we first outline our graph-based representation for a single clip, describing the graph attention layer and our adjacency matrix. In the remainder of the section, we will then extend this approach to handle sequences of clips.

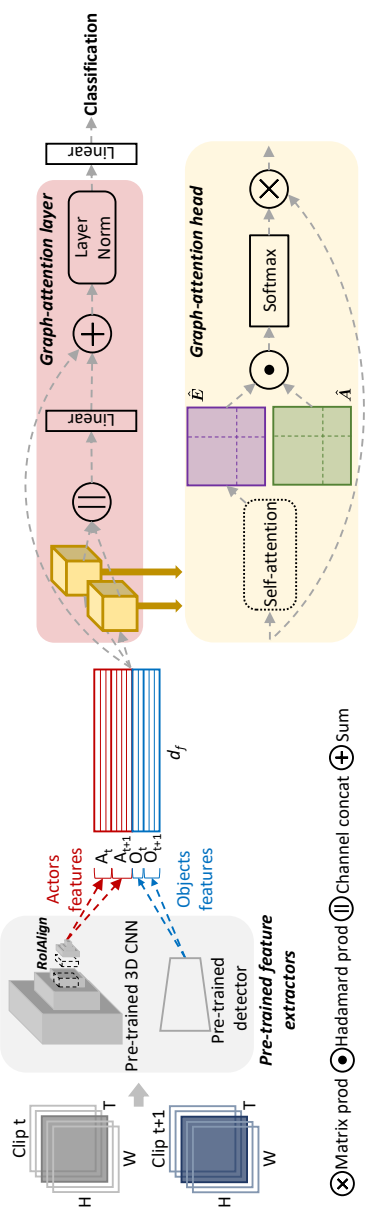


Figure 3.2: Architecture of our high-level video understanding module. Given consecutive clips, the input of our model consists of actor and object features. It then applies a sequence of graph-attention layers (red-background box), each of them composed of a number of graph-attention heads (yellow-background box) applied in parallel. The figure depicts a single layer with two heads.

3.1.1 Graph-based clip representation

We propose a graph-based representation of each clip, where nodes consist of actor and object features predicted by pre-trained detectors, as shown in the left side of Fig. 3.2. Denoting the number of actors and objects belonging to clip t (*i.e.* the clip centered in frame t) as A_t and O_t respectively, the total number of graph entities is $N_t = A_t + O_t$. Under this configuration, a clip can be represented as an $N_t \times d_f$ matrix, where d_f is the node feature size.

Since actors can have meaningful relations both between them and with objects in the scene, we employ a fully-connected graph representation, in which all nodes are connected to the others, as the input of our network. Following the assumption that the closer two entities are, the more they influence each other, a link between two entities in the graph is made stronger if they are spatially close. The graph configuration is therefore given by a dense $N_t \times N_t$ adjacency matrix $\boldsymbol{A}$, in which $\boldsymbol{A}_{ij}$ is defined as the proximity between entities i and j, computed as follows:

$$\boldsymbol{A}_{ij} = e^{-\sqrt{(x_{ci}-x_{cj})^2+(y_{ci}-y_{cj})^2}}, \tag{3.1}$$

where x_{ci} and y_{ci} are the center coordinates of entity i.

In Sec. 3.1.3, this single-clip adjacency matrix representation will be extended to a multi-clip adjacency matrix, allowing us to easily link graphs coming from subsequent clips of the same video.

3.1.2 Spatial-aware graph attention

Besides the introduced adjacency matrix, we adopt a graph attention module, inspired by [142]. The input of the model consists of N_t node features which represent actors and objects, $\{\boldsymbol{f}_1, \boldsymbol{f}_2, ..., \boldsymbol{f}_{N_t}\}$ with $\boldsymbol{f}_i \in \mathbb{R}^{d_f}$. First, the module applies a linear transformation to these features, in order to obtain a new representation of each entity $\{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_{N_t}\}$, $\boldsymbol{h}_i \in \mathbb{R}^{d_h}$. Then a *self-attention* operator $\mathcal{S}$ is applied to the nodes. In particular, the operator is defined as $\mathcal{S} : \mathbb{R}^{d_h} \times \mathbb{R}^{d_h} \rightarrow \mathbb{R}$, as follows:

$$\boldsymbol{E}_{ij} = \mathcal{S}(\boldsymbol{h}_i, \boldsymbol{h}_j) \tag{3.2}$$

with the scalar $\boldsymbol{E}_{ij}$ representing the *importance* of entity j with respect to entity i. Since we propose to represent a clip as a fully-connected graph, $\boldsymbol{E}_{ij}$ is computed for each pair of entities belonging to the same clip, avoiding the need for masking disconnected couples. Based on the original graph attention implementation [142],

$\mathcal{S}$ is implemented with a feedforward layer with $2 \times d_h$ parameters, followed by a LeakyReLU nonlinearity:

$$\boldsymbol{E}_{ij} = \text{LeakyReLU}(\mathsf{FC}(\boldsymbol{h}_i \parallel \boldsymbol{h}_j)), \tag{3.3}$$

where $\parallel$ indicates concatenation on the channel axis and FC is a linear layer. The resulting matrix, $\boldsymbol{E}$, will be a squared matrix with the same shape as the adjacency matrix. Separating it into its components, it can be rewritten as:

$$\boldsymbol{E} = \begin{pmatrix} \boldsymbol{E}_{aa} & \boldsymbol{E}_{ao} \\ \boldsymbol{E}_{oa} & \boldsymbol{E}_{oo} \end{pmatrix} \tag{3.4}$$

where $\boldsymbol{E}_{aa}$ is the matrix of attentive weights between actors and actors, $\boldsymbol{E}_{ao}$ is the matrix of objects weights to actors, $\boldsymbol{E}_{oa}$ is the matrix of actors weights to objects and $\boldsymbol{E}_{oo}$ is the matrix of objects weights to objects.

Introducing spatial proximity. The proposed self-attention module, when applied to a clip graph, computes the mutual influence of two entities in feature space, *i.e*. the influence of an entity on another based on their features. However, it does not consider mutual distances between entities.

To introduce the prior given by the spatial proximity inside the clip, we condition the self-attention matrix $\boldsymbol{E}$ with the adjacency matrix $\boldsymbol{A}$, which contains the proximity between detections, by taking their Hadamard product, *i.e*.:

$$\boldsymbol{D} = \boldsymbol{A} \odot \boldsymbol{E}. \tag{3.5}$$

This operation allows us to strengthen the *importance* of the features of an entity with respect to its neighbors and to weaken relations between entities that lie spatially far from each other. A row-wise softmax normalization is then applied to obtain an importance distribution over entities:

$$\boldsymbol{W}_{ij} = \frac{\exp(\boldsymbol{D}_{ij})}{\sum_{k=1}^{N_t} \exp(\boldsymbol{D}_{ik})}. \tag{3.6}$$

The updated features computed by the module are a linear combination of the starting features $\{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_{N_t}\}$ using $\{\boldsymbol{W}\}_{i,j}$ as coefficients. In particular, the self-attention module updates the initial features as follows:

$$\boldsymbol{h}'_i = \sigma\left(\sum_{j=1}^{N_t} \boldsymbol{W}_{ij} \boldsymbol{h}_j\right), \tag{3.7}$$

where σ is an ELU nonlinearity [23].

3.1.3 Temporal graph attention

In this section, we extend the proposed attention-based approach to jointly encode a batch of consecutive clips. Since different clips can have a different number of actors and objects, we devise a single adjacency matrix with as many rows and columns as the total number of entities in all clips of the batch. Besides allowing us to manage clips with a variable number of entities, this solution is suitable to link more graphs together and avoids padding. When encoding a batch of consecutive clips, the size of the adjacency matrix will be $\sum_{t=1}^{b} N_t \times \sum_{t=1}^{b} N_t$, being b the size of the batch of clips.

An example is shown in Fig. 3.3, for a three clips setting and a temporal receptive field of three consecutive clips. Here, dark red elements contain the proximity between actors of the same clip, dark blue elements contain the proximity between objects of the same clip, and dark violet elements contain the proximity between actors and objects of the same clip. Entities belonging to subsequent clips can be linked by computing their boxes proximity (as in Eq. 3.1), assuming that the temporal distance between clips is small enough to ensure the consistency of the scene. The light-colored elements of the adjacency matrix in Fig. 3.3 contain the proximity between actors (light red), objects (light blue), and actors/objects (light violet) belonging to two consecutive clips. The temporal receptive field of a single attentive layer can be potentially increased by adding the proximity of temporally distant entities in the adjacency matrix.

Self-attention over time. We extend the self-attentive operations to compute the *importance* of an entity with respect to all the other entities in the batch. In our implementation, the *self-attention* module computes attention weights for each pair of entity features, without any masking. For a three clips per batch setting, the complete attention weights matrix $\hat{\boldsymbol{E}}$ looks like the following:

$$\hat{\boldsymbol{E}} = \left(\begin{array}{ccc|ccc} \hat{\boldsymbol{E}}_{aa}^{t_0,t_0} & \hat{\boldsymbol{E}}_{aa}^{t_0,t_1} & \hat{\boldsymbol{E}}_{aa}^{t_0,t_2} & \hat{\boldsymbol{E}}_{ao}^{t_0,t_0} & \hat{\boldsymbol{E}}_{ao}^{t_0,t_1} & \hat{\boldsymbol{E}}_{ao}^{t_0,t_2} \\ \hat{\boldsymbol{E}}_{aa}^{t_1,t_0} & \hat{\boldsymbol{E}}_{aa}^{t_1,t_1} & \hat{\boldsymbol{E}}_{aa}^{t_1,t_2} & \hat{\boldsymbol{E}}_{ao}^{t_1,t_0} & \hat{\boldsymbol{E}}_{ao}^{t_1,t_1} & \hat{\boldsymbol{E}}_{ao}^{t_1,t_2} \\ \hat{\boldsymbol{E}}_{aa}^{t_2,t_0} & \hat{\boldsymbol{E}}_{aa}^{t_2,t_1} & \hat{\boldsymbol{E}}_{aa}^{t_2,t_2} & \hat{\boldsymbol{E}}_{ao}^{t_2,t_0} & \hat{\boldsymbol{E}}_{ao}^{t_2,t_1} & \hat{\boldsymbol{E}}_{ao}^{t_2,t_2} \\ \hline \hat{\boldsymbol{E}}_{oa}^{t_0,t_0} & \hat{\boldsymbol{E}}_{oa}^{t_0,t_1} & \hat{\boldsymbol{E}}_{oa}^{t_0,t_2} & \hat{\boldsymbol{E}}_{oo}^{t_0,t_0} & \hat{\boldsymbol{E}}_{oo}^{t_0,t_1} & \hat{\boldsymbol{E}}_{oo}^{t_0,t_2} \\ \hat{\boldsymbol{E}}_{oa}^{t_1,t_0} & \hat{\boldsymbol{E}}_{oa}^{t_1,t_1} & \hat{\boldsymbol{E}}_{oa}^{t_1,t_2} & \hat{\boldsymbol{E}}_{oo}^{t_1,t_0} & \hat{\boldsymbol{E}}_{oo}^{t_1,t_1} & \hat{\boldsymbol{E}}_{oo}^{t_1,t_2} \\ \hat{\boldsymbol{E}}_{oa}^{t_2,t_0} & \hat{\boldsymbol{E}}_{oa}^{t_2,t_1} & \hat{\boldsymbol{E}}_{oa}^{t_2,t_2} & \hat{\boldsymbol{E}}_{oo}^{t_2,t_0} & \hat{\boldsymbol{E}}_{oo}^{t_2,t_1} & \hat{\boldsymbol{E}}_{oo}^{t_2,t_2} \end{array}\right) \tag{3.8}$$

where $\hat{\boldsymbol{E}}_{aa}^{t,t'}$ is the weights matrix of actors belonging to clip t to actors belonging to clip t', $\hat{\boldsymbol{E}}_{ao}^{t,t'}$ is the weights matrix of actors belonging to clip t to objects belonging to clip t', and so on.

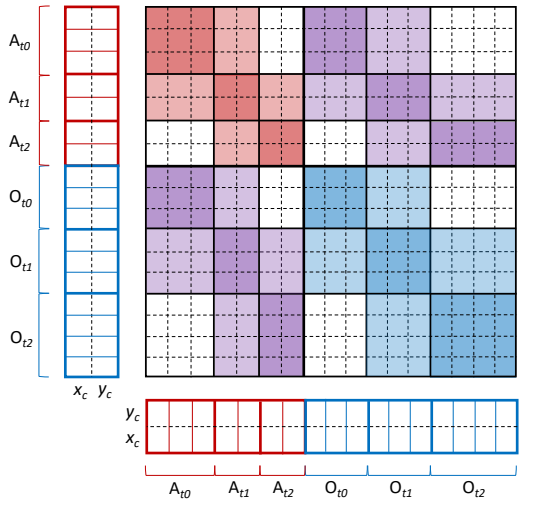


Figure 3.3: Adjacency matrix in a three clips per batch configuration, containing the spatial proximity between entities belonging to the same clip (dark-colored sub-matrices) and to consecutive clips (light-colored sub-matrices). White elements are zeros. Bounding box centers are indicated as x_c and y_c.

Given the new adjacency matrix, $\hat{\boldsymbol{A}}$, the Hadamard product:

$$\hat{\boldsymbol{D}} = \hat{\boldsymbol{A}} \odot \hat{\boldsymbol{E}} \tag{3.9}$$

is in charge of strengthening or weakening (based on the spatial distance) weights between entities belonging to the same timestamp or sufficiently close in time, and to zero weights between temporally distant entities. Finally, the linear combination of Eq. 3.7 replaces features of an entity with a weighted sum of features directly connected to it in the graph: these features come now from entities belonging to the same clip and to temporally close clips. As our approach works at the feature level, increasing the temporal receptive field of the module does not dramatically increase the demand of computational resources.

Multi-head multi-layer approach

A single graph-attention head (the box with yellow background of Fig. 3.2) performs the aforementioned operations in our model. A graph-attention layer (red-

background box of Fig. 3.2) concatenates the output of different heads and applies a linear layer, a residual connection, and a layer normalization [4]. As it will be analyzed in Sec. 3.3.2, a grid search on the number of parallel heads and subsequent layers allows us to obtain the best performance.

It is worth noting that the overall temporal receptive field of the model is also affected by the number of adopted graph attention layers. Considering a receptive field of three (corresponding to a graph where entities are directly connected only with other entities of the same clip and to entities from the previous and following clips), each layer after the first one increases the overall temporal receptive field by two. In a two layers setting, for instance, the second graph attention layer will compute the features of a clip as a weighted sum of its two temporal neighbors, but features from those have already been affected by features of other clips in the first graph attention layer.

3.2 Datasets, metrics and implementation details

In this section, we introduce the datasets and the evaluation metrics for the spatio-temporal action localization task. Moreover, we detail the implementation of the feature extraction backbones and the optimization parameters.

3.2.1 Datasets and metrics

We evaluate our model on versions 2.1 and 2.2 of the challenging AVA dataset [48], which contains annotations to localize people both in space and time and to predict their actions. It consists of 235 training and 64 validation videos, each 15 minutes long. The temporal granularity of annotations is 1 second, leading to 211k training and 57k validation clips centered in the annotated keyframes. Each actor is involved in one or more of 80 atomic action classes. One of the main challenges of AVA concerns its long-tail property: tens of thousands of samples are available for some classes while only a few dozen for others.

The performance of a model on AVA is measured by the keyframe-level mean Average Precision (mAP) with a 50% IoU threshold. Following the protocol suggested by the dataset authors and adopted in prior works, we train our architecture on all the 80 classes and evaluate its performance only on the 60 classes containing at least 25 validation examples.

We also report performances in terms of frame-mAP with the same 50% IoU threshold on two additional benchmarks, namely J-HMDB-21 [65] and UCF101-

24 [124]. These two datasets are relatively smaller than AVA, and provide a single label per video. On average, they also come with less complex scenes and fewer interactions between entities.

3.2.2 Network details

People and object detectors. When experimenting on AVA, we use a Faster R-CNN [111] with a ResNeXt-101-FPN [52, 87, 156] as people detector, applied on keyframes. The detection network is pre-trained on COCO [88] and fine-tuned on AVA [48] people boxes. The detector is the same used by [36] and [151], and reaches 93.9 AP@50 on the AVA validation set. Following previous works [36, 48], actor features are then obtained from a 3D CNN backbone (which is discussed in the next paragraph) by replicating boxes in time to obtain a 3D region of interest and applying RoIAlign [51]. During training, we employ ground-truth people regions. Objects features, instead, are extracted from a Faster R-CNN detector pre-trained on Visual Genome [75], and have a dimensionality of 2048.

Video Backbones. In all the experiments, we employ a pre-trained actor backbone which is kept fixed during training. Freezing the backbone allows us to increase the batch size and to explore longer temporal relations between consecutive clips. The pre-trained backbones take raw clips as input and output features for each actor. When considering the AVA dataset, all video-level backbones are trained on the Kinetics dataset [70] and fine-tuned on the AVA dataset [48] before applying our module.

On AVA, we consider three backbones for extracting actor features, namely I3D [18], R101-I3D-NL [151] and SlowFast-NL [36]. The I3D [18] backbone is pre-trained on ImageNet [28] before being "inflated", and then trained on Kinetics-400. RoIAlign is applied after the *Mixed_4f* layer and we fine-tune only the last layers (from the *Mixed_5a* layer to the final linear classifier) on AVA for 10 epochs. The R101-I3D-NL [151] and the SlowFast-NL [36] backbones are pre-trained on Kinetics-400 or Kinetics-600 (R101-I3D-NL on ImageNet, too), and fine-tuned end-to-end on the AVA dataset. For these backbones, we employ the original pre-trained weights released by the authors in their official repositories[1,2].

For the I3D backbone, we use ground-truth boxes and predicted boxes with any score during feature extraction, assigning labels of a ground-truth box to a predicted box if their IoU is 0.5 or more. We use predicted boxes with score

[1] https://github.com/facebookresearch/video-long-term-feature-banks
[2] https://github.com/facebookresearch/SlowFast

at least 0.7 for the evaluation. Following the authors implementation [36, 151], for the R101-I3D-NL and SlowFast-NL backbones we use ground-truth boxes and predicted boxes with score at least 0.9 during feature extraction, assigning labels of a ground-truth box to a predicted box if their IoU is 0.9 or more. We use predicted boxes with score at least 0.8 for the evaluation.

Features are always extracted from the last layer of the backbone before classification, after averaging in space and time dimensions: feature size, therefore, is 1024, 2048, and 2304 for I3D, R101-I3D-NL, and SlowFast-NL respectively. As additional features, we also add bounding boxes height, width and center coordinates to actors and objects, as we found it to be beneficial in preliminary experiments. A linear layer is employed to transform actor or object features to a common dimensionality d_f, making their concatenation feasible. In all experiments, when concatenating actor and object features we apply the linear layer to the feature vector with the largest dimensionality, leaving the other unchanged.

Implementation and training details

Each graph attention head consists of two fully-connected layers. The first one reduces the feature size depending on the number of heads used in that layer: with n_h heads, the output feature size is set to $\lfloor d_f/n_h \rfloor$. The second linear layer, instead, computes attention weights (Eq. 3.3). The outputs of different attention heads are then concatenated, and a fully connected layer followed by a residual block and a layer normalization block is applied (Fig. 3.2). Each graph attention head is followed by a dropout with keep probability 0.5, and the alpha parameter of the LeakyReLU in Eq. 3.3 is set to 0.2. After a sequence of layers of the proposed module, one last linear layer is employed to compute per-class probabilities, and a sigmoid cross-entropy loss (for AVA) or a softmax cross-entropy loss (for the other benchmarks) is applied. In our experiments, we adopt a temporal receptive field of three, connecting entities of a clip with those belonging to the same, the previous and the following clip. Table 3.1 lists the learnable blocks of our architecture in a 4-heads/1-layer setting. It is worthwhile to mention that our graph-attention block is trained without any data augmentation, while end-to-end approaches typically require random flipping, scaling, and cropping.

During training, we use a batch size of 6. Adam optimizer [72] is adopted in all our experiments, with a learning rate of 6.25×10^{-5} when using I3D features and 10^{-5} for R101-I3D-NL and SlowFast-NL features. The learning rate is decreased by a factor of 10 when the validation mAP does not increase for ten consecutive epochs. Early-stopping is also applied when the validation mAP does

Stage	Module	Input size	Output size
$Input$		$N_t \times 1024$	
GAL_1	FC_{11}	$[N_t \times 1024] \times 4$	$[N_t \times 256] \times 4$
	FC_{12}	$[N_t \times N_t \times 512] \times 4$	$[N_t \times N_t] \times 4$
	FC_{13}	$N_t \times 1024$	$N_t \times 1024$
	LNorm	$N_t \times 1024$	$N_t \times 1024$
	FC_3	$A_t \times 1024$	$A_t \times classes$

Table 3.1: Learnable blocks of STAGE, in a 4-heads/1-layer configuration, when using I3D features. GAL_i indicates the i-th graph-attention-layer, which consists of 4 attention heads (each with 2 fully-connected layers), a linear layer, and a LayerNorm. N_t is the number of entities (actors and objects), A_t is the number of actors.

not increase for five consecutive epochs. All the experiments are performed on a single NVIDIA V100 GPU; on average, a single experiment takes less than a day to converge.

3.3 Experimental results

In the following experiments, we show that our proposed module can improve video action detection performance by a significant margin, reaching state-of-the-art results. If not specified otherwise, we employ a 2-layers 4-heads setting when using the I3D backbone and when testing on J-HMDB-21 and UCF101-24, and a 2-layers 2-heads setting when using the R101-I3D-NL and the SlowFast-NL backbones.

3.3.1 Main results

Here we report the results obtained on three different benchmarks, namely AVA (for both versions 2.1 and 2.2), J-HMDB-21, and UCF101-24. The first one represents the most challenging dataset, with complex scenes from broadcast videos and with the highest number of classes. For this reason, we also conduct a per-class analysis over the AVA dataset.

Model	Pretraining	mAP@50
AVA [48]	Kinetics-400	15.6
ACRN [126]	Kinetics-400	17.4
STEP [160]	Kinetics-400	18.6
Better baseline [45]	Kinetics-600	21.9
SMAD [169]	Kinetics-400	22.2
RTPR [82]	-	22.3
ACAM [136]	Kinetics-400	24.4
VATX [46]	Kinetics-400	24.9
SlowFast [36]	Kinetics-400	26.3
LFB (R101-I3D-NL) [151]	Kinetics-400	26.8
I3D [18]	Kinetics-400	19.7
STAGE (I3D)	Kinetics-400	**23.0**
R101-I3D-NL [151]	Kinetics-400	23.9
STAGE (R101-I3D-NL)	Kinetics-400	**26.3**
SlowFast-NL,8×8 [36]	Kinetics-600	28.2
STAGE (SlowFast-NL,8×8)	Kinetics-600	**29.8**

Table 3.2: Comparison with previous approaches on AVA 2.1 validation set, in terms of mean Average Precision.

Results on AVA 2.1

Table 3.2 shows the mean Average Precision with 50% IoU threshold on AVA 2.1 for our method, considering the three backbones, and for a number of competitors. All the experiments refer to a single-crop validation accuracy (no multi-scale and horizontal flipping are adopted for testing) and for a single model (*i.e.*, without using ensemble methods).

When applying our approach, we observe a relative improvement of more than 16% on the I3D backbone (19.7 $\rightarrow$ 23.0), of about 10% for the R101-I3D-NL backbone (23.9 $\rightarrow$ 26.3) and of almost 6% for the SlowFast-NL backbone (28.2 $\rightarrow$ 29.8). Noticeably, the presence of non-local operations [147] in these last two backbones does not prevent our model from improving performance, underlying that these two techniques are complementary. Also, the results obtained using the I3D backbone are superior to many approaches that employ the same backbone and train end-to-end. The adoption of a long-term feature bank in [151] brings slightly better performance (26.8) compared to our solution (26.3) using the same R101-I3D-NL backbone. It is worth noting, although, that [151] uses two instances of the backbone, one to compute long-term features and another to compute short-term features, both fine-tuned end-to-end. Our model, instead, uses only one

Model	Pretraining	val mAP	test mAP
SlowFast-NL,8×8 [36]*	Kinetics-600	29.1	-
STAGE (SlowFast-NL,8×8)	Kinetics-600	**30.0**	**29.6**
SlowFast-NL,16×8 [36]*	Kinetics-600	29.4	-
STAGE (SlowFast-NL,16×8)	Kinetics-600	**30.3**	**29.9**
STAGE (SlowFast-NL,16×8,8×8)	Kinetics-600	**31.8**	**31.6**

Table 3.3: Comparison with previous approaches on AVA 2.2 validation and test sets, using different backbones. Numbers marked with "*" are obtained from models released in the official SlowFast [36] repository. In [36], the reported performances are 29.0 and 29.8 for SlowFast-NL,8x8 and SlowFast-NL,16x8 respectively.

Model	Pose	Person-Person	Person-Object
I3D [18]	37.4	20.4	12.2
STAGE (I3D)	**40.4**	**23.5**	**15.7**
R101-I3D-NL [151]	41.4	26.5	15.5
STAGE (R101-I3D-NL)	**43.4**	**29.0**	**18.1**
SlowFast-NL [36]	48.3	28.8	19.8
STAGE (SlowFast-NL)	**50.3**	**31.4**	**20.8**

Table 3.4: Mean Average Precision on different AVA 2.1 class groups (actions involving person-pose, person-person and person-object interactions), when using the I3D backbone.

backbone instance, which is also kept fixed during training. Single-crop validation mAP obtained with the SlowFast-NL backbone (**29.8 mAP**) represents a new state of the art for the AVA v2.1 dataset.

Results on AVA 2.2

Table 3.3 reports the performance of our approach on the more recent AVA v2.2. Here, the test mAP has been computed using the official AVA evaluation server, after training on both train and val splits, following the common practice in literature. As it can be observed, adopting STAGE on top of SlowFast-NL,8x8 is better than doubling the number of input frames to the backbone (*i.e.*, using SlowFast-NL,16x8). This underlines that modelling high-level entities is at least

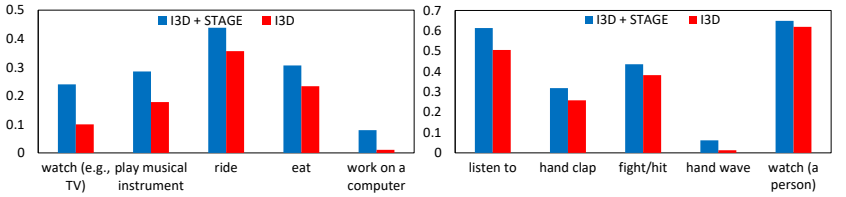


Figure 3.4: Per-class Average Precision of an I3D backbone with and without our module. We report the five classes with the highest absolute gain among person-object interaction classes (top) and person-person interaction classes (bottom).

as important as extracting better spatio-temporal features.

Finally, leveraging the fact that STAGE is backbone independent, we train it using both SlowFast-NL,8x8 and SlowFast-NL,16x8 features, which are averaged before being forwarded through STAGE. This model achieves single-crop **31.8 mAP**, a new state of the art for AVA v2.2.

Per-class analysis. In Table 3.4, we show the performances of our approach on different AVA class groups, *i.e*. actions involving person-pose (13 classes), person-person interactions (15 classes) and person-object interactions (32 classes). As it can be seen, our model shows a higher improvement for actions that involve an interaction between entities (which are also the majority in AVA). We also note that the recognition of pose classes (like *dance* or *martial art*) benefits from interactions and elements from the context. Finally, in Fig. 3.4, we show the five person-object interaction classes (left) and five person-person interaction classes (right) with the top AP gain, when considering the I3D backbone. Classes with the highest absolute gain are *watch (e.g., TV)* (+14.0 AP), *listen to (a person)* (+10.7 AP), *play musical instrument* (+ 10.7 AP), all involving interactions with other objects or actors.

Results on J-HMDB-21 and UCF101-24

We also experimented the capabilities of STAGE on two additional benchmarks, namely J-HMDB-21 [65] and UCF101-24 [124], in comparison with the models presented in [47] and in [116]. For fairness of evaluation, we adopt STAGE on top of the actor detection backbones presented in [47] and in [116]. Hence, only actor boxes with no objects are considered. J-HMDB-21 mAP is averaged over the three splits, while UCF101-24 mAP refers to the first split of the dataset, following the

Model↓ Dataset→	J-HMDB-21	UCF101-24
Action tubes [47]	27.0	-
Action tubes* [47]	28.6	-
STAGE (Action tubes*)	**29.6**	-
AMTnet [116]	45.0	-
AMTnet* [116]	47.0	67.7
STAGE (AMTnet*)	**48.1**	**69.1**

Table 3.5: Experimental results on J-HMDB-21 and UCF101-24, using the detection backbones presented in [47] and [116]. For both backbones, the three rows report the mAP presented in the original paper, the mAP we obtained training a linear classifier on top of pre-extracted features (marked by *) and the mAP of STAGE on the same features, respectively.

standard practice in literature. Results are reported in Table 3.5, where, for each dataset, we also report the frame-mAP obtained training a linear classifier on top of pre-extracted features (marked with *), and the frame-mAP obtained through STAGE applied on the same features. Only RGB features are considered, without exploiting optical flow. As it can be seen, in both settings applying STAGE leads to a performance improvement of around 1 mAP point.

3.3.2 Ablation study

To validate the importance of the design choices made in our graph-attention module, we run several ablation experiments. We explore different combinations of heads and layers, we remove key components in the architecture, and we modify the graph structure to use existing techniques in place of our choices. In all the following experiments we employ AVA v2.1.

Varying the number of heads and layers. Table 3.6 shows the effect of varying the number of graph-attention heads and layers when using STAGE with features from the three adopted backbones. As it can be noticed, stacking multiple layers together brings better performance: each layer after the first increases the temporal receptive field, generating indirect edges between temporally distant nodes. The best configuration is obtained when using a 4-heads/2-layers setting for I3D features and a 2-heads/2-layers setting for R101-I3D-NL and SlowFast-NL. These configurations are also used in the following ablation experiments.

Comparison with the STO baseline. In Table 3.7 we compare the STAGE and

Layers→	1			2			3		
Heads→	2	4	8	2	4	8	2	4	8
STAGE (I3D)	21.2	21.7	22.0	21.7	**23.0**	22.8	21.7	22.7	21.9
STAGE (R101-I3D-NL)	26.2	26.1	**26.3**	**26.3**	26.1	25.8	**26.3**	26.0	25.6
STAGE (SlowFast-NL)	29.7	29.2	29.6	**29.8**	29.3	29.6	29.6	29.3	29.5

Table 3.6: Validation mAP obtained considering different combinations of graph-attention heads and layers.

Head↓ Backbone→	I3D	R101-I3D-NL	SlowFast-NL
1L STO	20.2	24.7	28.5
2L STO	20.4	25.0	28.7
STO (STAGE)	20.5	25.5	29.5
STAGE	**23.0**	**26.3**	**29.8**

Table 3.7: Comparison with the STO baseline.

the STO modules [151], applied on top of the considered backbones. The STO baseline consists of a short-term operator that updates actor features on the basis of other actors from the same clip, using non-local blocks [147]. STO lacks the graph structure, the objects, and the temporal interactions, leading to worse performances compared to STAGE (Table 3.7 first two rows, corresponding to one-layer and two-layers STO, respectively). In the third row of Table 3.7, instead, we report the performance of STAGE when replacing Eq. 3.3 with the non-local-based attention, and removing temporal interactions. The original STAGE design demonstrates higher mAP in this case, too.

Removing key components. In Table 3.8 we report the validation mAP obtained when removing key components. Performances drop when removing the spatial prior between detections. Moreover, when the temporal links between consecutive clips are removed and only edges between nodes of the same clip are kept, we observe a reduction in mAP. To evaluate the design of the graph structure, we first remove actor-actor interactions to quantify the role of objects: in this setting, actor features are updated with a weighted sum of object features, leading to better performances compared to those of graph-free backbones. One can question if object-object interactions are useful: when removing them from the graph, performances drop for both I3D and R101-I3D-NL backbones. Our insight is that some recurring combinations of objects can be useful for prediction: a closed door

Model↓ Backbone→	I3D	R101-I3D-NL	SlowFast-NL
Original Backbone	19.7	23.9	28.2
STAGE	**23.0**	**26.3**	**29.8**
STAGE w/o boxes proximity	21.5	25.8	29.6
STAGE w/o temporal connections	22.1	25.7	29.4
STAGE w/o actors-actors interactions	22.1	26.1	28.9
STAGE w/o object-object interactions	22.3	25.6	**29.8**
STAGE w/ Transformer attention	21.6	25.6	29.6
STAGE w/ nodes Euclidean distance	21.4	**26.3**	29.5

Table 3.8: Validation mAP obtained removing some key components in our model or replacing them with other existing techniques.

in clip t, for instance, related to the same open door in clip $t+1$ could help to recognize the *open* action.

Attention and adjacency alternatives. In the last two rows of Table 3.8 we show results obtained by replacing our attention mechanism and our adjacency matrix design with other proposals. We investigate the use of dot-product attention, by replacing the weights of Eq. 3.3 with weights computed through a Transformer-like self-attention [140], as follows:

$$\boldsymbol{E} = \frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\boldsymbol{V}, \tag{3.10}$$

where $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ come from three linear projections of the input features. In this setting, as it can be noticed, we again observe a significant drop in performance. This could be attributed to the limited number of tokens, since the performance of transformer-based models usually increase with the number of tokens [115], and to the additional adjacency matrix, which may not benefit dot-product self-attention.

Taking inspiration from [169], we also replace the Euclidean distance between bounding box coordinates with the Euclidean distance between bounding box features in the adjacency matrix. We found this choice to lower the performance on both I3D and SlowFast-NL backbones. In this setting, Eq. 3.1 is replaced by:

$$\boldsymbol{A}_{ij} = \frac{1}{||\boldsymbol{h}_i - \boldsymbol{h}_j||_2}. \tag{3.11}$$

It shall be noted that all the aforementioned ablations do not change the number of parameters in the model (except for the Transformer attention experiment, where

Model	# GPUs	Clips/GPU	Epochs	Training time
ACRN [126]	11	1	63	-
Better baseline [45]	11	3	78	-
SMAD [169]	8	2	11	-
VATX [46]	10	3	71	$\sim$ 7 days
SlowFast [36]	128	-	68	-
LFB [151]	8×2	2	10×2	$\sim 2 \times 2$ days
STAGE	1	6	20	$<$ 1 day

Table 3.9: Training times and computational requirements of STAGE and existing approaches involving an end-to-end finetuning of the 3D backbone. The LFB model uses two instances of the backbone, thus has twice the complexity of its base model.

each attention head uses three linear layers instead of two), thus confirming the effectiveness of our approach. We finally note that STAGE with SlowFast-NL,8×8 backbone reaches 36.5 validation mAP on AVA 2.1 when tested with ground-truth actor boxes, suggesting that a stronger person detector could boost performances.

3.3.3 Computational analysis

Our module can reach state-of-the-art results without requiring end-to-end training of the backbone. This has an impact on the computational requirements of STAGE at training time, since the convolutional backbone incorporates most of the model complexity. Table 3.9 shows a comparison with different approaches employing end-to-end training in terms of training time and resource requirements. For each approach, we report the number of GPUs used during training, the batch size per GPU, the number of epochs, and the overall training time. The comparison is based on the implementation details reported in the original papers and refers to a training on the AVA [48] dataset. Our module requires a single GPU for training when pre-extracting backbone features, and less than a day to converge.

Finally, in Table 3.10, we report the additional FLOPs and trained parameters introduced by STAGE during inference. Please note that both the number of floating-point operations and the number of trained parameters depend on the dimensionality of the features produced by each backbone. As the amount of FLOPs also depends on the number of detections predicted on each clip, we consider a clip with a number of actor and object detections equal to the average number of actors and objects in all AVA training clips, *i.e.* 4 actors and 25 objects.

Model	GFLOPs	Parameters
I3D [18]	108	12M
+STAGE	+0.11	+6.4M
R101-I3D-NL [151]	359	54.3M
+STAGE	+0.24	+17M
SlowFast-NL,16×8 [36]	234	59.9M
+STAGE	+0.26	+21.8M

Table 3.10: Computational complexity analysis for inference.

3.3.4 Qualitative analysis

We present some qualitative results obtained on clips of the AVA validation set in Fig. 3.5. Here, we only show the central keyframe of the clip; red and blue boxes represent predicted actors and objects respectively. For simplicity, we highlight only the actor involved in the action (despite other actors could be found in the scene), except for the *Kiss* class, where two actors perform the same action. Only predicted objects with score greater than 0.8 are shown, even if all detections are used during training. Fig. 3.6 shows sample failure cases. On average, we qualitatively observe that our spatio-temporal graph-based module is able to improve the recognition of human actions, especially for actions involving relationships with objects and other people.

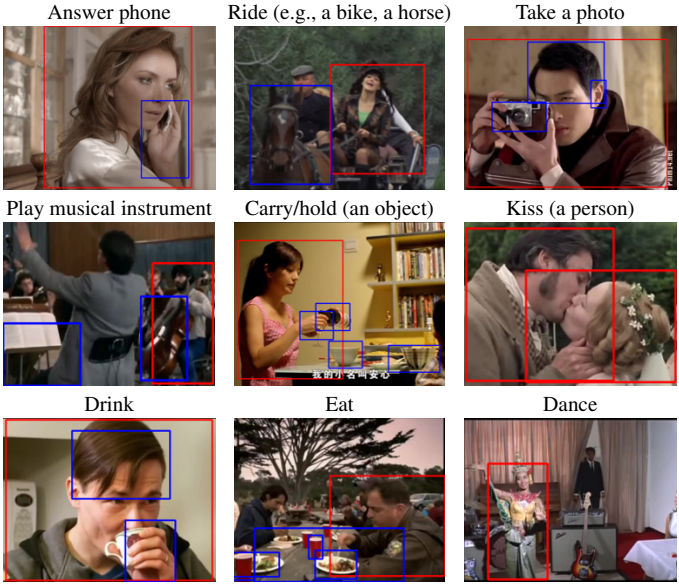


Figure 3.5: Qualitative results showing the keyframes of the evaluated clips, with red boxes denoting actors performing actions, and blue boxes denoting objects.

Figure 3.6: Sample failure cases. Actions are sometimes assigned to an actor very close to the one actually performing the action, and some object-interaction classes are wrongly assigned to people very close to the objects.

Chapter 4

Soccer action spotting

As already mentioned, all the work presented in this thesis has been granted by and done in collaboration with Metaliquid S.R.L. Since the needs of the research community and those of a company are not often the same, all the applications and models developed in this thesis had to meet the requirements of both of them. We followed the standard practice of tackling well-defined research problems available in literature, by recasting them to company-specific applications and needs. The approach presented in Chapter 3 has been developed without focusing on a specific real-world application, since the fine-grained scene representation could be suitable in different scenarios.

In this chapter, instead, we tackle a problem raised by a specific company requirement, by turning it into a well-known research challenge. With the goal of automatically generate highlights from broadcast soccer matches, we decided to address the soccer event spotting task [44] as a first step in this direction.

Recently, there has been an increasing effort in developing architectures for automatic soccer match understanding starting from videos, and in action spotting [22, 44, 141], specifically. The task requires to temporally localize all significant events happening inside the match like goals or yellow/red card events. Differently from the task presented in the previous chapter, we do not need to recognize actions for each person in the clip, but a single event possibly involving multiple actors. For this reason, we rely on more coarse-grained features, and we do not detect individual players. As these events are sparse within a video,

This chapter is related to publication "M. Tomei et al, Rms-net: Regression and masking for soccer event spotting, ICPR 2020". See Appendix B for details.

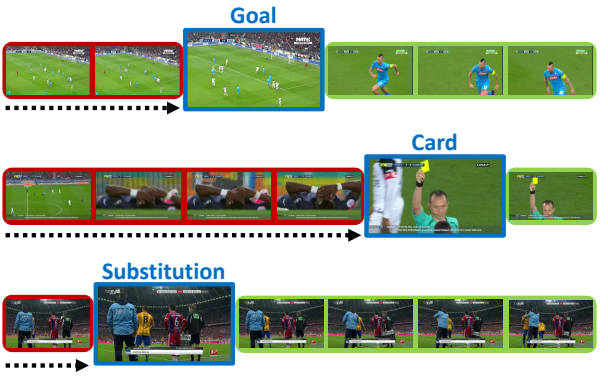


Figure 4.1: We propose a lightweight and effective network for spotting relevant events in soccer matches. Our model features a masking strategy which increases training performance by constraining the network to focus on the most relevant parts of the video and a data sampling solution which handles class imbalance.

the task couples the need for effective feature extraction with that of properly handling the data imbalance and event sparsity issues. An effective action spotting approach needs to provide accurate temporal localization, too, which requires proper architectural and training strategies.

Contributions. We treat the action spotting as a detection problem and devise a novel network which takes inspiration from the regression strategies used in object detection. Specifically, our network takes a short video snippet as input and predicts the presence of a candidate event together with its class and relative temporal position inside the input snippet. This choice is innovative compared to recent works in the field, which usually assign the event to the central frame of the video chunk, thus preventing the model from learning an accurate localization of actions. The novelty also consists in dealing with the sparsity and class imbalance issues: through a sampling approach we ensure a uniform distribution in training batches in terms of ground-truth classes and action locations.

Further, we develop a strategy to increase the generalization capabilities of the network, which is inspired by the masking strategies used in self-supervised pre-training techniques [29]. During training, indeed, we randomly mask and replace the frames preceding an event and constrain the network to focus on the most relevant parts of the action.

We conduct extensive evaluations on the recently released SoccerNet dataset [44], which features 500 full broadcast soccer matches, annotated with relevant events. We provide experiments to validate the architectural choices behind the proposal and the effectiveness of the masking and data sampling strategies. When compared to the state of the art, our network achieves a gain of 3 Average-mAP points exploiting the same features used by previous works, while being significantly more lightweight. We also conduct experiments with different feature extraction backbones, and investigate the role of fine-tuning such backbones, devising a combination which further pushes the state of the art of additional 10 Average-mAP points.

The rest of the chapter is organized as follows: in Sec. 4.1 we detail our soccer event spotting method and the proposed training strategies, in Sec. 4.2 we present experiments with quantitative results, ablation study and qualitative analysis, and in Sec. 4.3 we describe our submission to the 2021 SoccerNet-v2 challenge, in which our solution placed third.

4.1 Proposed Method

In the following, we present our approach for soccer event spotting. Our formulation features a lightweight network that can be integrated with any existing backbone, jointly predicting classification scores and temporal offsets. This is combined with a masking strategy that increases the training performance, and with an effective handling of data imbalance.

4.1.1 Proposed architecture

A soccer match can be represented as a sequence of frames $(x_1, x_2, ..., x_N)$, with $x_i \in \mathbb{R}^{H \times W \times Ch}$, extracted with a given frame rate from the original video clip. The goal of our approach is to spot and classify a set of interesting events inside the match, *i.e.* to predict a set of pairs $\{(t_j, e_j)\}_{j=1}^{n}$, where t_j indicates the timestamp of the action from the beginning of the video, and $e_j \in \{0, ..., C-1\}$ indicates its class label.

Taking inspiration from the regression strategies used in object detection [111], our model takes as input a short video chunk $\boldsymbol{X} = (x_1, x_2, ..., x_T)$ with length T, and predicts the temporal offset of a possible action inside the chunk, plus a classification score over $C+1$ classes, where the additional class indicates the background one. At test time, predictions can be obtained by applying the model

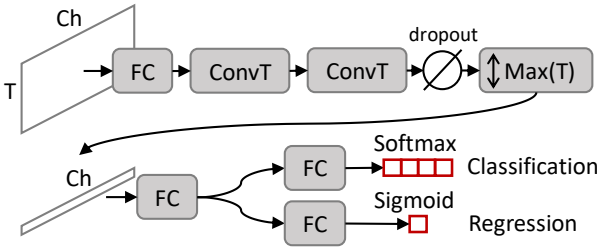


Figure 4.2: Architecture of the proposed spotting network.

on chunks extracted from the input video with a given stride, converting relative offsets to absolute timestamps, and accumulating predictions over time.

In our network, the T input frames are independently passed to a 2D backbone to extract a feature vector for each frame, which is obtained by averaging the activation map's spatial dimensions from the last convolutional layer and linearly projecting the result to a common dimensionality. As depicted in Fig. 4.2, our model then applies a stack of two 1D convolutions over the temporal dimension to combine the features of different frames. After the convolutions, a *maximum* operation is applied to remove the time axis, and a stack of linear layers is added. The tail of the model consists of two sibling fully-connected layers, one for action classification and the other for temporal offset regression.

Given a "foreground" input clip containing an event (t_j, e_j), the classification layer outputs per-class probabilities and a cross-entropy loss is applied:

$$\mathcal{L}_{cls} = -\sum_{c=0}^{C} \mathbb{1}_{c=e_j} \log(p_c), \tag{4.1}$$

where $C+1$ is the number of different event labels, including the *background* label, $\mathbb{1}_{c=e_j}$ is the indicator function, and p_c is the model output probability for event label c. On the other hand, the regression layer predicts a single scalar per clip, which is projected in the range $[0, 1]$ through a sigmoid function and trained to predict the normalized relative offset of the event in the video chunk. Formally, we apply a squared-error loss as follows:

$$\mathcal{L}_{regr} = (\sigma(o) - r_j)^2, \tag{4.2}$$

where o is the output of the regression branch of the network, and the normalized

relative offset r_j is computed as $(t_j - s)/T$, where s is the starting timestamp of the video chunk.

The final loss for a foreground chunk is a weighted sum of $\mathcal{L}_{cls}$ and $\mathcal{L}_{regr}$:

$$\mathcal{L} = \mathcal{L}_{cls} + \lambda \mathcal{L}_{regr}; \tag{4.3}$$

for "background" video chunks, *i.e.* chunks which do not contain any event, we instead apply only the classification loss, using the background class as ground truth label. As it can be noticed, the proposed method predicts a single event per clip. It could be easily extended to predict more than one event, *e.g.* including for instance a bipartite matching strategy between predictions and ground truth labels [16]. In practice, we noticed that interesting events are very sparse in a soccer match. The only rare circumstances in which more than one action occurs in a few seconds interval correspond to double substitutions or double yellow/red cards. While the choice of having a single action predictor could slightly affect spotting performances, it does not affect automatic highlight generation, which is the final goal of event spotting in soccer videos and allows us to maintain a simple and lightweight architecture.

4.1.2 Masking strategy

The majority of visual cues that contribute to the recognition of an event occur *just after* the event itself [22]. This is particularly evident in the case of soccer matches, in which reactions to an event are often a good indicator of the presence of the event itself, like in the case of the celebrations following a *goal* event. Taking inspiration from the masking strategies used in self-supervised pre-training approaches [29], we endow our approach with a masking policy that allows the network to focus on the most relevant portions of the clips at training time.

In our formulation, a foreground training clip is masked with a given probability by replacing the frames before the event with a randomly chosen background sequence (*i.e.* which does not contain any event). We also make sure that there is a sufficient number of frames after the event, by avoiding the application of masking on clips in which the event comes too late. Background training clips, instead, are not masked at all.

Formally, we define a masking function $M(p, q)$, in which p indicates the masking probability and q is the maximum normalized relative temporal offset of the event inside the clip. Given a foreground clip $\boldsymbol{X}$ containing an event (t_j, e_j),

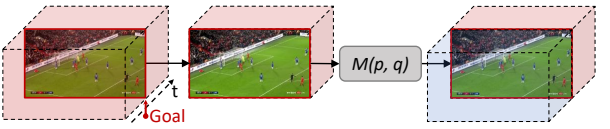


Figure 4.3: Masking strategy. Frames before the event are masked with probability p, if the event lies before the offset q.

the masking function $M(p, q)$ is defined as follows:

$$M(p,q)(\boldsymbol{X}) = \begin{cases} (z_1, ..., z_{t_j-s-1}, x_{t_j-s}, ..., x_T) \\ \quad \text{if } r_j \leq q,\, u < p \\ \\ (x_1, ..., x_{t_j-s-1}, x_{t_j-s}, ..., x_T) \\ \quad \text{otherwise,} \end{cases} \tag{4.4}$$

where s is the starting timestamp of the video chunk, r_j is the normalized relative offset of the event inside the video chunk, $(z_i)_{i=1}^{t_j-s-1}$ is a sequence of consecutive frames randomly selected from a background clip, and u is a random value sampled from the uniform distribution $U\,[0, 1]$. A visualization of the masking strategy is also reported in Fig. 4.3.

Through this masking approach, we randomly force the model to recognize an event using just the frames *following* the event itself, and without relying on the previous ones. We will experimentally show how this masking strategy, which does not require any architectural change or additional resources, is robust to different values of the masking probability p, and how it can increase the recognition and localization accuracy of the model.

4.1.3 Data Sampling and Balancing

The training set of our architecture is composed of all possible clips with length T which can be extracted from a set of matches. Since relevant events in a soccer match are quite sparse, a balancing strategy is needed to make sure that the network can learn to properly classify relevant events, without losing the capability of distinguishing background clips. At the same time, we need to make sure that the distribution of the offsets used to train the regression branch is sufficiently uniform.

Given a training soccer match and an anchor event (t_j, e_j), we extract all clips with length T containing the event, sliding a window along the time axis with stride 1. Doing so, we generate many clips for the same event, where each of them contains the event in a different relative temporal location. Repeating this procedure for each event in a match, and then for each match in the training set, we collect a set of interesting events, where the distribution of the relative position of events inside clips is balanced by construction.

The remaining parts of the matches, which do not contain any event, are sliced with a window of size T and stride T (thus avoiding overlapping), to build the set of background video chunks. In each training epoch, we randomly sample n_F foreground clips from the above mentioned set, and $n_B = n_F/C$ clips (where C is the number of classes) from the set of background clips, to balance the number of samples per class. During inference we extract non-overlapping clips, each with T frames, in a sliding-window manner, assuming that T is small enough to prevent that more than one action occurs inside a clip.

4.2 Experimental results

4.2.1 Dataset and evaluation protocol

Data. We train and evaluate the proposed method on the SoccerNet dataset [44]. SoccerNet gathers 500 full broadcast soccer matches, spanning 764 hours of video, which are split into 300 games for training, 100 for validation, and 100 for testing. Interesting events belong to three categories (*goal*, *card*, *substitution*) and are manually annotated with a temporal resolution of one second. The average separation between events is 6.9 minutes, thus leading to a very sparse annotation.

For fairness of comparison with previously published baselines and state-of-the-art approaches, all experiments are performed with the pre-computed ResNet-152 [52] features released with the dataset itself, unless otherwise specified. These have been obtained by Giancola *et al*. [44] by resizing and cropping videos at a resolution of 224×224 and extracting frames at a frequency of 25 fps. Starting from this extraction, a feature vector was computed every 0.5 seconds. They also applied a PCA step to reduce the dimensionality to 512. The feature extraction backbone was pre-trained on ImageNet [28].

Evaluation metric. The action spotting task requires to correctly predict the anchor *spot* that identifies an event. For instance, a ground truth spot for a *card* event is defined as the timestamp in which the referee extracts the card.

Layer	Input Channels	Output Channels
FC_1	512	256
Conv_1	9×256	256
Conv_1	9×256	128
Dropout	–	–
Max over time	–	–
FC_2	128	64
FC_{cls}	64	C (number of classes)
FC_{regr}	64	1

Table 4.1: Structure of the proposed architecture.

Following previous literature, we use the Average-mAP defined in [44] as the evaluation metric, which accounts for multiple temporal tolerances. Given a temporal tolerance δ, we compute the average precision for a class by considering a prediction as positive if the distance from its closest ground truth spot is less than δ. The mean average precision is then obtained by averaging the AP of each class. The final metric is computed as the area under the mAP curve obtained by varying δ in the interval ranging from 5 to 60 seconds.

4.2.2 Implementation details

In all our experiments, the model takes $T = 41$ frames as input, therefore spanning 20 seconds of the match. The weight λ of the regression loss (defined in Eq. 4.3) is set to 10, unless otherwise specified. Table 4.1 reports the architectural details of the model, including the number of input and output channels. Each 1D convolutional layer has a kernel size of 9, a stride of 1, and zero-padding to keep the temporal dimension shape. The drop rate of the dropout layer is set to 0.1.

Training. Following our data sampling strategy, we obtain a total of around 150,000 foreground and 72,000 background video chunks. At each training epoch, we randomly sample 30,000 foreground sequences and 10,000 background sequences, being the number of foreground classes C equal to three in SoccerNet. During training, we also drop all *substitution* events occurring at half time, since no visual cues suggest that a substitution is happening.

The masking probability p is set to $1/3$, while the maximum relative temporal offset q is set to 0.5, unless otherwise specified. We train our model for a maximum of 50 epochs using an SGD optimizer with momentum 0.9. The batch size is set to 64 and we apply a learning rate of 0.05, with a linear warm-up during the first

Model	Clip len (s)	Features	Val Avg-mAP	Test Avg-mAP
SoccerNet baseline [44]	5	ResNet-152 (PCA)	-	34.5
SoccerNet baseline [44]	60	ResNet-152 (PCA)	-	40.6
SoccerNet baseline [44]	20	ResNet-152 (PCA)	-	49.7
Vanderplaetse *et al*. [137]	20	ResNet-152 (PCA) + Audio	-	56.0
Vats *et al*. [141]	15	ResNet-152 (PCA)	-	60.1
Cioppa *et al*. [22]	120	ResNet-152 (PCA)	-	62.5
Ours	20	ResNet-152 (PCA)	**67.8**	**65.5**

Table 4.2: Comparison with baselines and state-of-the-art approaches.

epoch and a cosine annealing scheduling from the second epoch. A weight decay of 10^{-4} is also adopted. Early stopping on the Average-mAP computed over the validation set is applied. Training is done on a NVIDIA RTX 2080Ti GPU.

Inference. During inference, we slide a window containing T frames on the input video with a stride of T, and the clip class, together with the event relative temporal offset, is predicted by the two sibling fully connected layers. The predicted relative offset is then converted to absolute timestamp, to obtain the predicted spot location. No masking is applied during inference.

4.2.3 Main results and comparison with the state of the art

In the following, we validate the proposed approach for action spotting in soccer video. We first perform a comparison with baselines and state-of-the-art methods and then analyze and ablate the key components of the approach. Finally, we investigate the performance of the proposed architecture when using different 2D convolutional backbones.

Comparison with baselines and previous methods

In Table 4.2 we report the performance of our model, in comparison with previous approaches for action spotting. All the reported approaches adopt the ResNet-152 features released with the dataset, and we do the same for fairness of comparison. The only exception is the approach presented by Vanderplaetse *et al*. [137], which enriched the same visual features by encoding the audio stream through a VGG network [122].

As it can be noticed, our approach outperforms the best SoccerNet baseline (with $T = 20$ seconds) by 15.8 Average-mAP points and the to-date best perform-

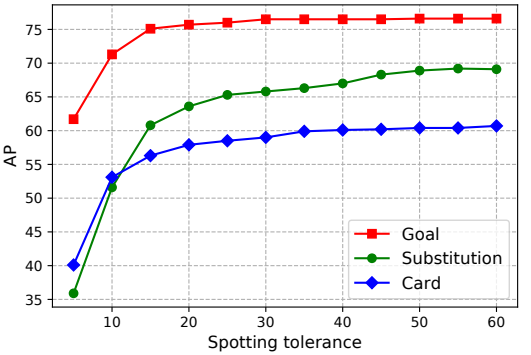


Figure 4.4: Per-class Average Precision, as a function of spotting tolerance.

ing method by 3 Average-mAP points on the test set of SoccerNet. Noticeably, this result is achieved without any matching strategy between predictions and ground truth, and without post-processing steps like non-maxima-suppression.

We also conduct a grid search to find the optimal clip duration T, as done in Giancola *et al*. [44] on the SoccerNet baselines (and reported in Table 4.2). Also with our architecture, the optimal clip duration is 20 seconds.

Per-class performances. In Fig. 4.4 we report the AP computed for each class, as a function of the spotting tolerance. The best performing events are *goals*, with a margin of around 5 AP points compared to the *substitution* class when allowing high spotting tolerances. This is in line with previous literature, and can be attributed to the richness of visual cues which are usually found after a *goal* event (*e.g*. celebration, replays). *Card* events are, instead, the most difficult to spot, as the presence of a yellow or red card is the only visual indication that can distinguish those events from a general foul. Finally, it can be observed that *goals* are also the events which can be localized with the greatest temporal precision, compared to the other classes.

4.2.4 Ablation studies

Removing key components. We investigate the role of the key components of our approach by conducting an ablation study. First, we assess the role of employing a

Model	Val Avg-mAP	Test Avg-mAP
Ours	**67.8**	**65.5**
Ours w/o uniformly distributed offsets	48.7	46.2
Ours w/o offset regression branch	58.5	55.7
Ours w/o masking	66.5	64.0

Table 4.3: Performance of the proposed model when removing key components.

uniformly distributed normalized relative offset in foreground clips. To this aim, we modify our data sampling strategy by considering only the chunks that contain an event in the middle of the clip (thus, with a normalized relative offset of 0.5), and removing the regression output. In this case, we still balance training batches by ensuring a uniform number of clips for each class, including background. At prediction time, we assume a normalized relative offset of 0.5. Noticeably, this evaluation setting is similar to the one adopted in [44] and [141]. As can be seen from Table 4.3, this leads to a significant performance drop, thus testifying the need for uniformly distributing relative offsets.

In a second ablation experiment, we instead maintain the original data sampling strategy and exclude the regression branch, to estimate its contribution to the final performance. Also in this case, we assume that the relative temporal offset of a predicted event is always 0.5. This leads to a drop of the test Average-mAP of almost 10 points, as reported in Table 4.3. Fig. 4.5 also shows a more detailed comparison for different values of the tolerance threshold δ using this regression-free model (in blue) and our full proposal (in red). We notice that, for high values of δ, the resulting mAP is similar for both models. When δ decreases (and in particular when δ is lower than the clip duration) our model avoids an abrupt decrease of the mAP, which instead occurs when removing the regression branch. This confirms that the regression loss can increase the localization accuracy of the prediction.

Further, we also test the contribution given by the masking strategy. When removing the masking, we observe a decrease of 1.5 Average-mAP points on the test set, as reported in Table 4.3. We also underline how masking represents a cost-free improvement, which does not require any additional resource or model parameter.

Masking analysis. In Table 4.4 we show how performances vary when changing the masking probability p and the maximum relative temporal offset q for masking. We achieve the best configuration with $p = 1/3$ and $q = 0.5$, which means that

p	q	Val Avg-mAP	Test Avg-mAP
1/5	0.5	66.8	64.6
1/4	0.5	67.0	64.4
1/3	0.5	**67.8**	**65.5**
1/2	0.5	67.1	64.4
1	0.5	64.7	60.7
1/3	0.1	65.5	63.4
1/3	0.25	67.4	64.7
1/3	0.5	**67.8**	**65.5**
1/3	0.75	66.5	64.0
1/3	1	64.7	62.6

Table 4.4: Performance when varying the masking probability p and the maximum relative temporal offset q for masking.

p	q	Val Avg-mAP	Test Avg-mAP
1/5	0.5	65.2	62.5
1/4	0.5	64.6	62.9
1/3	0.5	63.8	61.8
1/2	0.5	61.4	60.7
1	0.5	54.1	54.1

Table 4.5: Performance when varying p, keeping $q = 0.5$ fixed and masking frames *after* the event.

clips with the event in their first half have a $1/3$ probability of being masked during training. Keeping $q = 0.5$ fixed and varying p, the Average-mAP always exceeds the performance of the masking-free model (64.0 Avg-mAP) except for $p = 1$. When $p = 1$, indeed, a clip is always masked if it has the event in the first half. This creates a big gap between the train and test distributions and performances drop as expected. Similarly, the model is robust to different values of q, except when all the clips have probability $p = 1/3$ to be masked, independently of where the event lies inside it (*i.e.* when $q = 1$). In this case, clips with too little context after the event can be masked too, resulting in lower Average-mAP.

One can question if the majority of visual cues actually occurs just after the event, and not before. Table 4.5 shows the results when masking frames *just after* the event with different values of p, while always keeping the frames *before* the event. Under this setting, masking is not beneficial and lowers the final performance.

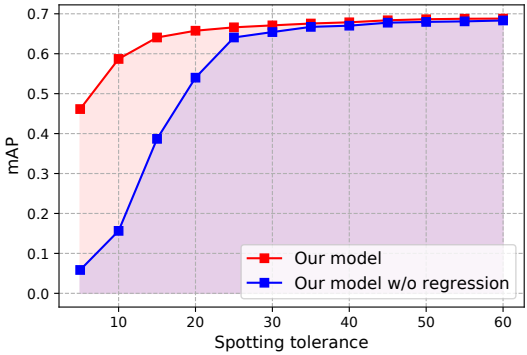


Figure 4.5: Mean average precision when varying the tolerance δ in the interval between 5 and 60 seconds, for our complete model (red curve) and when removing the offset regression branch (blue curve). The Average-mAP gain can be quantified as the area under the red curve and above the blue one.

Additional analysis. Finally, we look for the best value of hyperparameter λ, which controls the relative weight of the regression loss with respect to the classification loss (see Eq. 4.3). Fig. 4.6 shows the test Average-mAP when varying λ in the interval between 0 and 100. The best value is achieved with $\lambda = 10$. When $\lambda = 0$ there is no temporal offset regression in training, and we fall back in a setting similar to that of Table 4.3, third row. In this case, however, the relative temporal offset is still predicted by the network (without any supervision) and not fixed to the clip center timestamp. The gap between the setting with $\lambda = 0$ and $\lambda = 1$ exceeds 10 Average-mAP points. Performances decrease when $\lambda > 10$.

4.2.5 Changing the convolutional backbone

Further, we present additional results when changing the convolutional backbone used for feature extraction, and when fine-tuning part of it during the training of the action spotting model. Here, we employ different variants of ResNet, pre-trained on ImageNet. For assessing the role of fine-tuning, only the parameters belonging to the last residual block and our model are trained, while the other parameters of the backbone are kept fixed, as we did not observe significant improvements when fine-tuning larger portions of the backbone.

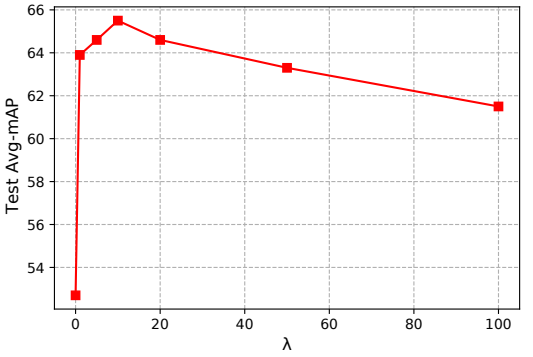


Figure 4.6: Performance when varying the regression weight λ.

Model	Pre-train	Val Avg-mAP	Test Avg-mAP
ResNet-18 + Our	ImageNet	73.8	70.9
ResNet-50 + Our	ImageNet	76.6	74.9
ResNet-152 + Our	ImageNet	**77.5**	**75.1**

Table 4.6: Performance analysis when finetuning different variants of ResNet.

Table 4.6 shows the performances obtained when finetuning ResNet-18, ResNet-50 and ResNet-152. We train these models on two RTX 2080Ti GPUs, with a batch size of 24 clips and a base learning rate of 0.025. RGB frames are extracted from the low resolution videos (224×398) at a rate of 2 fps, thus keeping the entire frame instead of center cropping as done in the features released with SoccerNet. No spatial augmentation is performed, and all the other implementation details are kept unchanged.

When using ResNet-152 as our backbone, we observe a boost of 9.6 Average-mAP points on the test set when compared to our best performing model trained on the pre-computed ResNet-152 features. A similar performance level is obtained when finetuning ResNet-50, while a more significant performance loss is visible when using ResNet-18. While pre-computed features are a good starting point for research and comparison purposes, our findings underline that end-to-end training still guarantees a significant performance boosting.

4.2.6 Qualitative analysis

Finally, we qualitatively assess the spotting capabilities of our model. For this purpose, we extract frames from the raw video with a resolution of 2 fps and create overlapping clips having a length of T frames, with stride 1. For each clip, our model predicts the event temporal offset and its label, and we count the number of times a frame is predicted as a spot. As can be seen from Fig. 4.7, the most voted frame index (the peak of the blue plot) is often very close to the ground truth spot (highlighted in red), and usually lies in an interval of 10 frames around it, corresponding to the lowest considered tolerance δ in the Average-mAP metric.

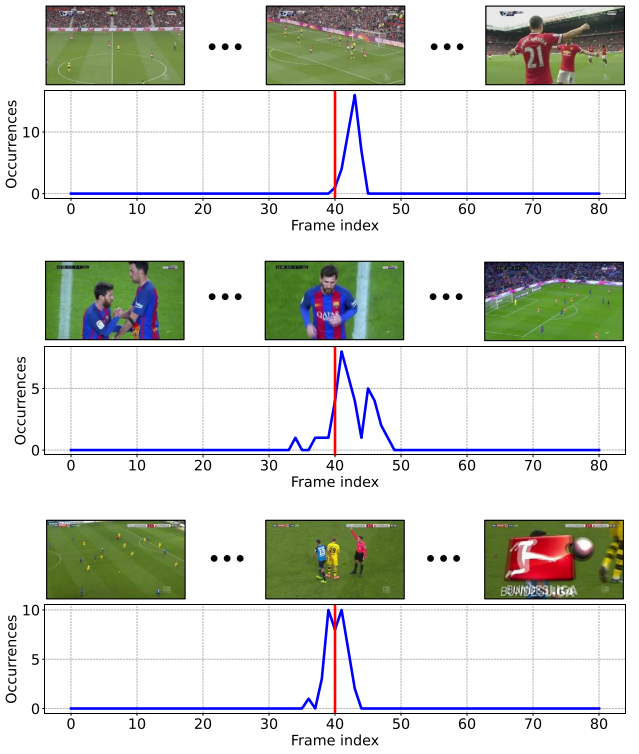


Figure 4.7: Qualitative results. The ground truth action timestamp is shown in red, while the blue curve shows the number of times a frame index was predicted as a spot. A *goal*, a *substitution* and a *card* events are shown, from top to bottom.

4.3 SoccerNet challenge submission

In this section, we present our solution for soccer event spotting used to obtain the results submitted to the SoccerNet-v2 evaluation server by the "AImageLab-RMS" participant team. Our solution placed third in the SoccerNet-v2 challenge of the CVPR 2021 International Challenge on Activity Recognition Workshop.

SoccerNet-v2 [27] is an extension of the original SoccerNet dataset [44]. It is composed of the same original 500 full broadcast soccer matches, but with enriched annotations, for both event spotting and other new tasks. It provides 17 event classes, annotations of camera replays linked to actions, and annotations of camera changes and camera types for 200 games. The leaderboard of the challenge is determined by the average-mAP over an undisclosed challenge set, while authors provide three open sets for training, validation, and test, respectively.

Our code is publicly available at: `https://github.com/aimagelab/RMSNet_Soccer`.

Data preparation

Here we describe how we processed the original SoccerNet [27, 44] videos before training our action spotting network. The original Full HQ videos are all converted to 25 FPS, before extracting and saving frames at 2 FPS for each match. Only those frames in [start_time_second, start_time_second + duration_second] are considered, as specified in the "video.ini" annotation files provided with the SoccerNet-v2 dataset. These operations are carried out using the ffmpeg libraries. Frames are stored at their original high-quality resolution. About 1 TB of free space is needed to store frames extracted in this way.

Adopted model

We use the same network presented in Sec. 4.1 as our action spotting model, by fine-tuning the feature-extraction backbone together with it. Specifically, we adopt a ResNet152 [52] backbone, which is initialized with ImageNet [28] pre-trained weights. Each frame in the input clip is independently forwarded through the backbone, before applying the module presented in Sec 4.1.1. The classification branch outputs 18 class probabilities, 17 for the SoccerNet-v2 classes and 1 for the additional "background" class. The network takes 41-frames video clips as input, at a temporal resolution of 2 FPS, *i.e.* spanning 20 seconds of the match.

Before being forwarded through the network, each frame is resized to 300×533, and no spatial crop or other augmentation techniques are applied.

Data loading and balancing

Since SoccerNet-v2 is highly unbalanced, we define a balancing strategy before starting training. Specifically, following the data sampling approach presented in Sec. 4.1.3, given a ground truth spot we put all the possible 41-frames clips containing that spot in our pool of possible training samples, by sliding a 41-frames window along the time axis with stride 1. Therefore, the size of our training pool is equal to the number of ground truth spots times 41. At the beginning of each training epoch, for each class we randomly sample a number of clips equal to the number of ground truth spots for that class. For those classes with more than 10,000 ground truth spots we divide this number by 5, while for those classes with less than 100 ground truth spots we multiply this number by 10, for data augmentation. For the additional "background" class, we sample a number of clips in each epoch equal to the average number of ground truth spots in the other 17 classes. Background clips are randomly sampled from the remaining portions of the videos, which do not overlap with clips containing ground truth spots.

Training and inference

The adopted backbone is initialized with ImageNet pre-trained weights, and is fine-tuned from conv4 to the final classification and regression branches. The original masking strategy presented in Sec. 4.1.2 is not performed in these experiments during training. We train our model on all the available SoccerNet-v2 videos (training, validation, and test sets) for a total of 20 epochs, before computing the predictions on the challenge set and submitting the results to the evaluation server. We train our models using an SGD optimizer with a batch size of 32 clips and a base learning rate of 0.0032, with a linear warm-up during the first epoch and a cosine annealing scheduler from the second epoch. A momentum of 0.9 and a weight decay of 10^{-4} are also adopted. Training has been performed on the CINECA Marconi100 accelerated system, using a total of 32 synchronized nodes, each consisting of 4 V100 GPUs.

During inference, we predict class labels and regression offsets for each clip obtained by sliding a 41-frames window along time with a stride of 4 frames. Non-maxima suppression is applied on the predictions using a 15-seconds (31-frames) window.

Results

The public leaderboard is available at: https://eval.ai/web/challenges/challenge-page/761/leaderboard. Our method placed third in the challenge, reporting an average-mAP of 60.92 on the private challenge set. The performance on the test set (which is not considered for the challenge, since annotations for the test set are publicly available) is 63.49 Average-mAP, after training on both training and validation splits.

Chapter 5

Privacy-preserving action recognition

As the importance and the ubiquity of video understanding systems, ranging from action recognition to actor localization and moment retrieval [121, 89], become more and more relevant, the same consideration does not seem to be given to their drawbacks. Among these, users' privacy and computation sustainability are of primary importance in the development of video recognition systems, which usually depend on data disclosing sensible information and on huge computational requirements. In this chapter, we investigate privacy-related issues, while we will address the computational aspects in the next chapter of this thesis.

Most visual understanding models are trained to work on unaltered videos easily containing private information from which identities can be revealed [159]. If the whole prediction pipeline – ranging from video acquisition and processing to the execution of the prediction model – can be trusted, this might not be an issue. When, instead, the prediction model is run by a third-party service provider which might not be trusted, employing a visual understanding model comes at the cost of transmitting sensitive information and potentially breaking the users' privacy.

With the goal of protecting the privacy and creating trusted pipelines, video understanding models should ideally work on videos with obfuscated private

This chapter is related to publication "M. Tomei et al, Estimating (and fixing) the effect of face obfuscation in video recognition, CVPRW 2021". See Appendix B for details.

Figure 5.1: Recognizing actions in videos comes at the cost of transmitting sensitive information to service providers. We investigate the development of privacy-preserving action prediction networks which can work on obfuscated videos, either when blurring faces or when blurring the full body. The top row shows sample clips (*archery* on the left, *arm wrestling* on the right) when blurring faces only, while the bottom row reports samples of full body obfuscation.

information. Blurring faces, for instance, is a reasonable first step to protect the privacy and hide identities. However, as sensitive information can be found in clothes and fashion accessories as well, this might not be enough in some contexts, and a more severe protection level could be needed, such as obfuscating the entire human figure (Figure 5.1). Clearly, this comes at the cost of removing a significant portion of the information contained in video frames and creates non-trivial challenges when it comes to extracting spatio-temporal features needed to perform video understanding tasks.

Contributions. In this chapter, we focus on the development of *privacy-preserving video models*, *i.e*. architectures which can work on anonymized videos while still guaranteeing high accuracy levels, regardless of the loss of information caused by anonymization. We investigate the role of face and body anonymization, and their impact on the effectiveness of existing video networks. Further, we analyze three training strategies for privacy-preserving video prediction architectures, which focus on the reduction of the domain gap, on the transfer of privileged knowledge from non-anonymized data, and on the transfer of mutual relationships between samples through a relational criterion. A high-level representation of the approach is presented in Fig. 5.2. Through extensive experiments on the Kinetics-400 dataset [70], we validate the proposed solutions and demonstrate that, with proper training strategies, state-of-the-art video backbones can work with negligible accuracy loss even in presence of full-body obfuscation. In this chapter, we first assess the role of people's identities in common action recognition

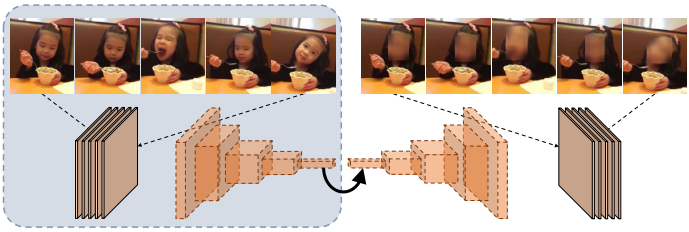


Figure 5.2: We propose a training schema that can prevent the performance gap caused by anonymization in video classification networks. A network is trained to work on anonymized images by reading knowledge from a network that works on full images.

datasets (Sec. 5.1). Then we present our training strategies in Sec. 5.2, and finally evaluate them in Sec. 5.3.

5.1 Sensitive information in action videos

In this section, we first introduce the two privacy preservation levels we employ, in which we hide the identities of the actors by masking either their faces or their full bodies. We also conduct analyses on Kinetics-400 [70], one of the largest publicly available video classification databases, to quantify the presence of sensitive information in video classification datasets, and we motivate the appropriateness of building privacy-preserving video networks.

Anonymizing sensitive information in videos

Given a video clip, we independently apply a pre-trained frame-level face detector [83] to detect faces or an instance segmentation model [51] to detect full-body masks. We detect faces and segment instances for all video frames, obtaining face and body masks for all the involved actors. We then build an anonymized version of the initial video, by blurring all faces or bodies with a normalized box filter. Figure 5.1 reports qualitative results obtained by applying the two anonymization levels. In our initial experiments, we found that applying a frame-level detection or segmentation strategy on Kinetics-400 [70] is enough to guarantee complete anonymization of all face and body instances in the vast majority of the

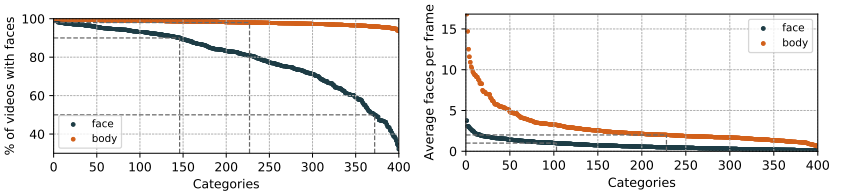


Figure 5.3: The left plot reports the per-class percentage of videos containing potentially sensitive information (faces or body). The right plot shows the average number of per-frame potentially sensitive instances (faces or body). Both of them are computed for each category of the Kinetics-400 training set.

videos, without sorting to the usage of tracking or temporal coherency techniques, which could, however, be necessary for more complex scenarios.

Sensitive information in Kinetics-400. Kinetics-400 [70] consists of roughly 240k training videos and almost 20k validation videos. Each video lasts around 10s and belongs to one of 400 different action categories. Since video clips have been sourced from YouTube, there are no constraints on camera motion, illumination, and viewpoint.

To assess the extent of sensitive information in the dataset, and hence its possible influence in training action recognition networks, we measure the number of faces and bodies found using the aforementioned strategies. The left plot of Figure 5.3 reports the per-class percentage of training videos containing potentially sensitive information – *i.e.* that have at least one frame containing a bounding box predicted by the face detector model [83] or at least a person mask as predicted by the instance segmentation model [51]. As it can be seen, for all classes at least 93% of the videos involve a person instance which might disclose sensitive information (*e.g.* clothes, fashion accessories, etc.). This level further increases over 98% for 227 classes out of 400.

When considering face instances, instead, in the majority of the classes (372 out of 400), more than 50% of the available videos contain faces. For 146 categories, instead, more than 90% of videos show at least one face, and there are no classes with less than 30% of videos containing faces. This highlights the importance of dealing with a potential domain gap when building privacy-preserving video networks, as most of the training data contain sensitive information.

Per-class instance count analysis. We further quantify the role of sensitive information by analyzing the number of sensitive instances found in Kinetics-400

Figure 5.4: Sample video from Kinetics-400, annotated with face bounding boxes.

videos. In the right plot of Figure 5.3, indeed, we report the average number of body instances and faces per-frame, for all categories. Out of 400 classes, 228 have more than two body instances, and 103 have more than one face per frame, on average. The majority of classes have more than one body instance and between zero and one face per frame, underlining that many videos capture the actor but not its face (*e.g*. when the actor is captured from behind or when only a portion of his body is visible). It must be also noted that for some actions it is common to have the actor's face visible for a limited period while being occluded or not visible in the remaining frames (like in the *golf driving* sample reported in Figure 5.4).

5.2 Privacy-preserving action recognition

Traditional action recognition approaches require the transmission of sensitive information to the model, and hence to the service provider. To protect the privacy of people involved in videos, we investigate the development of privacy-preserving action recognition pipelines, which can extract spatio-temporal features from input videos and classify actions even after aggressive privacy preservation strategies have been applied.

In this regard, we analyze three strategies for training video models that do not need sensitive information to reach high accuracy levels, *i.e*. which can recognize actions even in presence of largely obfuscated areas (cfr. Figure 5.1). In all the examined strategies, we consider our data as formed by a collection of triplets

$$\{(x_1, x_1^*, y_1), ..., (x_n, x_n^*, y_n)\}, \tag{5.1}$$

where each pair (x_i, y_i) is an anonymized video-label pair, *i.e*. where sensitive data has been obfuscated in video clip x_i; while x_i^* is the original, non obfuscated, version of the video that discloses actors' identities and sensitive information. De-

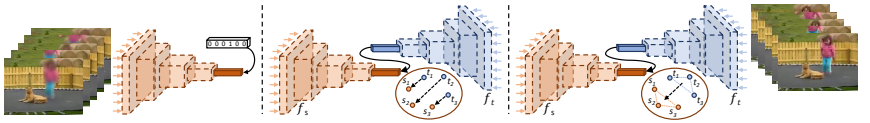


Figure 5.5: The three training strategies we consider for developing privacy-preserving video networks. From left to right: (a) reduction of the domain gap by training on obfuscated data; (b) Knowledge Distillation with privileged (non-obfuscated) clips; (c) Relational Knowledge Distillation, which ensures the preservation of pair-wise distances in the representations extracted from obfuscated samples.

pending on the particular strategy at hand, we will exploit the identity information in x_i^* during training; however, during inference, only the obfuscated clip x_i will be leveraged in any case to obtain the final predictions.

5.2.1 Reducing the domain gap

Because a traditional action recognition model is trained on non-obfuscated pairs $\{(x_i^*, y_i)\}_i$, it can easily exhibit degraded performances when tested on blurred video clips because of the gap between the non-obfuscated domain $\{x_i^*\}_i$ and the obfuscated domain $\{x_i\}_i$. This becomes particularly evident when the obfuscation technique is aggressive – a strategy which nevertheless is appealing to ensure high protection levels, for instance when obfuscating the full body. In the latter case, for instance, cues learned on people's appearance and motion can no more be employed when forcing the model to work in the obfuscated domain. The same applies, even though in a less threatening way, when the obfuscation strategy is less intrusive, *e.g.* when blurring faces only.

With the objective of reducing the extent of this domain gap, as a first strategy we consider training the video prediction backbone on obfuscated data, *i.e.* pairs $\{(x_i, y_i)\}_i$ (visually depicted in Figure 5.5, left). Intuitively, this will encourage the network to learn visual, motion, and contextual cues which can be extracted from blurred clips, rather than concentrating on features that will not be visible once the clip has been anonymized.

5.2.2 Exploiting Privileged Information and Knowledge Distillation

Beyond encouraging the network to learn proper features on anonymized data, as a second strategy we also investigate the exploitation of non-anonymized videos during the training phase. Since non-anonymized clips $\{x_i^*\}_i$ act as privileged information in this setting, we employ knowledge distillation to transfer the knowledge which can be learned by a non-privacy-preserving model to a completely anonymized model which is allowed to access x_i but not x_i^* (see Figure 5.5, middle).

In short, the approach is as follows: we first train a teacher network on non-anonymized pairs (x_i^*, y_i). Then we train a student network on the (x_i, y_i) pairs, using both hard labels y_i and soft labels from the teacher, with the latter still being computed from the original videos x_i^*. The student and the teacher networks always share the same underlying architecture.

Teacher training with privileged information

First, a video understanding backbone is trained on the original non-anonymized dataset to obtain a teacher network f_t. The teacher, therefore, is trained to solve the following optimization problem:

$$\arg\min_{\boldsymbol{W}} \frac{1}{n} \sum_{i=1}^{n} \ell(\sigma(f_t(x_i^*)), y_i), \tag{5.2}$$

where x_i^* is an input video clip with $x_i^* \in \mathbb{R}^{T \times H \times W \times 3}$, n is the number of training samples and $\boldsymbol{W}$ are the teacher weights. Denoting with Δ^c the set of c-dimensional probability vectors, σ instead represents the softmax operator $\sigma : \mathbb{R}^c \to \Delta^c$:

$$\sigma(z)_k = \frac{e^{z_k}}{\sum_{j=1}^{c} e^{z_j}} \tag{5.3}$$

for all $k \in \{1...c\}$, where c indicates the number of classes. Finally, $\ell : \Delta^c \times \Delta^c \to \mathbb{R}$ is the categorical cross-entropy loss function, *i.e.*

$$\ell(\hat{y}, y) = -\sum_{k=1}^{c} y_k \log \hat{y}_k. \tag{5.4}$$

Student training

We then train a student network f_s using anonymized video clips x_i. As our goal is not that of compressing the resulting network, but rather that of encouraging the transfer of privileged information, we employ the same architecture for teacher and student. Under this setting, we enhance the information available to the student network with the high-level representation of non-anonymized videos x_i^* from the teacher.

In particular, the student network is trained according to the following optimization problem:

$$\arg\min_{\boldsymbol{W}} \frac{1}{n} \sum_{i=1}^{n} \left[\ell(\sigma(f_s(x_i)), y_i) + \lambda \ell_d(\hat{s_i}, s_i) \right], \tag{5.5}$$

which aims to both identify the correct action label employing anonymized clips, and to distill [56] the teacher knowledge obtained from the privileged access to the corresponding non-anonymized clip. In the formula above, $\boldsymbol{W}$ indicates the student weights, ℓ_d a knowledge distillation loss, while $\hat{s_i}$ and s_i are the soft-predictions from the student and teacher model, respectively. These are computed as

$$\begin{aligned} \hat{s_i} &= \sigma(f_s(x_i)/T) \in \Delta^c \\ s_i &= \sigma(f_t(x_i^*)/T) \in \Delta^c, \end{aligned} \tag{5.6}$$

where the temperature parameter $T > 0$ smooths the probability distributions over classes from both the teacher and the student, and $\lambda > 0$ weights the strength of the knowledge transfer with respect to the cross-entropy loss applied on anonymized data and ground truth labels.

As knowledge distillation loss, we employ the Kullback-Leibler divergence between the soft-predictions $\hat{s}$ and s from the student and the teacher, respectively, which is defined as

$$\ell_d(\hat{s}, s) = -\sum_{k=1}^{c} s_k \log \left(\frac{\hat{s_k}}{s_k} \right). \tag{5.7}$$

As it can be observed, under this schema the original non-anonymized videos $\{x_i^*\}_i$ are only given as input to the teacher network, while the student can leverage anonymized clips and the high-level representation of the original videos from the teacher, as a form of privileged information. Although one could question that

the knowledge transferred from the teacher to the student represents a leakage of sensitive information, it shall be noted that in our formulation we only consider a logit-level distillation, and do not transfer activations from intermediate layers. Under this setting, it is practically unfeasible to retrieve sensible data from features that lack any spatio-temporal support. Nevertheless, even if some identity-related information would leak from this high-level representation, only the identities contained in training data would be affected. During inference, instead, the student network can classify obfuscated videos without any support from the teacher network, ensuring complete privacy preservation.

5.2.3 Relational Knowledge Distillation

Adopting a Knowledge Distillation schema allows a privacy-preserving action recognition model to access privileged information, and to mimic the feature extraction process of a traditional video model even in presence of severe blurring. Nevertheless, knowledge transfer only happens between corresponding pairs of obfuscated and non-obfuscated videos, *i.e.* between pairs $\{(x_i, x_i^*)\}_i$, ignoring contrastive relationships between different samples, which might be informative and enhance the knowledge transfer process.

To avoid the limit of transferring activations from corresponding samples only, as a third strategy we enrich the previously defined distillation approach by considering pairwise relations between samples in a mini-batch (visually depicted in Figure 5.5, right). Exploiting the correlation between instances during knowledge transfer has been recently studied [39, 106, 130]. In our case, we adopt a relational knowledge distillation strategy that takes inspiration from [106] and ensures that pairwise distances between logits are preserved when transferring knowledge from the teacher to the student. Intuitively, under this schema, we require that the representations obtained from a traditional action recognition model match those produced by the privacy-preserving model in terms of the structure of the embedding space they create.

In particular, we first obtain the logits for all the videos in a mini-batch, then we compute the normalized Euclidean distance between each pair of these logits:

$$\psi_D(p_i, p_j) = \frac{1}{\mu} \left\| p_i - p_j \right\|_2 , \tag{5.8}$$

where μ is obtained as the average Euclidean distance between pairs of representations inside the mini-batch. The pairwise Euclidean distance is computed for both the teacher and the student mini-batches, separately.

Finally, we match the distance corresponding to the same pair of samples for the teacher and the student (with the latter completely anonymized), and compute the relational knowledge distillation loss as follows:

$$\ell_r = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} l_\delta(\psi_D(\hat{p}_i, \hat{p}_j), \psi_D(p_i, p_j)) \tag{5.9}$$

where $\hat{p_i}$ and $\hat{p_j}$ are the logits computed by the student network on the video clips x_i and x_j, while p_i and p_j are the logits computed by the teacher network on the corresponding x_i^* and x_j^*. Here, l_δ is the Huber loss, which corresponds to the smooth L_1 loss in this case:

$$l_\delta(x, y) = \begin{cases} \frac{1}{2}(x-y)^2 & \text{for } |x-y| \leq 1 \\ |x-y| - \frac{1}{2} & \text{otherwise.} \end{cases} \tag{5.10}$$

As it can be seen, the criterion defined above encourages the distance between $f_s(x_i)$ and $f_s(x_j)$ to be as similar as possible to the distance between $f_t(x_i^*)$ and $f_t(x_j^*)$, without directly considering the representation of the sample. To include such a requirement in the final formulation as well, we use the relation knowledge distillation loss in conjunction with the Kullback-Leibler divergence loss (Eq. 5.7). The optimization problem we solve thus becomes:

$$\arg\min_{\boldsymbol{W}} \frac{1}{n} \sum_{i=1}^{n} [\ell(\sigma(f_s(x_i)), y_i) + \\ + \lambda \ell_d(\hat{s_i}, s_i) + \beta \ell_r(f_s(x_i), f_t(x_i^*))] \,, \tag{5.11}$$

where β determines the weight of the relational knowledge distillation with respect to the other terms.

5.3 Experimental results

In the following, we describe the experimental setting in which we analyze the three previously defined strategies for developing privacy-preserving video networks. To broaden the value of the analysis, we perform experiments on three popular video understanding backbones, namely ResNet(2+1)D-18, ResNet3D-18 [133], and SlowFast-R50 [36]. We also employ both the anonymization strategies described in Sec. 5.1, *i.e.* obfuscating faces and whole body masks. As our reference dataset, we employ Kinetics-400 [70].

5.3.1 Implementation details

For ResNet(2+1)D and ResNet3D, we follow the implementation and training recipe of the original paper [133]. Both networks work on 16 frames clips, where each of them is resized to a fixed size of 128×171 before being randomly cropped to 112×112 during training. During each epoch, we randomly sample three clips from each of the training videos and run the optimization process for a total of 45 epochs. To reduce training time, we employ synchronized SGD in a distributed training setting on different NVIDIA Tesla V100 GPUs. We employ a batch size of 1536 and a base learning rate of 0.96, which is linearly warmed-up during the first 10 epochs and then divided by a factor of 10 every 10 epochs.

For the SlowFast network, we employ ResNet50 [52] as the backbone. The overall network takes 64 consecutive frames as input, which are then fed with different temporal granularities to a "slow" and a "fast" pathway. The slow path takes 4 frames as input (which are sampled from the input clip using a temporal stride of 16), while the fast path samples 32 frames (sampled from the input clip using a temporal stride of 2). During training, we resize the shorter side of the video clip to a random value in the interval [256, 320] and preserving the aspect ratio. We then randomly crop the result to 224×224. As for ResNet(2+1)D and ResNet3D, we randomly sample three clips from each training video during one epoch. The training lasts for 45 epochs, using a total batch size of 2048. The base learning rate is set to 1.6, which is linearly warmed-up in the first five epochs, and cosine annealed after.

During inference, we uniformly sample 10 clips from the input video and average their predictions to obtain a video-level output. When using ResNet(2+1)D-18 and ResNet3D-18, we resize each clip to 128×171, and then center crop to 112×112. For SlowFast-R50, the shorter side of each frame is resized to 256, maintaining the aspect ratio, and then cropped in the center to a size of 256×256.

The values of the hyperparameters λ, β, and T are set to 12.500, 10, and 5, respectively, in all experiments.

5.3.2 Impact of anonymization in videos

We first assess the capabilities of standard video classification backbones when tested with anonymized data, to quantify their degradation in performance. To this end, we consider a standard ResNet(2+1)D-18 [133] trained on the full non-anonymized Kinetics-400, and measure its accuracy on both original and obfuscated videos from the validation set. As shown in Table 5.1, the top-1 accuracy

Model	Anonymization				top-1	top-5	$\Delta_{\text{top-1}}$
	Train		Test				
R(2+1)D	☐	none	☐	none	**69.2**	**88.1**	–
R(2+1)D	☐	none	◩	faces	67.2	86.3	**-2.9 %**
R(2+1)D	☐	none	■	full body	56.5	77.6	**-18.4 %**

Table 5.1: Performances of R(2+1)D [133] when tested on anonymized and non-anonymized videos.

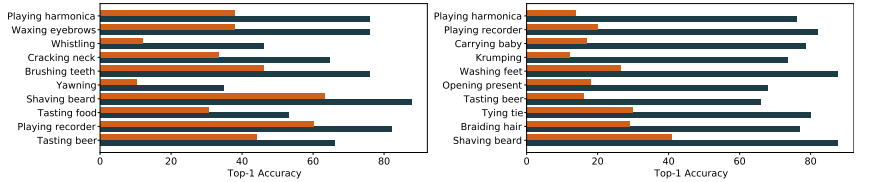


Figure 5.6: Performance degradation of a traditional action recognition model when obfuscating faces (R(2+1)D model, 69.2 vs 67.2 top-1 accuracy) on the left, and performance degradation of the same action recognition model when obfuscating body instances (69.2 vs 56.5 top-1 accuracy) on the right.

of the network on non-anonymized videos is 69.2 (first row). This decreases to 67.2 when testing on videos with obfuscated faces (second row), underlying that even this less intrusive anonymization strategy has a relevant impact on the effectiveness of the predictions. When applying full-body anonymization, the drop in accuracy becomes even larger: in this case, the model reaches only 56.5 top-1 accuracy, which corresponds to an 18.4% relative drop in accuracy (third row). The same degradation can be observed when considering a top-5 accuracy measure: from 88.1, the network degrades to 86.3 when obfuscating faces and to 77.6 when applying a full-body obfuscation.

Per-class analysis. To further investigate the reduction in performance, we conduct a class-based analysis. In Figure 5.6 we report the ten classes with the highest drop in top-1 accuracy when blurring faces or bodies (left and right, respectively). Here, blue bars indicate the per-class accuracy of the model when tested on non-anonymized data, while orange bars report the per-class accuracy of the same model when tested on anonymized data. As expected, when blurring faces, the actions with the largest accuracy drop involve the actor's face itself (*e.g. whistling*

Model	Anonymization				KD	RKD	top-1	top-5
	Train		Test					
R(2+1)D	□	none	□	none			69.2	88.1
R(2+1)D	□	none	□	none	✓		70.6	89.1
R(2+1)D	□	none	□	none	✓	✓	70.4	88.8
R(2+1)D	◩	faces	◩	faces			68.6	87.7
R(2+1)D	◩	faces	◩	faces	✓		70.3	88.8
R(2+1)D	◩	faces	◩	faces	✓	✓	**70.4**	**88.7**
R(2+1)D	■	full body	■	full body			65.7	85.5
R(2+1)D	■	full body	■	full body	✓	✓	**68.5**	**87.9**

Table 5.2: Performances of R(2+1)D [133] when trained using Knowledge Distillation (KD) and the relational KD criterion (RKD).

or *yawning*) or interactions between objects and the actor's face (*e.g*. *playing harmonica* or *brushing teeth*). On the contrary, we did not observe specific common points among the classes in the right histogram of Figure 5.6 (blurred body): however, we notice that all these classes are not strongly context-dependent, *i.e*. it would be hard to recognize these classes with the only help of the context.

5.3.3 Privacy preserving networks

Reducing the domain gap

As noticed, part of the aforementioned accuracy drop can be attributed to the domain gap between non-anonymized and anonymized videos, which becomes particularly evident when blurring whole bodies. For this reason, we experiment by both training and testing a ResNet(2+1)D-18 on obfuscated videos. Results are reported in Table 5.2: as it can be observed, this privacy-preserving model reaches a 68.6 top-1 accuracy when working with obfuscated faces, and 65.7 top-1 accuracy when trained and tested with obfuscated bodies.

Even though the performance gap with the original video prediction model has been greatly reduced, it shall be noted that it is still not-negligible: when compared to the non-privacy preserving counterpart, the aforementioned network still presents a 5,06% relative drop in top-1 accuracy and a 2,95% relative drop in top-5 accuracy when obfuscating bodies.

Applying Knowledge Distillation

We now turn to the application of the proposed Knowledge Distillation strategies. Before assessing their impact on the development of privacy-preserving networks, we investigate the role of self-distillation, *i.e*. that of applying the same distillation technique on the original, non-privacy preserving network. Self-distillation, indeed, has been demonstrated to be an effective strategy for improving performances in many vision tasks [102, 164, 166], and we therefore investigate its effect for fair comparison.

When a non privacy-preserving student ResNet(2+1)D-18 is trained via knowledge distillation using the same backbone as the teacher, it reaches a **70.6** top-1 accuracy on the Kinetics-400 validation set (Table 5.2, second row). In this setting, all sensible information are available to both the teacher and the student, quantifying to 1.4% the absolute top-1 accuracy gain given by KD. This can also be conceived as an upper bound for the anonymized models when using the same backbone. We further notice that in this non-anonymized setting, the addition of the relational KD brings no advantage, leading to a 70.4 top-1 accuracy (Table 5.2, third row).

Moving to the anonymized setting, *i.e*. when the student is trained and tested with obfuscated faces/bodies, we show how the privileged information from the teacher helps to abruptly reduce the gap with the non-anonymized backbone. When obfuscating faces, the KL loss alone brings a 70.3 top-1 accuracy on the anonymized Kinetics-400 validation set, which is further increased to **70.4** when using the relational KD (fifth and sixth row of Table 5.2). Hence, when only face information is removed, the gap with the non-anonymized counterpart is almost completely closed (70.4 vs 70.6).

When blurring whole bodies, instead, the student reaches **68.5** top-1 accuracy (Table 5.2, last row): as expected, totally removing actor-related features has a negative impact on performance. However, compared to training and testing on anonymized bodies without distillation (Table 5.2, penultimate row), privileged information from the teacher brings a 2.8% absolute improvement (65.7 vs 68.5).

Experimental evaluation on other backbones

To assess the generalization ability of the approach, we apply the same distillation schema with anonymized student and privileged teacher on other backbones. Table 5.3 shows the results for a ResNet3D-18 [133] and a SlowFast-R50 [36]. For each backbone, we report the accuracy of the teacher trained with cross-

Model	Anonymization				top-1	top-5
	Train		Test			
R3D	□	none	□	none	65.3	85.5
R3D	◩	faces	◩	faces	66.4	86.4
R3D	■	full body	■	full body	64.9	85.3
SlowFast	□	none	□	none	73.5	91.3
SlowFast	◩	faces	◩	faces	73.9	91.3
SlowFast	■	full body	■	full body	72.1	90.3

Table 5.3: Performances of R3D [133] and SlowFast [36] and of their privacy-preserving counterparts, trained using Knowledge Distillation and the relational KD criterion.

entropy on the original non-privacy-preserving data (first row), the accuracy of a student model trained with both KL and relational distillation losses on obfuscated faces (second row), and that of a student model trained with the same losses on obfuscated bodies (third row).

As it can be observed, removing face information has a minimal impact on the performances of ResNet3D-18 and SlowFast-R50, and the accuracy of the privacy-preserving networks can even exceed that of the teacher. The removal of body information, instead, is more challenging to handle. In this case, a small performance drop is observable, even if the performances of the privacy-preserving models remain close to those of the non-anonymized teachers.

Per-class analysis. Finally, in Figure 5.7 we present a per-class analysis between R(2+1)D models trained with the KL divergence and the relational distillation losses of Eq. 5.11. In both histograms, the ten classes with the highest gap are reported: the blue bars indicate the per-class top-1 accuracy of a student network trained and tested on non-anonymized videos (70.4 overall top-1 accuracy), while the orange bars show the per-class accuracy for an anonymized student model. In the left histogram, the student is trained and tested with obfuscated faces (70.4 overall top-1 accuracy), while in the right one it is trained and tested with obfuscated bodies (68.5 overall top-1 accuracy).

When comparing these per-class differences with those of Figure 5.6, we notice a strong reduction in the average gap. Specifically, when obfuscating faces, these classes are no longer related to actions involving the actor's face, and the accuracy loss in this top-10 analysis goes from a minimum of 8% (*applauding*) to a maximum of 16.3 (*yoga*). When blurring bodies, the maximum and minimum accuracy loss refer to *opening bottle* and *unboxing*, with 16% and 10.2% absolute

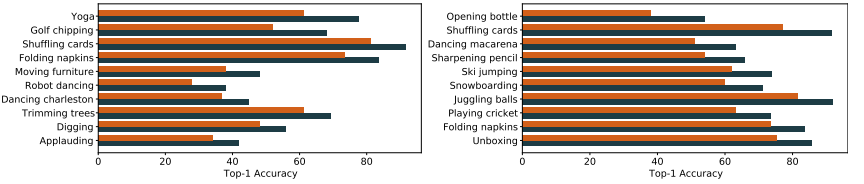


Figure 5.7: Performance difference between non privacy-preserving and privacy-preserving R(2+1)D student when obfuscating faces (70.4 vs 70.4 top-1 accuracy) on the left and when obfuscating bodies (70.4 vs 68.5 top-1 accuracy) on the right.

drop in accuracy, respectively.

Chapter 6

Efficiency in action recognition

As anticipated in the previous chapter, we now tackle the computational drawback of video processing. The research community has recently focused on developing more principled convolutional [157, 84, 133] and attentive [147, 153, 3] operators for handling time, showing great interest in novel and more accurate deep learning based video understanding solutions. Starting from models that adapt 2D CNNs to spatio-temporal volumes [131, 121], new architectural choices [36, 35] have gained considerable advances in terms of effectiveness and accuracy in the last few years.

However, being videos sequences of frames, the development of video models requires significant computational efforts, with a non-negligible impact on the cost of each experiment and, ultimately, on the speed of the research pace. Not only training video models can be slow and expensive, but also their deployment in production environments could be largely affected by the huge amount of parameters and floating-point operations. While the computational cost of researching novel architectures is well known and has been properly managed by the research community, the size and energy requirements of state-of-the-art networks still severely limit their applicability in production environments. Endowing any content sharing platform based on real-time video analysis with a state-of-the-art spatio-temporal

This chapter is related to publication "M. Tomei et al, A computational approach for progressive architecture shrinkage in action recognition, Softw. Pract. Exp. 2021". See Appendix B for details.

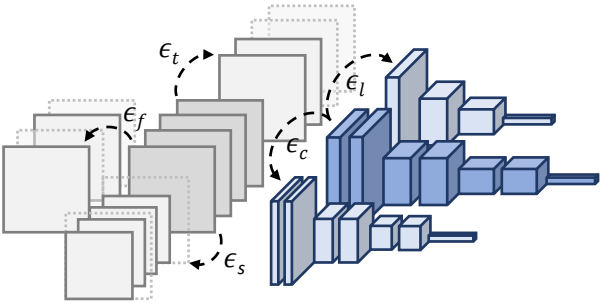


Figure 6.1: Conceptual overview of our approach. We progressively shrink a video understanding model by modifying its input and architectural hyperparameters through a set of reduction operators (in Figure, outlined with ϵ_*), minimizing the loss in accuracy during the overall process – so to obtain a smaller but effective model.

network, for example, would be almost unsustainable in the majority of cases.

In the attempt to make a step forward in the development of more sustainable spatio-temporal models, in this chapter we describe a generic strategy for turning any video model into a more efficient one, limiting the loss in accuracy caused by the reduction of computational demand.

Contributions. We look at the process of shrinking a spatio-temporal network as that of applying a sequence of "reduction" operations over the axes affecting its computational complexity and sustainability (Fig. 6.1). For instance, one could reduce the computational footprint by halving the number of channels of the initial architecture, and then reduce the spatial size of the input. In the same manner, one could gain a similar improvement in efficiency by increasing the temporal stride between the input frames and then halving the number of channels. While both these choices would increase the efficiency of the resulting network, surely they would have a different impact on the final accuracy computed over a given dataset. Our approach, named *Progressive Architecture Shrinkage* (PAS), iteratively shrinks a base network by selecting an optimal sequence of reduction operators, so to lower the computational complexity while limiting the loss in accuracy.

To maintain high accuracy levels after the application of each reduction operator, we employ a Knowledge Distillation paradigm that aims at preserving the

knowledge learned in the initial network. Further, we transfer knowledge between each reduction step by applying an adaptive fine-tuning strategy. The resulting approach is general enough to be applied to any video backbone and, after a sequence of reduction steps, produces a smaller network with a computational complexity of choice. Although the overall approach requires a high computational load, since at each iteration the reduced network needs to be re-trained, PAS achieves impressive results in finding a good trade-off in computational complexity and accuracy.

To validate the effectiveness of our approach, we perform experiments by shrinking two implementations of recently proposed backbones, *i.e*. R(2+1)D [133] and SlowFast [36], when training on the Kinetics-400 dataset [70]. Further, we also assess the capabilities of the reduced networks on the UFC101 [124] and HMDB51 [76] datasets.

6.1 An iterative approach for reducing computation

We propose a methodology that progressively reduces the computational needs of a given video architecture by modifying either its input size or architectural hyper-parameters while minimizing the impact of these modifications on the resulting accuracy of the network. Starting from an initial network, this is achieved following an iterative approach, requiring large scale and parallel computing during training, but providing a smaller model with far fewer resource requirements for testing in the end.

Given the initial network $\mathcal{B}$, at each iteration τ the procedure selects a reduction operator ϵ_τ that can alter either the input size or the architectural parameters of the current network, and which produces a reduced network $\mathcal{R}_\tau = \epsilon_\tau(\mathcal{R}_{\tau-1})$. The set of possible reduction operations is an hyper-parameter of the approach, which is described in the following section. To ease the choice of the best reduction to keep, they are devised in order to ensure a fixed computation reduction. The choice of the sequence of reduction operations is optimized by following a coordinate descent schema which aims at maximizing the trade-off between computational demands and accuracy. While the new reduced network is trained on the same dataset on which $\mathcal{B}$ was trained, we both distill the activations of the base model and apply a sequential fine-tuning strategy to limit the loss of accuracy. This sequential shrinkage of the network continues until a satisfactory computational complexity is reached. Experimental results will show that three iterations increase efficiency by 8 times, with limited loss in accuracy.

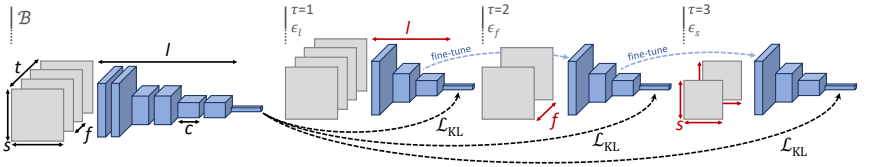


Figure 6.2: Example of a sequence of reduction operations obtained with Progressive Architecture Shrinkage (PAS). At each iteration, PAS selects a reduction operator ϵ which is applied over the current network. The latter is trained with a KD criterion with respect to the base network $\mathcal{B}$ and with an adaptive fine-tuning when possible.

6.1.1 Reduction Operations

The structure of a spatio-temporal convolutional network is defined by both the shape of its input, in terms of the number of frames and their spatial resolution, and architectural hyper-parameters, *e.g.* the number of layers in the network and their filters. In the following, we define a basic set of reduction operations that are used to sequentially modify the network's structure at each iteration. All these operations have an impact on the input or on the architecture. Since these belong to integer arithmetic, we omit rounding stages for clarity in the following.

- ϵ_s reduces the spatial resolution of the input by spatially downsampling the video clip by a factor γ_s. Given an input clip shape of $(\mathsf{T}, \mathsf{C}, \mathsf{H}, \mathsf{W})$, where T is the number of frames in the video clip, C the number of RGB channels for each frame, H the frame's height, and W the frame's width, this operation returns an output shape of $(\mathsf{T}, \mathsf{C}, \mathsf{H}/\gamma_s, \mathsf{W}/\gamma_s)$, where the two spatial axes have been downsampled through a resize operation.

- ϵ_t reduces the temporal length of the input, by cutting the input sequence up to a length proportional to γ_t. Given the same shape of the original clip, this operation returns an output shape of $(\mathsf{T}/\gamma_t, \mathsf{C}, \mathsf{H}, \mathsf{W})$, where the new tensor only contains the first T/γ_t frames of the original tensor.

- ϵ_f reduces the frame rate of the clip by increasing its temporal stride. Given an input clip shape of $(\mathsf{T}, \mathsf{C}, \mathsf{H}, \mathsf{W})$, this operations again returns an output shape of $(\mathsf{T}/\gamma_f, \mathsf{C}, \mathsf{H}, \mathsf{W})$, where the new tensor is obtained from the first one by sampling a frame every γ_f.

- ϵ_c reduces the number of output channels of all the convolutional layers of the network by a factor of γ_c. Given a convolutional layer with C filters, the application of this operation amounts to setting the number of filters of the same layer to C/γ_c.
- ϵ_l reduces the number of layers in the network by a factor of γ_l. As modern ResNet-like networks are organized as sequences of convolutional blocks, we implement this operation as a reduction of the number of layers inside each block – so that the resulting network maintains the original architectural choices while having fewer layers.

6.1.2 Progressive Architecture Shrinkage

The goal of progressive architecture shrinkage is to find the best sequence of reduction operations $(\epsilon_1^*, ..., \epsilon_T^*)$, with $\epsilon_i^* \in \{\epsilon_s, \epsilon_t, \epsilon_f, \epsilon_c, \epsilon_l\}$, to be applied to the base network $\mathcal{B}$ in order to reduce its computational cost and keep its effectiveness unaltered as much as possible. To jointly take into account these objectives, we define the quality of a network as the ratio between its accuracy and computational cost.

Following previous works on video recognition [35, 36, 132, 133, 147], we measure the efficiency of a network as the number of floating point operations it requires to process a single sample during the evaluation phase, and evaluate the effectiveness of the network on a given benchmark through its top-1 classification accuracy on the validation set.

Taking inspiration from previous works [35, 150] we optimize the shrinkage objective by applying a form of coordinate descent in the hyper-parameter space defined by the reduction axes. At each iteration, given the current network $\mathcal{R}_\tau$ (where $\mathcal{R}_0$ corresponds to the base network $\mathcal{B}$) we explore a number of hypotheses equal to the number of reduction axes, each of them obtained by applying a reduction operator to $\mathcal{R}_\tau$. We then select the hypothesis that maximizes the ratio between the reduction in computational cost and the reduction in accuracy. This amounts to selecting the reduction operator ϵ_τ^* that satisfies the following:

$$\epsilon_\tau^* = \arg\max_{\epsilon_j} \frac{C(\mathcal{B}) - C(\epsilon_j(\mathcal{R}_{\tau-1}))}{\text{Acc}(\mathcal{B}) - \text{Acc}(\epsilon_j(\mathcal{R}_{\tau-1}))}, \quad \epsilon_j \in \{\epsilon_s, \epsilon_t, \epsilon_f, \epsilon_c, \epsilon_l\} \tag{6.1}$$

where $C(\cdot)$ indicates the number of floating points operations required by a network to process a single sample, $\text{Acc}(\cdot)$ indicates the top-1 validation accuracy

of a network, and $\mathcal{R}_{\tau-1}$ is the reduced network obtained at the previous iteration, defined as

$$\mathcal{R}_{\tau-1} = \epsilon^*_{\tau-1}(\epsilon^*_{\tau-2}(...\epsilon^*_2(\epsilon^*_1(\mathcal{B})))) \tag{6.2}$$

being ϵ^*_i the reduction operation chosen at iteration i, and $\epsilon_j(\cdot)$ the application of a reduction operator over a network.

Exploring each hypothesis requires to re-train a new network, and each step of the coordinate descent procedure described above requires to independently train a number of networks equal to the number of reduction operators. As opposed to a recent work by Feichtenhofer *et al*. [35], which investigated the use of coordinate descent for progressively increasing the size of a model, in our case the size and computational cost of the models decrease at each iteration.

Applying a constant computational complexity scaling factor

To ensure that the steps taken in the coordinate descent approach are consistent along each direction, we design our reduction operations so to keep a constant complexity reduction factor between the application of two subsequent reduction operations whenever possible. Under the hypothesis that $C(\epsilon_j(\mathcal{R}_{\tau-1}))/C(\mathcal{R}_{\tau-1})$ is constant, the rule for selecting the best hypothesis (see Eq. 6.1) can be reduced to simply selecting the hypothesis with maximum accuracy, *i.e*.

$$\epsilon^*_\tau = \arg\max_{\epsilon_j} \text{Acc}(\epsilon_j(\mathcal{R}_{\tau-1})), \epsilon_j \in \{\epsilon_s, \epsilon_t, \epsilon_f, \epsilon_c, \epsilon_l\}. \tag{6.3}$$

In practice, we apply reduction operators which lead to a complexity reduction factor $C(\epsilon_j(\mathcal{R}_{\tau-1}))/C(\mathcal{R}_{\tau-1})$ roughly equal to 2 (*e.g*., halving the temporal resolution of a model usually leads to halving the FLOPs required by a model). It shall be noted, however, that the reduction in computational complexity depends on both the reduction operator and on the architecture of the current network. For instance, when the temporal resolution of the reduced model becomes lower than the total temporal downsampling factor of the network (determined by the overall stride in temporal convolutions), the temporal resolution of the activation maps in the network tail will become equal to 1. In this case, halving the temporal resolution would reduce the computational complexity by a factor ≤ 2. Only in these cases, we choose to maintain the reduction operators unaltered and opt for Eq. 6.1 for choosing the best reduction operation, while using Eq. 6.3 in all the other cases.

6.1.3 Training via Distilling the Knowledge

As the reader will have noticed, the base network $\mathcal{B}$ has not been involved in the optimization process until now, except for being employed as the architectural starting point for a sequence of progressively reduced networks. At each iteration τ, we want to keep the knowledge from $\mathcal{B}$ as much as possible, while cutting down the complexity of $\mathcal{R}_\tau$ with respect to the network produced at the previous iteration, $\mathcal{R}_{\tau-1}$. Therefore, when training a reduced network $\mathcal{R}_\tau$, we distill knowledge [56] from the base network $\mathcal{B}$ to the current reduced network. Knowledge Distillation [56] has recently emerged as a powerful technique to transfer knowledge from large models to smaller ones: these two roles are fulfilled by the base network $\mathcal{B}$ and the progressively reduced networks $\mathcal{R}_\tau$, respectively.

The process of transferring knowledge from the base network to a reduced one works as follows. The base network $\mathcal{B}$ is trained on a dataset D with a standard cross-entropy loss on each training sample, *i.e.*:

$$\mathcal{L}_{CE} = -\sum_{k=1}^{K} y_k \log\left(\frac{e^{\hat{p}_k}}{\sum_{j=1}^{K} e^{\hat{p}_j}}\right), \tag{6.4}$$

where K is the number of different classes, y represents the ground-truth one-hot vector of a sample and $\hat{p}$ represents the output logits from the base network. The same loss is applied when training a hypothesis of reduced network $\mathcal{R}_\tau$, using its own output logits in place of $\hat{p}$.

Besides employing a cross-entropy loss, which maximizes the probability of the correct labels, we train each reduced network hypothesis to minimize a Kullback-Leibler divergence loss with respect to the output probabilities of the base network. Formally, this is defined as

$$\mathcal{L}_{KL} = -\sum_{k=1}^{K} \hat{z}_k \log\left(\frac{z_k}{\hat{z}_k}\right), \tag{6.5}$$

where $\hat{z}_k$ represents the soft targets from the base network $\mathcal{B}$, and z_k indicates the normalized output from the reduced network hypothesis. Formally,

$$\hat{z}_k = \frac{e^{\hat{p}_k/t}}{\sum_{j=1}^{K} e^{\hat{p}_j/t}} \quad \text{and} \quad z_k = \frac{e^{p_k/t}}{\sum_{j=1}^{K} e^{p_j/t}}, \tag{6.6}$$

where the same softmax temperature $t > 1$ is applied to both probability distributions. Following [56], we also multiply the KL loss by t^2 when using both hard

and soft targets. The final objective used to train each reduced network hypothesis is a weighted sum of the cross-entropy and KL losses, as follows:

$$\mathcal{L}_S = \mathcal{L}_{CE} + \alpha t^2 \mathcal{L}_{KL}. \tag{6.7}$$

Besides distilling knowledge from the base network by employing the activations from the last layer, in our preliminary experiments we also explored with different Knowledge Distillation methods involving intermediate feature maps – *e.g.* by adopting a MSE loss or a partial L_2 distance together with a margin ReLU [53] between the Student's and the Teacher's activations, although without observing significant improvements. More details can be found in Sec. 6.2.5.

6.1.4 Adaptively fine-tuning from previous iterations

So far, the reduced network obtained at a given iteration, $\mathcal{R}_\tau$, has been trained from scratch on the target dataset, starting from random weights, and by distilling knowledge from the base model $\mathcal{B}$. While this training strategy clearly creates a link between the current reduced network and the base model, we also aim at creating a training dependency with the reduced model obtained at the previous iteration, $\mathcal{R}_{\tau-1}$. Being the latter the best model reached so far in terms of the complexity/accuracy trade-off, we expect its knowledge to be beneficial for the network hypotheses which are developed at the next stage. To this aim, we implement an adaptive fine-tuning strategy that aims at recovering the knowledge from the reduced models obtained at previous iterations. As shown in Fig. 6.2 with the blue dashed lines, at each iteration τ and before training the reduced hypotheses $\epsilon_j(\mathcal{R}_{\tau-1})$, $\epsilon_j \in \{\epsilon_s, \epsilon_t, \epsilon_f, \epsilon_c, \epsilon_l\}$, we initialize them with the weights of the reduced network from the previous iteration $\mathcal{R}_{\tau-1}$, instead of using random weights. This fine-tuning is not always straightforward, since architectural modifications are possible during the iterations. We apply the adaptive fine-tuning when all the weights of $\epsilon_j(\mathcal{R}_{\tau-1})$ and $\mathcal{R}_{\tau-1}$ have the same shape. This is verified only when applying an operator impacting the input shape, *i.e.* $\epsilon_s, \epsilon_t, \epsilon_f$. When a hypothesis requires to use an operator impacting the weights shape, instead, we do not apply the fine-tuning strategy and train from random weights. Future works will investigate the possibility of applying fine-tuning even in these cases: when the number of layers in the network is reduced, or the number of channels in each layer shrinks, one could initialize weights starting from a subset of those of the model obtained at the previous iteration.
Our progressive shrinkage mechanism is presented in Algorithm 1, assuming that Eq. 6.3 can be used in place of Eq. 6.1.

Algorithm 1: Progressive Architecture Shrinkage

Data: Pre-trained base network $\mathcal{B}$, reduction operations $\{\epsilon_s, \epsilon_t, \epsilon_f, \epsilon_c, \epsilon_l\}$, number of iterations I;
Result: Reduced networks $\mathcal{R}_\tau$, $\tau \in \{1, ..., I\}$;
$\mathcal{R}_0 \leftarrow \mathcal{B}$;
for $\tau := 1 \rightarrow I$ **do**
  TopAcc $= 0$;
  for ϵ_j *in* $\{\epsilon_s, \epsilon_t, \epsilon_f, \epsilon_c, \epsilon_l\}$ **do**
    if ϵ_j *in* $\{\epsilon_s, \epsilon_t, \epsilon_f\}$ **then**
      $\mathcal{R}_\tau^j \leftarrow \epsilon_j(\mathcal{R}_{\tau-1})$;
      Initialize $\mathcal{R}_\tau^j$ with weights from $\mathcal{R}_{\tau-1}$;
      Train $\mathcal{R}_\tau^j$ with $\mathcal{L}_S$;
    else
      $\mathcal{R}_\tau^j \leftarrow \epsilon_j(\mathcal{R}_{\tau-1})$;
      Initialize $\mathcal{R}_\tau^j$ with random weights;
      Train $\mathcal{R}_\tau^j$ with $\mathcal{L}_S$;
    end
    if $Acc(\mathcal{R}_\tau^j) > TopAcc$ **then**
      $\epsilon_\tau^* \leftarrow \epsilon_j$;
      $\mathcal{R}_\tau \leftarrow \mathcal{R}_\tau^j$;
      TopAcc $\leftarrow$ Acc$(\mathcal{R}_\tau^j)$;
    end
  end
end

6.2 Experimental results

6.2.1 Datasets

We adopt the **Kinetics-400** [70] dataset for training all the reduced models obtained with our architecture shrinkage approach. As already introduced in Sec. 5.1, the Kinetics-400 consists of approximately 240k training and 20k validation videos belonging to 400 different human action classes, with each class comprising at least 400 videos. Following a common procedure in literature, we report both the top-1 and top-5 classification accuracy on the validation set as a metric of effectiveness, and the number of FLOPs as a metric for computational cost. Since the standard practice for inference consists averaging class-probabilities for multiple spatio-temporal crops of the same video, we underline that the computational cost linearly increases with the number of crops adopted during inference.

We also evaluate the transfer capabilities of our reduced models by fine-tuning

them on **UCF101** [124] and **HMDB51** [76]. UCF101 consists of about 13k videos belonging to 101 different action classes, while HMDB51 has only about 6k videos split across 51 classes. They both provide three different splits for training and testing, and we report the average accuracy over these splits. In our preliminary experiments, we also tested our progressive architecture shrinkage technique when training from scratch on UCF101 and HMDB51: although our strategy still improved the computational performances, we observed strong overfitting. More details can be found in Sec. 6.2.5.

6.2.2 Implementation details

While in principle our algorithm could be applied to any video backbone, here we consider two recent spatio-temporal CNNs to showcase the effectiveness of our approach. Namely, we employ our PyTorch re-implementation of R(2+1)D-18 [133] and SlowFast-4×16-R50 [36] for all our experiments. Some differences (primarily in training strategies) between our implementation and the original ones can be found, mainly for computational reasons.

R(2+1)D-18. This backbone decomposes 3D convolutions into 2D spatial convolutions followed by 1D temporal ones. We adopt the 18-layers version of this backbone and train a base model $\mathcal{B}$ following the original implementation presented in Tran *et al*. [133]: during training, we resize video frames to 128×171 and apply random crop with size 112×112. Each input clip of the base network consists of 16 consecutive frames. In the reduced models, when downsampling the spatial size, we also downsample the crop size accordingly.

During each training epoch, we sample 3 clips per video from random temporal locations for temporal jittering. Synchronized SGD is adopted on 64 NVIDIA GPUs, with a total mini-batch size of 1536 (24 per GPU). The base learning rate is set to 0.96, with linear warm-up during the first 10 epochs. Afterwards, the learning rate is divided by 10 every 10 epochs, in both the base model and reduced models. When training a reduced model starting from pre-trained weights, *i.e*. when applying the adaptive fine-tuning presented in Sec. 6.1.4, the base learning rate is instead divided by 10. Training is always completed in 45 epochs. During inference, we use 112×112 center crops from 10 clips uniformly sampled from the video. Output probabilities of these 10 clips are averaged to obtain video-level prediction.

SlowFast-4×16-R50. SlowFast networks consist of two pathways, a Slow path operating at low frame-rate, and a Fast one operating at higher frame-rate and

employing fewer channels in convolutional layers. We consider a base SlowFast instantiation with a ResNet-50 backbone [52]. During the training of the SlowFast base network, the shorter side of the input video is resized to a random value in the interval [256, 320] and keeping the aspect ratio, then 224 $\times$ 224 clips are randomly cropped from the video. The raw clips length is set to 64 frames, and the Slow path samples 4 frames with stride 16 from each clip, while the Fast path samples 32 frames with stride 2 from each clip [36]. In reduced models, we downsample input and temporal sizes accordingly.

As with R(2+1)D, 3 clips are randomly sampled from a video for an epoch. SGD on 128 GPUs is adopted, with a total mini-batch size of 2048 (16 per GPU). The base learning rate is set to 1.6, with linear warm-up during the first 5 epochs and cosine annealing after. Also in this case, the base learning rate is divided by 10 when training a reduced model with adaptive fine-tuning. Again, 45 epochs are performed on both the base and reduced models to lighten the computational requirements, while the original SlowFast implementation [36] suggested 256 epochs without temporal jittering. In inference, 256 $\times$ 256 center crops are extracted from 10 uniformly sampled clips (instead of 30 [36]), and their softmax scores are averaged to obtain video-level prediction.

Hyper-parameters tuning. Our approach requires different hyper-parameters to be tuned. The reduction operations could be arbitrarily chosen, and may depend on the task at hand. For video understanding, the easiest choice involves spatio-temporal resolution, network layers, and layer filters. The number of iterations should also be tuned depending on the resource requirements, but we observe that three to five iterations are enough for obtaining a good accuracy/computation trade-off (we fix the maximum number of iterations of the coordinate descent to 5). Moreover, the constant reduction factor at each iteration has been fixed to 2: a smaller factor could result in better trade-offs, however with more iterations to be performed. In order to roughly halve FLOPs, we set $\gamma_s = 1.4$, $\gamma_t = 2$, $\gamma_f = 2$, $\gamma_c = 1.4$ and $\gamma_l = 2$. For γ_l, we uniformly halve the layers in each residual block. The t value in Eq. 6.6 is set to 5, while the α value in Eq. 6.7 is 500. For both R(2+1)D-18 and SlowFast-4$\times$16-R50, we used a momentum of 0.9 and a weight decay of 10^{-4}. A dropout of 0.5 has been applied before the final classification layer when using the SlowFast backbone. Finally, since SlowFast consists of two paths which sample frames differently from the input video, we clarify how we handle the two sampling strategies when reducing the temporal resolution or the frame rate. Specifically, we first apply the chosen reduction operations to the input clip and then allow each path to sample frames from the reduced input according to its sampling strategy.

Model	top-1	top-5	GFLOPs
R(2+1)D-18 [133]	69.2	88.1	40.8 × 10
R(2+1)D-18-PAS-1	69.5	88.7	20.4 × 10
R(2+1)D-18-PAS-2	68.8	88.1	10.2 × 10
R(2+1)D-18-PAS-3	67.5	87.0	5.4 × 10
R(2+1)D-18-PAS-4	66.0	86.3	3.2 × 10
R(2+1)D-18-PAS-5	63.6	84.6	2.5 × 10
SlowFast-4×16-R50 [36]	73.5	91.3	36.6 × 10
SlowFast-4×16-R50-PAS-1	73.8	91.7	18.9 × 10
SlowFast-4×16-R50-PAS-2	73.4	91.1	10.2 × 10
SlowFast-4×16-R50-PAS-3	71.7	90.3	5.1 × 10
SlowFast-4×16-R50-PAS-4	69.8	88.9	2.9 × 10
SlowFast-4×16-R50-PAS-5	67.0	87.3	1.4 × 10

Table 6.1: Top-1 and top-5 accuracy, together with inference cost in terms of GFLOPs per view × adopted views, on Kinetics-400 validation set. Progressively reduced models obtained from our PAS algorithm are reported, starting from a R(2+1)D-18 and a SlowFast-4×16-R50 backbone, respectively.

6.2.3 Main Results

R(2+1)D-18 results. We start presenting the results obtained using the R(2+1)D-18 [133] backbone on the Kinetics-400 dataset. The upper side of Table 6.1 reports the top-1 and top-5 accuracy of the reduced models obtained by our progressive architecture shrinkage approach, as well as the number of GFLOPs required by each architecture to process a single view, multiplied by the number of views adopted during inference for a given video. In the Table, R(2+1)D-18-PAS-x indicates the reduced model obtained at iteration x. As it can be observed, the total drop in accuracy from the base network (R(2+1)D-18) to the reduced model obtained in the last iteration (R(2+1)D-18-PAS-5) is **5.6%** (from 69.2 to 63.6), while the number of GFLOPs required to process each view is reduced from **40.8 to 2.5**. It is also worth noting that the reduced model obtained at the first iteration (R(2+1)D-18-PAS-1) has a slightly increased accuracy compared to the base network (69.5 versus 69.2), despite requiring half of its GFLOPs per view (40.8 vs 20.4).

Fig. 6.3 shows in detail the overall procedure and the performances of all the hypotheses obtained at each iteration. The red dashed line highlights the best performing hypothesis selected at each iteration. Specifically, the sequence of reduction operations which has been chosen by the coordinate descent is the

Model	w/ PAS		w/o PAS	
	top-1	top-5	top-1	top-5
R(2+1)D-18-PAS-1	69.5	88.7	67.9	87.5
R(2+1)D-18-PAS-2	68.8	88.1	65.0	85.4
R(2+1)D-18-PAS-3	67.5	87.0	61.1	83.1
R(2+1)D-18-PAS-4	66.0	86.3	60.7	82.5
R(2+1)D-18-PAS-5	63.6	84.6	59.5	81.3
SlowFast-4×16-R50-PAS-1	73.8	91.7	73.1	91.0
SlowFast-4×16-R50-PAS-2	73.4	91.1	70.5	89.5
SlowFast-4×16-R50-PAS-3	71.7	90.3	68.1	88.4
SlowFast-4×16-R50-PAS-4	69.8	88.9	66.0	86.8
SlowFast-4×16-R50-PAS-5	67.0	87.3	61.2	83.6

Table 6.2: Performance comparison between models with exactly the same architecture, trained with or without our PAS strategy, on the Kinetics-400 validation set.

following: $[\epsilon_c, \epsilon_f, \epsilon_t, \epsilon_t, \epsilon_t]$, so the procedure firstly selects to reduce the number of channels, then the frame rate of the input clip, and then its temporal length (three times). Please note that different reductions may lead to the same top-1 accuracy, as happens at iteration 5 for ϵ_t and ϵ_f: to choose the best operation in these cases, we adopt the top-5 accuracy as the performance metric. The accuracy obtained by the other hypotheses, on which the remaining reductions were applied, is also reported at each step.

The blue dashed line in Fig. 6.3 represents, at each iteration, the accuracy of a model with the same architecture as the best performing hypothesis, but trained without using the Knowledge Distillation and the adaptive fine-tuning strategy. This is further detailed in the top part of Table 6.2, where we report the advantage of progressively reducing R(2+1)D-18 models through our PAS algorithm, compared to a standard KD-free and cross-entropy based training from scratch. At the third iteration, for instance, after the reductions $[\epsilon_c, \epsilon_f, \epsilon_t]$, the proposed PAS provides a gain of 6.4% top-1 accuracy (67.5 vs 61.1).

SlowFast-4×16-R50 results. We also present results using the more recent SlowFast-4×16-R50 [36] network. The lower side of Table 6.1 shows the accuracy and GFLOPs of a progressively reduced SlowFast instantiation using our PAS strategy. Accuracy drops from 73.5 (base network) to 67 (SlowFast-4×16-R50-PAS-5) in the sequence of iterations, while the number of GFLOPs per view is reduced from 36.6 to 1.4. Again, SlowFast-4×16-R50-PAS-1 improves

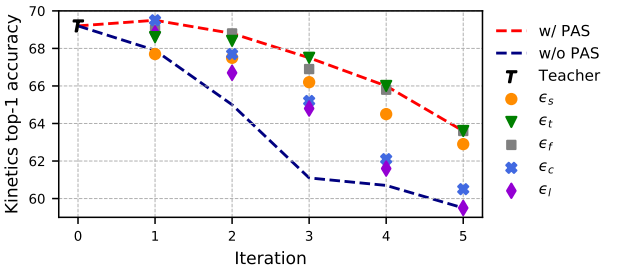


Figure 6.3: PAS strategy over 5 iterations applied to a R(2+1)D-18 (base network). Different markers represent the performance of different reduction operations. The red dashed line follows the best performing models, while the blue one shows what happens if the best model is trained without our PAS algorithm.

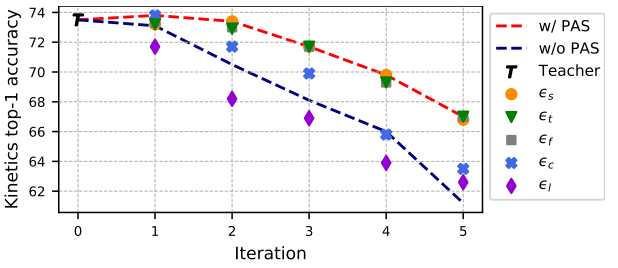


Figure 6.4: PAS strategy over 5 iterations applied to a SlowFast-4×16-R50 (base network).

the performance of the base model from 73.5 to 73.8, while cutting down the computational complexity from 36.6 to 18.9 GFLOPs per view.

Fig. 6.4 depicts the PAS procedure for this backbone, using the same notation as in Fig. 6.3. The best sequence of reduction operations here is $[\epsilon_c, \epsilon_s, \epsilon_t, \epsilon_s, \epsilon_f]$. Finally, in Table 6.2 bottom, we show the comparison between PAS-based SlowFast networks and their PAS-free counterparts at each iteration of our algorithm, as already observed for R(2+1)D-18. At iteration 5, the gap reaches its maximum value, with an absolute top-1 accuracy gain of 5.8%.

Observations

It is important to underline how the sequence of reduction operations chosen by our PAS method reflects which axes should be reduced in order to lower the computational requirements without compromising accuracy: for R(2+1)D-18, 4 out of 5 reductions involve time-related axes. This means that, for the given initial input shape ($16 \times 3 \times 112 \times 112$) defined by the authors [133], the temporal dimension is the most redundant. SlowFast networks rely on stronger temporal modeling: ϵ_t and ϵ_f are indeed chosen only once, respectively. The input spatial resolution of 256×256 has some redundancy, and ϵ_s is chosen twice. Moreover, Figures 6.3 and 6.4 share a common trend: going on with iterations, ϵ_c and ϵ_l degrade performance much more compared to ϵ_s, ϵ_t and ϵ_f. This gap confirms the usefulness of loading weights from previous iterations for input-related reductions, as explained in Sec. 6.1.4.

Another important aspect concerns the application of one reduction operation at a time, which may be sub optimal. Alternatives for reducing training resources and for faster shrinkage may consist in applying multiple reductions at each step (*e.g.* reductions operations affecting the same axes, like temporal resolution and frame rate). However, it should be noted that after obtaining a given reduction sequence for a model, similar networks probably benefit from the same sequence. For instance, this avoids performing again the coordinate descent for ResNet-like architectures. This together with the opportunity to search for the optimal set of operations to be applied at once, will be further studied in future works.

Comparison with other methods

In order to directly compare the models obtained in the succession of PAS iterations with other well-known models, we present the performance of several existing methods on the Kinetics-400 validation set in Table 6.3. For each method, the top-1 and top-5 accuracy are shown, along with pretraining strategy and inference GFLOPs. We report only the best performing model for the adopted backbones, obtained through PAS, and achieving state-of-the-art computation-accuracy trade-offs. Differently from X3D models [35], which exploit an expansion algorithm starting from a pre-defined architecture, the performance of our models highly depends on the base network capabilities: in this sense PAS differs from other strategies aiming to provide the highest possible accuracy. We believe that the main advantage of the PAS algorithm consists in its generalization ability: one can choose an existing strong network (based on some requirements) and apply PAS

Model	pretrain	top-1	top-5	GFLOPs
I3D [18]	ImNet	71.1	90.3	108 × N/A
Two-stream I3D [18]	ImNet	75.7	92.0	216 × N/A
Nonlocal R101 [147]	ImNet	77.7	93.3	359 × 30
Nonlocal R50 [147]	ImNet	76.5	92.6	282 × 30
TSM R50 [84]	ImNet	74.7	N/A	65.0 × 10
MF-Net [21]	ImNet	72.8	90.4	11.1 × 50
Two-stream I3D [18]	None	71.6	90.0	216 × N/A
ip-CSN-152 [132]	None	77.8	92.8	109 × 30
Oct-I3D+NL [20]	None	75.7	N/A	28.9 × 30
R(2+1)D-18 [133]	None	69.2	88.1	40.8 × 10
SlowFast-4×16-R50 [36]	None	73.5	91.3	36.6 × 10
R(2+1)D-18-PAS-1	None	**69.5**	**88.7**	**20.4 × 10**
SlowFast-4×16-R50-PAS-1	None	**73.8**	**91.7**	**18.9 × 10**

Table 6.3: Comparison with existing methods on Kinetics-400 validation set in terms of top-1 and top-5 accuracy. Inference cost is reported in terms of GFLOPs per view × adopted views. ImNet stands for ImageNet. PAS is limited by the performances of the base network, but its effectiveness is confirmed on two different well-known backbones.

on it to reduce its computation while maintaining performances almost unaltered. The ability to limit the accuracy loss when reducing computation is confirmed on two different backbones, with different designs and performances, so PAS could potentially be applied to X3D models, too.

Ablation experiments

PAS leverages two key techniques to avoid an abrupt decay in performance of progressively reduced models, namely Knowledge Distillation and fine-tuning from previous iterations. In order to quantify the role of these two solutions, we present ablation experiments in Table 6.4 for iteration 3 of PAS, applied to R(2+1)D-18. Here we first remove the Knowledge Distillation from the Student training process, but we load pre-trained weights from R(2+1)D-18-PAS-2. Then, we restore KD and train another Student from scratch, *i.e.* with randomly-initialized weights. In both cases, accuracy drops compared to our full solution. We explore only ϵ_s, ϵ_t and ϵ_f reductions for ablation purposes, since fine-tuning is unfeasible when applying ϵ_c and ϵ_l, while applying KD but not fine-tuning is exactly what we propose for ϵ_c and ϵ_l.

Model	KD	fine-tuning	ϵ_s	ϵ_t	ϵ_f
R(2+1)D-18-PAS-3	✓	✓	**66.2**	**67.5**	**66.9**
R(2+1)D-18-PAS-3	✗	✓	65.6	66.3	65.4
R(2+1)D-18-PAS-3	✓	✗	64.5	65.7	65.3

Table 6.4: Performance obtained when removing Knowledge Distillation or fine-tuning from previous iteration.

Metric	R(2+1)D	PAS-1	PAS-3	PAS-5
Avg GPU Util	39.1	20.8	12.7	7.7
Avg GPU Mem	11.3	6.1	3.2	1.0
Metric	SlowFast	PAS-1	PAS-3	PAS-5
Avg GPU Util	26.7	28.6	17.2	8.2
Avg GPU Mem	9.7	7.7	3.8	0.8

Table 6.5: Computational analysis of PAS-based models. Values are expressed as a percentage.

Resource utilization

Here we investigate the advantage of a model shrinked by PAS, in terms of both GPU utilization and memory consumption. Table 6.5 shows the average percentage utilization of CUDA cores and the average percentage of GPU memory consumption for a base backbone and PAS models. We used 4 NVIDIA V100 GPUs, each with 32 GB of memory. Values are collected for each GPU at every iteration over the whole Kinetics-400 validation set, with a batch size of 1 clip per GPU. Finally, we averaged the obtained measures over validation iterations and over GPUs. PAS allows resource saving while ensuring limited accuracy loss.

Computational capacity used during experiments

Our algorithm employs significant computational resources to find the best sequence of reduction operations. Nevertheless, after exploring different network hypotheses, it provides a compressed model with a complexity/accuracy trade-off of choice. In this section, we report some quantitative details about the number of GPU-hours required when performing five iterations with PAS. With the implementation details presented in Sec. 6.2.2, training each network hypothesis required about 15-16 hours for both R(2+1)D-18 and SlowFast-4$\times$16-R50. Using

Model	pretrain	UCF	HMDB	GFLOPs
R(2+1)D-18 [133]	K400	**94.4**	70.0	40.8 × 10
R(2+1)D-18-PAS-1	K400	94.0	**71.0**	20.4 × 10
R(2+1)D-18-PAS-2	K400	93.5	68.1	10.2 × 10
R(2+1)D-18-PAS-3	K400	91.6	63.3	5.4 × 10
R(2+1)D-18-PAS-4	K400	89.8	58.9	3.2 × 10
R(2+1)D-18-PAS-5	K400	87.0	54.0	2.5 × 10
SlowFast-4×16-R50 [36]	K400	95.0	-	36.6 × 10
SlowFast-4×16-R50-PAS-1	K400	**95.3**	-	18.9 × 10
SlowFast-4×16-R50-PAS-2	K400	95.1	-	10.2 × 10
SlowFast-4×16-R50-PAS-3	K400	94.0	70.4	5.1 × 10
SlowFast-4×16-R50-PAS-4	K400	92.8	68.3	2.9 × 10
SlowFast-4×16-R50-PAS-5	K400	90.4	63.3	1.4 × 10

Table 6.6: Transfer ability of PAS models on UCF101 and HMDB51 datasets. K400 stands for Kinetics-400.

64 GPUs for a single R(2+1)D-18 experiment, we employed ∼25.000 GPU-hours for training 25 network hypotheses (5 iterations × 5 reduction operators). For SlowFast-4×16-R50 training, we used 128 synchronized GPUs and employed a total number of ∼50.000 GPU-hours. In all our experiments we used distributed training on nodes with 4 NVIDIA V100 GPUs each, distributing the computation across 16 and 32 nodes, respectively, when employing the R(2+1)D-18 backbone and the SlowFast-4×16-R50 backbone. All the training and evaluations have been performed on the CINECA Marconi100 accelerated cluster, which consists of 980 nodes, and is ranked 18^{th} in the top500[1] ranking of the 500 most powerful commercially available computer systems in the world.

6.2.4 Transfer capabilities

In this section, we assess the transfer capabilities of PAS-generated models trained on the Kinetics-400 dataset, by fine-tuning them on UCF101 and HMDB51. When fine-tuning a model, the base learning rate is divided by 10, while all the other implementation details remain the same, as presented in Sec. 6.2.2. The PAS strategy is not adopted again in these experiments: we simply fine-tune available PAS models trained on the Kinetics-400 dataset, to verify if the transfer capabilities are maintained in the sequence of PAS iterations.

[1] https://top500.org/

Table 6.6 shows the top-1 classification accuracy when fine-tuning all the models obtained with our PAS strategy. As it can be seen, the transfer capability is preserved for both datasets: R(2+1)D-18-PAS-1 exceeds the performance of the base network on HMDB51, while SlowFast-4×16-R50-PAS-1 and SlowFast-4×16-R50-PAS-2 exceed the accuracy of the base network on UCF101, despite being much lighter. The first three SlowFast models in Table 6.6 are not evaluated on HMDB51, since many videos last between 1 and 2 seconds given a fixed frame-rate of 30 fps, which makes unfeasible to sample 64 frames (corresponding to the base temporal resolution for SlowFast networks). Being the chosen reductions sequence $[\epsilon_c, \epsilon_s, \epsilon_t, \epsilon_s, \epsilon_f]$ for SlowFast, the input temporal resolution is reduced to 32 in the third iteration, which allows us to report the accuracy of the last three models on HMDB51, too.

6.2.5 Additional experiments

Distilling knowledge from intermediate feature maps. In Table 6.7, we investigate the usage of a Knowledge Distillation loss employing the activations from intermediate layers. Specifically, we build an additional loss between the activations of the base network and of each network hypothesis, which is applied after each residual block of the backbone. We test the usage of both an L_2 loss and the margin ReLU-based approach presented by Heo *et al*. [53]. Table 6.7 shows the role of distilling knowledge from intermediate feature maps during the first iteration of PAS. As it can be observed, the introduction of an additional loss on intermediate layers does not increase the top-1 accuracy. In the final formulation of PAS, we only employ network logits for knowledge distillation.

Role of Batch Normalization. An important factor to consider when distilling the knowledge from the base network $\mathcal{B}$ to reduced models is the role of batch-normalization. As reported by Heo *et al*. [53], Batch-norm layers should behave in the same way in the teacher and in the student (*i.e*., they should be both in training mode or in evaluation mode). For this reason, our base model is set to training mode when computing its logits for Knowledge Distillation, since the students are in training mode, too. This ensures that the features from both the teacher and the student are normalized in the same way. Table 6.8 shows the advantage of distilling the knowledge from an R(2+1)D-18 base model in training mode with respect to evaluation mode. Top-1 accuracy on Kinetics-400 is reported for the first iteration of PAS and for all reduction operations.

Training PAS on smaller datasets without Kinetics-400 pretraining. As an-

Initial model	Red. Op.	Logits only	+MSE	+mReLU [53]
R(2+1)D-18	ϵ_l	68.8	68.1	68.0

Table 6.7: Performance when using different KD approaches on intermediate activations.

			Reduction operator				
Initial model	$\mathcal{B}$ train	$\mathcal{B}$ eval	ϵ_s	ϵ_t	ϵ_f	ϵ_c	ϵ_l
R(2+1)D-18	✓		67.7	68.6	69.0	69.5	68.8
R(2+1)D-18		✓	67.5	68.4	68.8	69.0	68.2

Table 6.8: Performance gain when setting the base model $\mathcal{B}$ in training mode with respect to evaluation mode.

ticipated in Sec. 6.2.1, we also trained PAS on UCF101 without employing a Kinetics-400 pre-training. Table 6.9 shows the top-1 accuracy of a progressively reduced R(2+1)D-18 model. For simplicity, we only report the top-1 accuracy obtained on the first split of UCF101. The performance of the base model $\mathcal{B}$ is 62.3 top-1 accuracy. For each PAS iteration, the accuracy obtained by applying each reduction operator is reported, along with the accuracy of the best reduced model trained from scratch without any knowledge distillation (last column).

In the first iteration, while the performance drop caused by ϵ_s, ϵ_t and ϵ_f is comparable with that observed in Kinetics-400, the same does not hold for ϵ_c and ϵ_l. Specifically, using PAS for reducing the number of channels or the number of layers increases the top-1 accuracy by 5.0% and 3.2%, respectively. Looking at the second row of Table 6.9, last column, it is clear that the accuracy gain is not completely due to PAS: when reducing the number of channels (best-performing reduction operator in the first iteration) and training the reduced model without PAS, accuracy still increases with respect to the base model (64.6 *vs*. 62.3), even with 50% fewer FLOPs. This highlights that the model capacity of R(2+1)D-18 is excessive for UCF101, which leads to strong overfitting. The same is visible in the following iteration. Despite overfitting, we notice that PAS still plays a key role in improving performance while reducing the number of FLOPs. From the third iteration on, reduced models trained without PAS start to get worse, while PAS-based training still ensures a minimal accuracy loss.

SlowFast-4×16-R50 Network architecture. For reference, in Table 6.10 we report the detailed architecture of the base network and of the reduced hypotheses

Initial model	Reduction operator ϵ_s	ϵ_t	ϵ_f	ϵ_c	ϵ_l	w/o PAS
R(2+1)D-18	-	-	-	-	-	62.3
R(2+1)D-18-PAS-1	60.5	62.0	61.5	**67.3**	65.5	64.6
R(2+1)D-18-PAS-2	63.8	66.3	66.2	**66.7**	66.5	64.0
R(2+1)D-18-PAS-3	63.5	65.4	**65.8**	65.3	65.1	60.3
R(2+1)D-18-PAS-4	62.8	**65.3**	64.1	63.0	63.5	59.3
R(2+1)D-18-PAS-5	62.2	63.7	**63.9**	61.8	62.2	57.8

Table 6.9: PAS applied on the UCF101 dataset with a R(2+1)D-18 backbone.

used in all our experiments with the SlowFast-4×16-R50 backbone. The architecture is reported as a function of the total reduction factors applied on spatial resolution, temporal length, frame rate, number of channels and layers (denoted as $\bar{\gamma}_s$, $\bar{\gamma}_t$, $\bar{\gamma}_f$, $\bar{\gamma}_c$, $\bar{\gamma}_l$) as a result of applying a sequence of reduction operators. The base network $\mathcal{B}$ corresponds to the configuration where all $\bar{\gamma}_*$ are equal to 1, and is identical to the one proposed by Feichtenhofer *et al*. [36]. At each iteration τ, a chosen reduction operation ϵ_j can increase the corresponding reduction factor $\bar{\gamma}_j$ by a factor of γ_j (see Sec. 6.1.1), and modifies the network architecture according to Table 6.10.

stage	*Slow* pathway	*Fast* pathway	output sizes $T \times S^2$
raw clip	-	-	$64/(\bar{\gamma}_t\bar{\gamma}_f) \times (224/\bar{\gamma}_s)^2$
data layer	stride 16, 1^2	stride 2, 1^2	*Slow* : $4/(\bar{\gamma}_t\bar{\gamma}_f) \times (224/\bar{\gamma}_s)^2$ *Fast* : $32/(\bar{\gamma}_t\bar{\gamma}_f) \times (224/\bar{\gamma}_s)^2$
conv_1	1×7^2, $64/\bar{\gamma}_c$ stride 1, 2^2	5×7^2, $8/\bar{\gamma}_c$ stride 1, 2^2	*Slow* : $4/(\bar{\gamma}_t\bar{\gamma}_f) \times (112/\bar{\gamma}_s)^2$ *Fast* : $32/(\bar{\gamma}_t\bar{\gamma}_f) \times (112/\bar{\gamma}_s)^2$
pool_1	1×3^2 max stride 1, 2^2	1×3^2 max stride 1, 2^2	*Slow* : $4/(\bar{\gamma}_t\bar{\gamma}_f) \times (56/\bar{\gamma}_s)^2$ *Fast* : $32/(\bar{\gamma}_t\bar{\gamma}_f) \times (56/\bar{\gamma}_s)^2$
res_2	$\begin{bmatrix} 1 \times 1^2, 64/\bar{\gamma}_c \\ 1 \times 3^2, 64/\bar{\gamma}_c \\ 1 \times 1^2, 256/\bar{\gamma}_c \end{bmatrix} \times 3/\bar{\gamma}_l$	$\begin{bmatrix} 3 \times 1^2, 8/\bar{\gamma}_c \\ 1 \times 3^2, 8/\bar{\gamma}_c \\ 1 \times 1^2, 32/\bar{\gamma}_c \end{bmatrix} \times 3/\bar{\gamma}_l$	*Slow* : $4/(\bar{\gamma}_t\bar{\gamma}_f) \times (56/\bar{\gamma}_s)^2$ *Fast* : $32/(\bar{\gamma}_t\bar{\gamma}_f) \times (56/\bar{\gamma}_s)^2$
res_3	$\begin{bmatrix} 1 \times 1^2, 128/\bar{\gamma}_c \\ 1 \times 3^2, 128/\bar{\gamma}_c \\ 1 \times 1^2, 512/\bar{\gamma}_c \end{bmatrix} \times 4/\bar{\gamma}_l$	$\begin{bmatrix} 3 \times 1^2, 16/\bar{\gamma}_c \\ 1 \times 3^2, 16/\bar{\gamma}_c \\ 1 \times 1^2, 64/\bar{\gamma}_c \end{bmatrix} \times 4/\bar{\gamma}_l$	*Slow* : $4/(\bar{\gamma}_t\bar{\gamma}_f) \times (28/\bar{\gamma}_s)^2$ *Fast* : $32/(\bar{\gamma}_t\bar{\gamma}_f) \times (28/\bar{\gamma}_s)^2$
res_4	$\begin{bmatrix} 3 \times 1^2, 256/\bar{\gamma}_c \\ 1 \times 3^2, 256/\bar{\gamma}_c \\ 1 \times 1^2, 1024/\bar{\gamma}_c \end{bmatrix} \times 6/\bar{\gamma}_l$	$\begin{bmatrix} 3 \times 1^2, 32/\bar{\gamma}_c \\ 1 \times 3^2, 32/\bar{\gamma}_c \\ 1 \times 1^2, 128/\bar{\gamma}_c \end{bmatrix} \times 6/\bar{\gamma}_l$	*Slow* : $4/(\bar{\gamma}_t\bar{\gamma}_f) \times (14/\bar{\gamma}_s)^2$ *Fast* : $32/(\bar{\gamma}_t\bar{\gamma}_f) \times (14/\bar{\gamma}_s)^2$
res_5	$\begin{bmatrix} 3 \times 1^2, 512/\bar{\gamma}_c \\ 1 \times 3^2, 512/\bar{\gamma}_c \\ 1 \times 1^2, 2048/\bar{\gamma}_c \end{bmatrix} \times 3/\bar{\gamma}_l$	$\begin{bmatrix} 3 \times 1^2, 64/\bar{\gamma}_c \\ 1 \times 3^2, 64/\bar{\gamma}_c \\ 1 \times 1^2, 256/\bar{\gamma}_c \end{bmatrix} \times 3/\bar{\gamma}_l$	*Slow* : $4/(\bar{\gamma}_t\bar{\gamma}_f) \times (7/\bar{\gamma}_s)^2$ *Fast* : $32/(\bar{\gamma}_t\bar{\gamma}_f) \times (7/\bar{\gamma}_s)^2$
global average pool, fc			# classes

Table 6.10: Architecture of the base model and of reduced hypotheses based on SlowFast-4×16-R50, as a function of the total reduction factor applied on spatial resolution, temporal length, frame rate, number of channels and number of layers ($\bar{\gamma}_s$, $\bar{\gamma}_t$, $\bar{\gamma}_f$, $\bar{\gamma}_c$, $\bar{\gamma}_l$). Kernels are denoted as $\{\text{T}\times\text{S}^2, \text{C}\}$ for temporal, spatial, and channel sizes, while strides as $\{$temporal stride, spatial stride$^2\}$. Convolutional residual blocks are represented in brackets. The speed ratio between the Fast and the Slow paths is 8, while the channel ratio is 1/8.

Chapter 7

Conclusions

This thesis reports the research activity I carried out during my PhD, from the definition of the problems to solution proposal and experimental evaluation. Being my PhD granted by and done in collaboration with Metaliquid S.R.L, it focused on the development of deep learning solutions for video-related tasks with the goal of proposing effective algorithms and models able to fulfill the requirements of both academia and industry.

Our research contributed to specific video-based tasks, *i.e*. spatio-temporal action detection and soccer event spotting, with the continuous goal of making them relevant for the academic world, but also suitable for production scenarios and feasible in practice. Moreover, general issues caused by video data have been addressed. Specifically, video databases are usually orders of magnitude larger compared to image datasets, determining the need for large deep neural networks in order to learn from this massive data variability, with millions (or even billions) of parameters. This entails huge computational and hardware requirements, resulting in the video research community being limited by this constraint. On the other hand, we tackled the inherent privacy issues of videos, common to all action recognition systems involving humans.

In the following, we briefly summarize the contributions presented in this thesis for each task, we describe possible future directions, and we acknowledge all those who played a role in the development of the proposed solutions. Finally, in Appendix A, we also present other research activities done in collaboration with other PhD students and/or researchers, and which fall under different topics from those dealt with up to now.

Spatio-temporal action detection

Spatio-temporal action detection aims to predict a fine-grained representation of human actions in video clips. Given an input video, the goal is to detect all the actors in the scene for specific keyframes and to classify their (possibly multiple) actions. Our proposal consists of a lightweight module based on a graph representation of the entities, considering multiple detections in a frame (space) and in subsequent frames, too (time). A graph-attention network is devised for learning the relations between graph nodes, and the strength of the edges is made stronger if they are spatially close. Our STAGE module does not need to be finetuned end-to-end with the feature extraction network, and could potentially be applied to every spatio-temporal backbone. Experiments over three different challenging benchmarks confirm the effectiveness of the approach, which is able to reach state-of-the-art performance with negligible training overhead and minimum additional parameters.

Soccer action spotting

The automatic highlight generation from broadcast sports videos is of great importance for the multimedia industry and represented a key goal in the collaboration with Metaliquid S.R.L. during my PhD. In this sense, soccer action spotting is a recently proposed task in literature, which we decided to tackle as a first step towards highlight generation. Specifically, given a broadcast soccer match, the objective is to find the exact timestamp in which relevant events (like *goals* or *substitutions*) occur. Our solution is a simple and lightweight network inspired by detection models: given a clip from the whole game, it simultaneously predicts the action class (which could also be *background*) and the relative offset of the event in the clip, by using both classification and regression losses. Moreover, a masking strategy forces the network to focus on the frames following the event, which usually bring the most informative visual cues. Our solution reaches state-of-the-art results even when using pre-computed features over the SoccerNet dataset. Our proposal placed third in the SoccerNet-v2 challenge in the CVPR 2021 International Challenge on Activity Recognition Workshop.

Privacy-preserving action recognition

When handling video clips, we realized that sensitive information of involved humans is often disclosed, which could result in privacy issues when the action recognition model is run by a third-party service provider. This problem is not

specific to action analysis but is inherent to all tasks involving humans and videos. Focusing on action recognition, in this thesis we proposed a knowledge-distillation approach for training a student network on blurred videos, thus hiding sensitive information of involved actors at two levels of granularity: masking people's faces (light anonymization) or whole bodies (strong anonymization). During training, the student network is given only anonymized videos as input but leverages the output logits of a teacher network pretrained on original, not blurred video clips. The distillation is based on both direct knowledge transfer between obfuscated and non-obfuscated videos and pairwise relations between corresponding couples of samples. During inference, the student is able to predict correct labels from anonymized videos, without requiring the teacher privileged information. In this way, only the training set needs to be disclosed, for pretraining the teacher model: in a client-server scenario, the user can choose an anonymization schema in agreement with the provider, and send blurred videos in order to perform inference and avoid a sensitive information leak. Results show that the student is able to reach (or even exceed) the teacher performance in case of face anonymization and to achieve negligible accuracy loss in presence of strong anonymization strategies.

Efficiency in action recognition

Another common issue of handling video datasets is their huge variability given by the additional temporal dimension, which usually determines large models with millions (or even billions) of parameters and long training times. In this thesis, we also presented a strategy for reducing the computational complexity of existing spatio-temporal networks, while limiting the loss in accuracy. Specifically, we define a given action recognition model using a number of axes, *e.g.* the number of layers, the input spatial resolution, or the input temporal resolution (number of input frames). We then devise an iterative algorithm, named PAS, which trains a number of network hypotheses at each iteration, where each of them is obtained from the model chosen at the previous iteration, by reducing one of the axes in order to save a fixed amount of computation. Each hypothesis is trained starting from the weights of the previously obtained model (when the reduction axis allows it), and by exploiting knowledge distillation with the original spatio-temporal network (the one from which the first iteration starts). At each iteration, we keep the hypothesis with the best computation/accuracy trade-off. PAS allows to choose a computational configuration of choice and to continue iterations until reaching it. For instance, PAS is able to reduce the computation (in terms of floating-point operations) by four times, with negligible accuracy loss.

Future works and open problems

Even if the work presented in this thesis provides research advancements in the mentioned fields, it also represents a starting point for future research and applications. Many of the possible future works have been highlighted by the reviewers of our published papers, some of which we report in the following. In spatio-temporal action detection, for instance, an end-to-end finetuning of the STAGE module together with the feature extraction backbone has not been explored, given the resource and computation requirements. For the privacy-preserving strategy, only a limited number of obfuscation techniques has been explored, which reduces the possibility for the final user to choose an anonymization technique of choice for his videos. Moreover, domain adaptation could be a possible class of approaches for reducing the domain gap presented in Sec. 5.2.1, and privileged distillation could be a solution for preserving sensitive information for many other tasks where people's identity should not be disclosed. For the PAS strategy, a limited number of existing spatio-temporal networks have been considered, and the overall training process consumes a huge amount of resources, which is not desirable for a method aiming to reduce the inference cost. The opportunity to search for the optimal set of operations to be applied at once represents a major area of improvement. Finally, for the soccer event spotting task, highlight generation has been partially tackled: there is still a lot of work to be done in order to generate highlights from automatically detected events.

Publications, achievements, and acknowledgments

The majority of the works presented in this thesis have been published in international conferences and journals. A detailed list of publications is available in Appendix B. Moreover, the work on soccer event spotting placed third in the SoccerNet-v2 challenge in the CVPR 2021 International Challenge on Activity Recognition Workshop. During my PhD, I also had the opportunity to support some customer-oriented activities of Metaliquid S.R.L. even for projects different from those presented in this thesis. As a final note, I would like to thank all my colleagues from the AImageLab, together with my tutors and supervisors for the advice and the opportunities, without which this work would not have been possible. Moreover, I thank Metaliquid for the continuous support, all the UNIMORE ICT school, the NVIDIA AI Technology Centre, and CINECA with their staff for the collaboration and assistance.

Appendix A

Other research activities

In addition to the research activities presented in the main chapters of this thesis, which follow my main research topic focused on deep learning applied to video understanding, I also had the opportunity to work on deep learning applied to the cultural heritage and to digital humanities. In the following, we briefly report two works in this field, related to image-to-image translation and face retrieval in the context of paintings and artworks in general.

A.1 Art2Real

Computer vision techniques have been rarely adapted to work in the domain of cultural heritage, mainly because applying state-of-the-art techniques to artworks is rather difficult, and often brings poor performance. This can be motivated by the fact that the visual appearance of artworks is different from that of photo-realistic images, due to the presence of brush strokes, the creativity of the artist and the specific artistic style at hand. As current vision pipelines exploit large datasets consisting of natural images, learned models are largely biased towards them. The result is a gap between high-level convolutional features of the two domains, which leads to a decrease in performance in the target tasks, such as classification, detection or segmentation.

This chapter proposes a solution to the aforementioned problem that avoids

This chapter is related to publications 1, 2, 3, 8 reported in Appendix B, by the author of the thesis. See Appendix B for details.

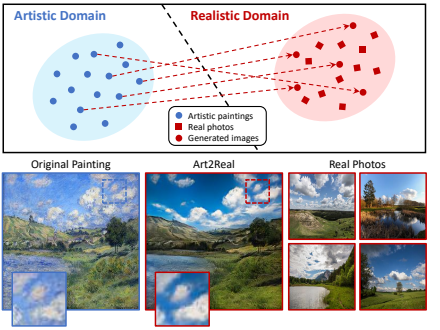


Figure A.1: We present *Art2Real*, an architecture which can reduce the gap between the distributions of visual features from artistic and realistic images, by translating paintings to photo-realistic images.

the need for re-training neural architectures on large-scale datasets containing artistic images. In particular, we propose an architecture which can reduce the shift between the feature distributions from the two domains, by translating artworks to photo-realistic images which preserve the original content. A sample of this setting is depicted in Fig. A.1.

As paired training data is not available for this task, we revert to an unpaired image-to-image translation setting [171], in which images can be translated between different domains while preserving some underlying characteristics. In our *art-to-real* scenario, the first domain is that of paintings while the second one is that of natural images. The shared characteristic is that they are two different visualizations of the same class of objects, for example, they both represent landscapes.

Contributions. In the translation architecture that we propose, new photo-realistic images are obtained by retrieving and learning from existing details of natural images and exploiting a weakly-supervised semantic understanding of the artwork. To this aim, a number of memory banks of realistic patches is built from the set of photos, each containing patches from a single semantic class in a memory-efficient representation. By comparing generated and real images at the patch level, in a multi-scale manner, we can then drive the training of a generator network which learns to generate photo-realistic details, while preserving the semantics of the original painting. As performing a semantic understanding of the original painting would create a chicken-egg problem, in which unreliable data is used to drive

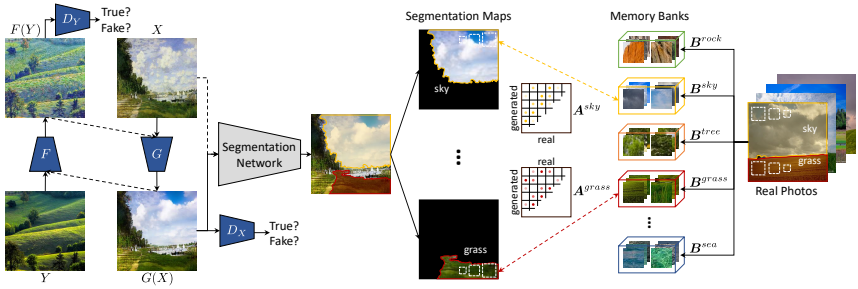


Figure A.2: Overview of our *Art2Real* approach.

the training and the generation, we propose a strategy to update the semantic masks during the training, leveraging the partial convergence of a cycle-consistent framework.

A.1.1 Proposed approach

Our goal is to obtain a photo-realistic representation of a painting. The proposed approach explicitly guarantees the realism of the generation and a semantic binding between the original artwork and the generated picture. An overview of our model is presented in Fig. A.2.

Patch memory banks

Given a semantic segmentation model, we define a pre-processing step with the aim of building the memory banks of patches which will drive the generation. Each memory bank $\boldsymbol{B}^c$ is tied to a specific semantic class c, in that it can contain only patches which belong to its semantic class. To define the set of classes, and semantically understand the content of an image, we adopt the weakly-supervised segmentation model from Hu *et al*. [59]: in this approach, a network is trained to predict semantic masks from a large set of categories, by leveraging the partial supervision given by detections. We also define an additional background memory bank, to store all patches which do not belong to any semantic class.

Following a sliding-window policy, we extract fixed-size RGB patches from the set of real images and put them in a specific memory $\boldsymbol{B}^c$, according to the class label c of the mask in which they are located. Since a patch might contain

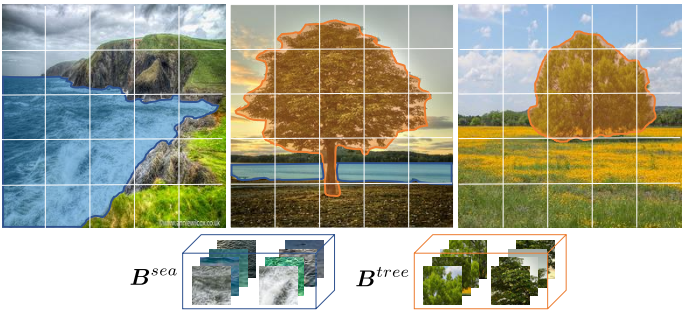


Figure A.3: Memory banks building. A segmentation model [59] computes segmentation masks for each realistic image in the dataset, then RGB patches belonging to the same semantic class are placed in the same memory bank.

pixels which belong to a second class label or the background class, we store in $\boldsymbol{B}^c$ only patches containing at least 20% pixels from class c.

Therefore, we obtain a number of memory banks equal to the number of different semantic classes found in the dataset, plus the background class, where patches belonging to the same class are placed together (Fig. A.3). Also, semantic information from generated images is needed: since images generated at the beginning of the training are less informative, we first extract segmentation masks from the original paintings. As soon as the model starts to generate meaningful images, we employ the segmentation masks obtained on generated images.

Semantically-aware generation

The unpaired image-to-image translation model that we propose maps images belonging to a domain X (that of artworks) to images belonging to a different domain Y (that of natural images), preserving the overall content. Suppose we have a generated realistic image $G(x)$ at each training step, produced by a mapping function G which starts from an input painting x. We adopt the previously obtained memory banks of realistic patches and the segmentation masks of the paintings in order to both enhance the realism of the generated details and keep the semantic content of the painting.

Pairing similar patches in a meaningful way. At each training step, $G(x)$ is split in patches as well, maintaining the same stride and patch size used for the

memory banks. Reminding that we have the masks for all the paintings, we denote a mask of the painting x with class label c as $\boldsymbol{M}_x^c$. We retrieve all masks $\boldsymbol{M}_x$ of the painting x from which $G(x)$ originates, and assign each generated patch to the class label c of the mask $\boldsymbol{M}_x^c$ in which it falls. If a patch belongs to different masks, it is also assigned to multiple classes. Then, generated patches assigned to a specific class c are paired with similar realistic patches in the memory bank $\boldsymbol{B}^c$, *i.e.* the bank containing realistic patches with class label c. Given realistic patches belonging to $\boldsymbol{B}^c$, $\boldsymbol{B}^c = \{b_j^c\}$ and the set of generated patches with class label c, $\boldsymbol{K}^c = \{k_i^c\}$, we center both sets with respect to the mean of patches in $\boldsymbol{B}^c$, and we compute pairwise cosine distances as follows:

$$d_{ij}^c = \left(1 - \frac{(k_i^c - \mu_b^c) \cdot (b_j^c - \mu_b^c)}{\|k_i^c - \mu_b^c\|_2 \left\|b_j^c - \mu_b^c\right\|_2}\right) \tag{A.1}$$

where $\mu_b^c = \frac{1}{N_c} \sum_j b_j^c$, being N_c the number of patches in memory bank $\boldsymbol{B}^c$. We compute a number of distance matrices equal to the number of semantic classes found in the original painting x. Pairwise distances are subsequently normalized as follows:

$$\tilde{d}_{ij}^c = \frac{d_{ij}^c}{\min_l d_{il}^c + \epsilon}, \text{ where } \epsilon = 1e-5 \tag{A.2}$$

and pairwise affinity matrices are computed by applying a row-wise softmax normalization:

$$\boldsymbol{A}_{ij}^c = \frac{\exp(1 - \tilde{d}_{ij}^c/h)}{\sum_l \exp(1 - \tilde{d}_{il}^c/h)} = \begin{cases} \approx 1 & \text{if } \tilde{d}_{ij}^c \ll \tilde{d}_{il}^c \; \forall \, l \neq j \\ \approx 0 & \text{otherwise} \end{cases} \tag{A.3}$$

where $h > 0$ is a bandwidth parameter. Thanks to the softmax normalization, each generated patch k_i^c will have a high-affinity degree with the nearest real patch and with other not negligible near patches. Moreover, affinities are computed only between generated and artistic patches belonging to the same semantic class.

Approximate affinity matrix. Computing the entire affinity matrix would require an intractable computational overhead, especially for classes with a memory bank containing millions of patches. In fact matrix $\boldsymbol{A^c}$ has as many rows as the number of patches of class c extracted from $G(x)$ and as many columns as the number of patches contained in the memory bank $\boldsymbol{B}^c$.

To speed up the computation, we build a suboptimal Nearest Neighbors index $\boldsymbol{I}^c$ for each memory bank. When the affinity matrix for a class c has to be computed, we conduct a k-NN search through $\boldsymbol{I}^c$ to get the k nearest samples of

each generated patch k_i^c. In this way, $\boldsymbol{A}^c$ will be a sparse matrix with at most as many columns as k times the number of generated patches of class c. The Softmax in Eq. A.3 ensures that the approximated version of the affinity matrix is very close to the exact one if the k-NN searches through the indices are reliable. We adopt inverted indexes with exact post-verification, implemented in the Faiss library [67]. Patches are stored with their RGB values when memory banks have less than one million vectors; otherwise, we use a PCA pre-processing step to reduce their dimensionality, and scalar quantization to limit the memory requirements of the index.

Maximizing the similarity. A contextual loss [96] for each semantic class in $\boldsymbol{M}_x$ aims to maximize the similarity between couples of patches with high affinity value:

$$\mathcal{L}_{CX}^c(\boldsymbol{K}^c, \boldsymbol{B}^c) = -\log\left(\frac{1}{N_K^c}\left(\sum_i \max_j \boldsymbol{A}_{ij}^c\right)\right) \tag{A.4}$$

where N_K^c is the cardinality of the set of generated patches with class label c. Our objective is the sum of the previously computed single-class contextual losses over the different classes found in $\boldsymbol{M}_x$:

$$\mathcal{L}_{CX}(\boldsymbol{K}, \boldsymbol{B}) = \sum_c -\log\left(\frac{1}{N_K^c}\left(\sum_i \max_j \boldsymbol{A}_{ij}^c\right)\right) \tag{A.5}$$

where c assumes all the class label values of masks in $\boldsymbol{M}_x$. Note that masks in $\boldsymbol{M}_x$ are not constant during training: at the beginning, they are computed on paintings, then they are regularly extracted from $G(x)$.

Multi-scale variant. To enhance the realism of generated images, we adopt a multi-scale variant of the approach, which considers different sizes and strides in the patch extraction process. The set of memory banks is therefore replicated for each scale, and $G(x)$ is split at multiple scales accordingly. Our loss function is given by the sum of the values from Eq. A.5 computed at each scale, as follows:

$$\mathcal{L}_{CXMS}(\boldsymbol{K}, \boldsymbol{B}) = \sum_s \mathcal{L}_{CX}^s(\boldsymbol{K}, \boldsymbol{B}) \tag{A.6}$$

where each scale s implies a specific patch size and stride.

Unpaired image-to-image translation baseline

Our objective assumes the availability of a generated image $G(x)$ which is, in our task, the representation of a painting in the photo-realistic domain. In our work,

we adopt a cycle-consistent adversarial framework [171] between the domain of paintings from a specific artist X and the domain of realistic images Y. The data distributions are $x \sim p_{data}(x)$ and $y \sim p_{data}(y)$, while $G : X \rightarrow Y$ and $F : Y \rightarrow X$ are the mapping functions between the two domains. The two discriminators are denoted as D_Y and D_X.

The full cycle-consistent adversarial loss [171] is the following:

$$\begin{aligned}\mathcal{L}_{cca}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) \\ + \mathcal{L}_{GAN}(F, D_X, Y, X) \\ + \mathcal{L}_{cyc}(G, F)\end{aligned} \tag{A.7}$$

where the two adversarial losses are:

$$\begin{aligned}\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[logD_Y(y)] \\ + \mathbb{E}_{x \sim p_{data}(x)}[log(1 - D_Y(G(x)))]\end{aligned} \tag{A.8}$$

$$\begin{aligned}\mathcal{L}_{GAN}(F, D_X, Y, X) = \mathbb{E}_{x \sim p_{data}(x)}[logD_X(x)] \\ + \mathbb{E}_{y \sim p_{data}(y)}[log(1 - D_X(F(y)))]\end{aligned} \tag{A.9}$$

and the cycle consistency loss, which requires the original images x and y to be the same as the reconstructed ones, $F(G(x))$ and $G(F(y))$ respectively, is:

$$\begin{aligned}\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[\|F(G(x)) - x\|] \\ + \mathbb{E}_{y \sim p_{data}(y)}[\|G(F(y)) - y\|].\end{aligned} \tag{A.10}$$

Full objective

Our full semantically-aware translation loss is given by the sum of the baseline objective, *i.e*. Eq. A.7, and our patch-level similarity loss, *i.e*. Eq. A.6:

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y, \boldsymbol{K}, \boldsymbol{B}) = \mathcal{L}_{cca}(G, F, D_X, D_Y) \\ + \lambda \mathcal{L}_{CXMS}(\boldsymbol{K}, \boldsymbol{B})\end{aligned} \tag{A.11}$$

where λ controls our multi-scale contextual loss weight with respect to the baseline objective.

A.1.2 Experimental results

Datasets. In order to evaluate our approach, different sets of images, both from artistic and realistic domains, are used. Our tests involve both sets of paintings from specific artists and sets of artworks representing a given subject from different authors. We use paintings from Monet, Cezanne, Van Gogh, Ukiyo-e style and landscapes from different artists along with real photos of landscapes, keeping an underlying relationship between artistic and realistic domains. We also show results using portraits and real people photos. All artworks are taken from Wikiart.org, while landscape photos are downloaded from Flickr through the combination of tags `landscape` and `landscapephotography`. To obtain people photos, images are extracted from the CelebA dataset [93]. All the images are scaled to 256×256 pixels, and only RGB pictures are used. The size of each training set is, respectively, Monet: 1072, Cezanne: 583, Van Gogh: 400, Ukiyo-e: 825, landscape paintings: 2044, portraits: 1714, real landscape photographs: 2048, real people photographs: 2048.

Architecture and training details. To build generators and discriminators, we adapt generative networks from Johnson *et al*. [68], with two stride-2 convolutions to downsample the input, several residual blocks and two stride-1/2 convolutional layers for upsampling. Discriminative networks are PatchGANs [63, 78, 81] which classify each square patch of an image as real or fake.

Memory banks of real patches are built using all the available real images, *i.e*. 2048 images both for landscapes and for people faces, and are kept constant during training. Masks of the paintings, after epoch 40, are regularly updated every 20 epochs with those from the generated images. Patches are extracted at three different scales: 4×4, 8×8 and 16×16, using three different stride values: 4, 5 and 6 respectively. The same patch sizes and strides are adopted when splitting the generated image, in order to compute affinities and the contextual loss. We use a multi-scale contextual loss weight λ, in Eq. A.11, equal to 0.1.

We train the model for 300 epochs through the Adam optimizer [72] and using mini-batches with a single sample. A learning rate of 0.0002 is kept constant for the first 100 epochs, making it linearly decay to zero over the next 200 epochs. An early stopping technique is used to reduce training times. In particular, at each epoch the Fréchet Inception Distance (FID) [55] is computed between our generated images and the set of real photos: if it does not decrease for 30 consecutive epochs, the training is stopped. We initialize the weights of the model from a Gaussian distribution with 0 mean and standard deviation 0.02.

Competitors. To compare our results with those from state-of-the-art techniques,

Method	Monet	Cezanne	Van Gogh	Ukiyo-e	Landscapes	Portraits	Mean
Original paintings	69.14	169.43	159.82	177.52	59.07	72.95	117.99
Style-transferred reals	74.43	114.39	137.06	147.94	70.25	62.35	101.07
DRIT [79]	68.32	109.36	108.92	117.07	59.84	44.33	84.64
UNIT [90]	56.18	97.91	98.12	89.15	47.87	43.47	72.12
Cycle-GAN [171]	49.70	85.11	85.10	98.13	44.79	**30.60**	65.57
Art2Real	**44.71**	**68.00**	**78.60**	**80.48**	**35.03**	34.03	**56.81**

Table A.1: Evaluation in terms of Fréchet Inception Distance [55].

we train Cycle-GAN [171], UNIT [90] and DRIT [79] approaches on the previously described datasets. The adopted code comes from the authors' implementations and can be found in their GitHub repositories. The number of epochs and other training parameters are those suggested by the authors, except for DRIT [79]: to enhance the quality of the results generated by this competitor, after contacting the authors we employed spectral normalization and manually chose the best epoch through visual inspection and by computing the FID [55] measure. Moreover, being DRIT [79] a diverse image-to-image translation framework, its performance depends on the choice of an attribute from the attribute space of the realistic domain. For fairness of comparison, we generate a single realistic image using a randomly sampled attribute. We also show quantitative results of applying the style transfer approach from Gatys *et al.* [42], with content images taken from the realistic datasets and style images randomly sampled from the paintings, for each set.

Visual quality evaluation

We evaluate the visual quality of our generated images using both automatic evaluation metrics and user studies.

Fréchet Inception Distance. To numerically assess the quality of our generated images, we employ the Fréchet Inception Distance [55]. It measures the difference of two Gaussians, and it is also known as Wasserstein-2 distance [139]. The FID d between a Gaussian G_1 with mean and covariance (m_1, C_1) and a Gaussian G_2 with mean and covariance (m_2, C_2) is given by:

$$d^2(G_1, G_2) = \|m_1 - m_2\|_2^2 + Tr(C_1 + C_2 - 2(C_1 C_2)^{1/2}) \tag{A.12}$$

For our evaluation purposes, the two Gaussians are fitted on Inception-v3 [128] activations of real and generated images, respectively. The lower the Fréchet

	Cycle-GAN [171]	UNIT [90]	DRIT [79]
Realism	36.5%	27.9%	14.2%
Coherence	48.4%	25.5%	7.3%

Table A.2: User study results. We report the percentage of times an image from a competitor was preferred against ours. Our method is always preferred more than 50% of the times.

Inception Distance between these Gaussians, the more generated and real data distributions overlap, *i.e*. the realism of generated images increases when the FID decreases. Table A.1 shows FID values for our model and a number of competitors. As it can be observed, the proposed approach produces a lower FID on all settings, except for portraits, in which we rank second after Cycle-GAN. Results thus confirm the capabilities of our method in producing images which looks realistic to pre-trained CNNs.

Human judgment. In order to evaluate the visual quality of our generated images, we conducted a user study on the Figure Eight crowd-sourcing platform. In particular, we assessed both the realism of our results and their coherence with the original painting. To this aim, we conducted two different evaluation processes, which are detailed as follows:

- In the *Realism* evaluation, we asked the user to select the most realistic image between the two shown, both obtained from the same painting, one from our method and the other from a competitor;
- In the *Coherence* evaluation, we presented the user the original painting and two generated images which originate from it, asking to select the most faithful to the artwork. Again, generated images come from our method and a competitor.

Each test involved our method and one competitor at a time leading to six different tests, considering three competitors: Cycle-GAN [171], UNIT [90], and DRIT [79]. A set of 650 images were randomly sampled for each test, and each image pair was evaluated from three different users. Each user, to start the test, was asked to successfully evaluate eight example pairs where one of the two images was definitely better than the other. A total of 685 evaluators were involved in our tests. Results are presented in Table A.2, showing that our generated images are always chosen more than 50% of the times.

Method	Classification	Segmentation	Detection
Real Photos	3.99	0.63	2.03
Original paintings	4.81	0.67	2.58
Style-transferred reals	5.39	0.70	2.89
DRIT [79]	5.14	0.67	2.56
UNIT [90]	4.88	0.69	2.54
Cycle-GAN [171]	4.81	0.67	2.50
Art2Real	**4.50**	**0.66**	**2.42**

Table A.3: Mean entropy values for classification, segmentation, and detection of images generated through our method and through competitor methods.

Reducing the domain shift

We evaluate the capabilities of our model to reduce the domain shift between artistic and real data, by analyzing the performance of pre-trained convolutional models and visualizing the distributions of CNN features.

Entropy analysis. Pre-trained architectures show increased performances on images synthesized by our approach, in comparison with original paintings and images generated by other approaches. We visualize this by computing the entropy of the output of state-of-the-art architectures: the lower the entropy, the lower the uncertainty of the model about its result. We evaluate the entropy on classification, semantic segmentation, and detection tasks, adopting a ResNet-152 [52] trained on ImageNet [28], Hu *et al*. [59]'s model and Faster R-CNN [112] trained on the Visual Genome [1, 75], respectively. Table A.3 shows the average image entropy for classification, the average pixel entropy for segmentation and the average bounding-box entropy for detection, computed on all the artistic, realistic and generated images available. Our approach is able to generate images which lower the entropy, on average, for each considered task with respect to paintings and images generated by the competitors.

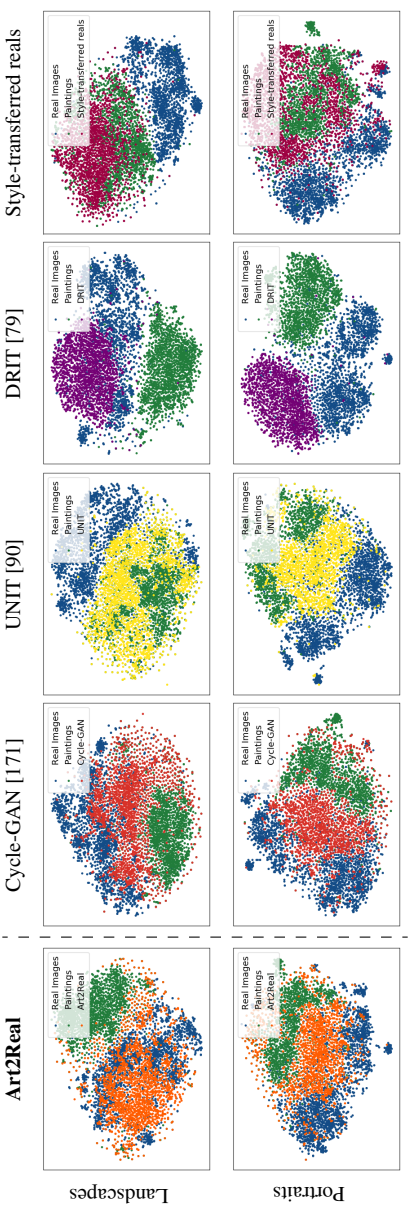


Figure A.4: Distribution of ResNet-152 features extracted from landscape and portrait images. Each row shows the results of our method and competitors on a specific setting.

Feature distributions visualization. To further validate the domain shift reduction between real images and generated ones, we visualize the distributions of features extracted from a CNN. In particular, for each image, we extract a visual feature vector coming from the average pooling layer of a ResNet-152 [52], and we project it into a 2-dimensional space by using the t-SNE algorithm [95]. Fig. A.4 shows the feature distributions on two different sets of paintings (*i.e.*, landscapes and portraits) comparing our results with those of competitors. Each plot represents the distribution of visual features extracted from paintings belonging to a specific set, from the corresponding images generated by our model or by one of the competitors, and from the real photographs depicting landscapes or, in the case of portraits, human faces. As it can be seen, the distributions of our generated images are in general closer to the distributions of real images than to those of paintings, thus confirming the effectiveness of our model in the domain shift reduction.

Qualitative results

Besides showing numerical improvements with respect to state-of-the-art approaches, we present some qualitative results coming from our method, compared to those from Cycle-GAN [171], UNIT [90], and DRIT [79]. We show examples of landscape and portrait translations in Fig. A.5. We observe increased realism in our generated images, due to more detailed elements and fewer blurred areas, especially in the landscape results. Portrait samples reveal that brush strokes disappear completely, leading to a photo-realistic visualization. Our results contain fewer artifacts and are more faithful to the paintings, more often preserving the original facial expression. Moreover we show, through a number of qualitative examples, that a fake-realistic image generated by our architecture is easily understandable by state-of-the-art models, unlike its original painted version. Fig. A.6 shows painting-generated image pairs which are both given as input to Mask R-CNN [51] pre-trained on COCO [88]: besides improving the score for well-labeled masks, we are also able to reduce the number of false positives (top-left and bottom-right) and false negatives (bottom-left). Finally, Fig. A.7 illustrates bounding boxes predicted by Faster R-CNN [112] pre-trained on Visual Genome [75]: again we demonstrate improved results, detecting true clouds instead of pillows (top-right) or true sky instead of water (top-left and bottom-left).

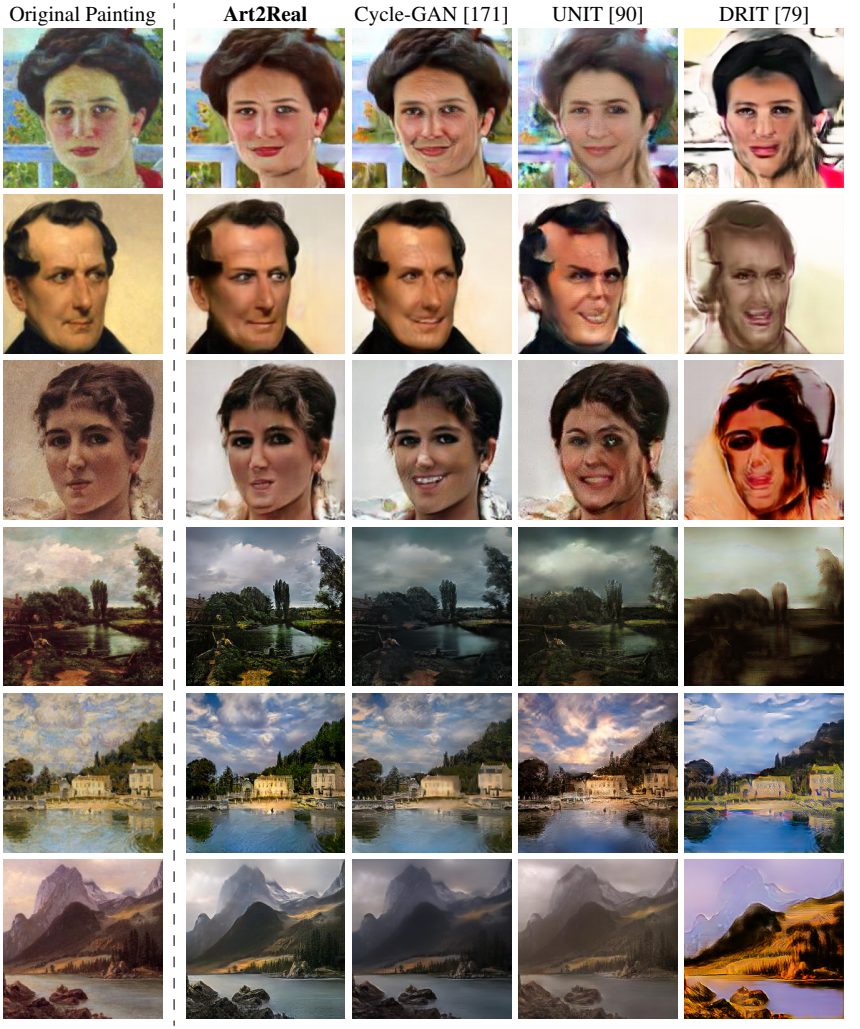


Figure A.5: Qualitative results on portraits and landscapes.

Figure A.6: Segmentation results on original portraits and their translated versions. Our method leads to improved segmentation performance of existing models on artistic data.

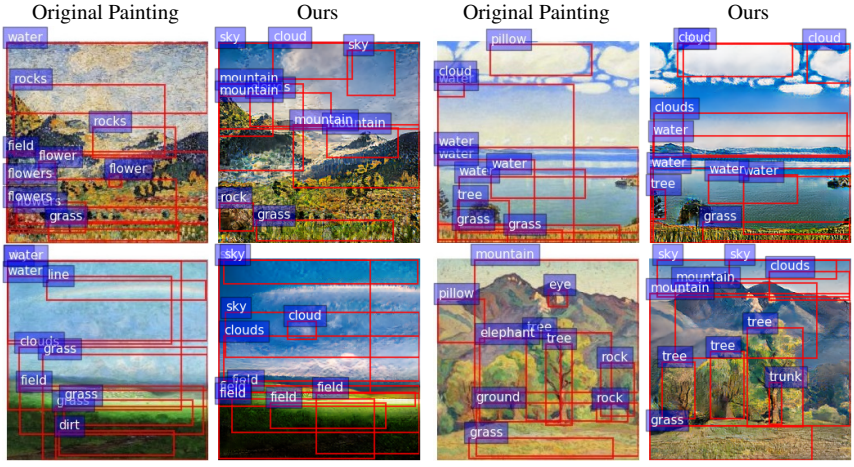


Figure A.7: Detection results on original paintings and their translated versions. Our model leads to improved results of existing detection models on the artistic domain.

A.2 PersonArt

Even though there isn't an artwork depicting yourself, there are probably some which contain faces that look just like you, as testified by the number of people who found their "Doppelgänger" in a painting while visiting a museum[1] [24]. Taking inspiration from these accidental discoveries, we develop an interactive multimedia solution which can retrieve similar faces in a collection of paintings given a query photo from a visitor. Once the visitor has arrived at the museum, we imagine a situation where he can take a photo of himself, and get back the name and the location of the painting where his Doppelgänger lies, as a possible starting point for his visit.

This setting requires to detect faces in both paintings and selfies taken by the visitors, and also to retrieve faces that look similar between the real domain (that of photographs) and the artistic one (that of paintings). As images from these two domains can look very different in terms of low-level statistics due to the presence of strokes and the peculiarities of the artistic style, the task demands for a domain adaptation stage, in which features extracted from the two domains can be merged and compared to get the final retrieval result. Additionally, as often happens when dealing with artistic data, there are no annotated datasets for the task, since no dataset contains photos of real people annotated with the most similar paintings in a given collection.

Contributions. To tackle these issues, we define a deep learning-based domain translation strategy, which lets us recover artistic proxies of real faces while still exploiting annotations coming from datasets with real faces. This allows us to exploit the supervision given by datasets originally designed for tasks similar to the one we are addressing (*e.g.* face recognition and attribute prediction) while working in a domain where no supervision is given. In this regard, this is the first work to propose a face retrieval system, trained with a shared embedding space and with no explicit supervision, for the artistic domain.

Beyond developing a retrieval algorithm, we also design a multimedia system for face retrieval in the artistic domain with user interaction. In our system, we let the user customize the final result by interacting with the application. Moreover, we let the user impose constraints on the attributes of retrieved faces, *e.g.* requiring that retrieved results should be smiling, wearing a hat, having a big nose, and so on. This in turn permits a better exploration of the space of artworks. An overview of our approach is depicted in Figure A.8.

[1] `https://www.thoughtco.com/art-museum-doppelgangers-4154789`

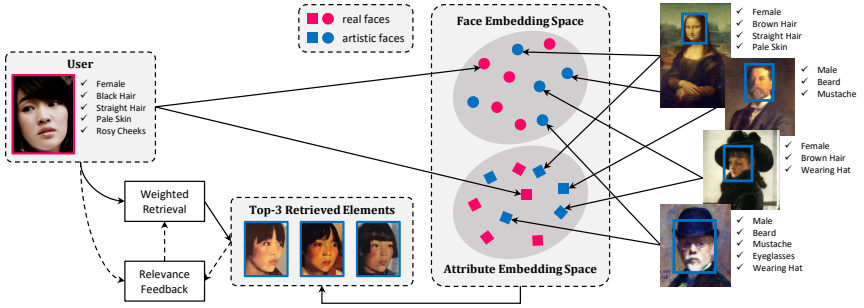


Figure A.8: Overview of our approach: given a real face as query, we retrieve similar faces from paintings, jointly taking into account the aesthetic similarity and the correspondence between semantic attributes. The user can further refine the retrieved set by providing feedback and imposing constraints on attributes.

The proposal is evaluated on both synthetic and realistic settings, on a variety of different datasets. Experimental evaluations will show the effectiveness of the presented architecture, when employing different face embeddings, and the role of both face recognition and attribute detection in retrieval performance.

A.2.1 Learning cross-domain face retrieval

As it is rarely feasible to find and annotate a variety of Doppelgängers from paintings, almost no data exist for training a retrieval algorithm to find similar faces between the artistic and the real domain. While sources of supervision are available on photo-realistic images, our method needs to be aware of the domain shift between artistic and real images, which demands different feature extractors and a domain adaptation strategy.

To address these challenges, we define a translation approach that generates *artistic proxies* of real faces and lets us exploit the annotations of datasets containing real faces. Secondly, we address the lack of training data by exploiting proxy tasks for which training data is available and which share some of the underlying characteristics of our task. We employ face recognition and face attribute detection, as they are closely related to the goal of finding similar faces and define a learned retrieval strategy which lets us recover faces across the two domains. Finally, we also discuss how our architecture can be combined with relevant feedback strategies to consider suggestions from the user during the retrieval phase. As an

additional result, we provide explanation maps to highlight which regions of the retrieved results contribute most to the final similarity.

Addressing the domain shift

To be robust to the domain shift between artistic and real images, we define a way of converting real images to artistic proxies, which we will later employ as a data translation strategy to train the retrieval network. Here, we take inspiration from style transfer techniques, firstly proposed in [43]: given a realistic input face I^R and an artistic face I^A, we build an artistic version of I^R, called I^G, which preserves the content of I^R and the artistic style of I^A. The generated image can be built by back-propagating directly on its pixel values while minimizing a loss function L that takes into account both the artistic style of I^A and the content of I^R. Formally, pixel values of the input real image are updated as

$$I^G_{i,j} \leftarrow I^G_{i,j} - \eta \frac{\partial L}{\partial I^G_{i,j}}, \tag{A.13}$$

where (i, j) represents the coordinates of a pixel on the input image, and η is the step size of Stochastic Gradient Descent. To encode the artistic style of I^A, we focus on textures and employ the Gram matrix obtained from the activations of a pre-trained CNN when applied on I^A. Given a layer l from the CNN, we define an ℓ_2 loss function between the Gram matrix of the generated image and that of the artistic image, as follows:

$$E_s = \frac{1}{4} \sum_{i,j} \left(\mathcal{G}^l(I^A)_{i,j} - \mathcal{G}^l(I^G)_{i,j} \right)^2, \tag{A.14}$$

where $\mathcal{G}^l$ denotes the Gram matrix obtained from activations at layer l, and $\mathcal{G}^l(\cdot)_{i,j}$ denotes its element in position (i, j). To encode the content of I^R, and to make sure that the same content is preserved in I^G, we combine the previously defined loss with a regularization term built on top of CNN activations. Given a layer l, we define an ℓ_2 loss function between raw activations as follows:

$$E_c = \frac{1}{2} \sum_{i,j} \left(\mathcal{F}^l(I^R)_{i,j} - \mathcal{F}^l(I^G)_{i,j} \right)^2, \tag{A.15}$$

where $\mathcal{F}^l$ denotes the activations at layer l. By combining the style loss with the reconstruction regularizer into the final loss (*i.e.* $L = E_s + \alpha E_c$), and exploiting

appropriate pre-trained layers to encode style and content, we obtain a *style-transferred* version of I^R which matches the artistic style of I^A. To encode multi-level and multi-scale information, we also combine the activations extracted at multiple layers and define a separate loss function for each layer.

Learning from face identities

Having obtained artistic proxies of real faces, we can train a similarity function which is aware of the domain gap between artistic and real faces. Firstly, we employ the preservation of face identities across domains as a proxy task. We employ a face recognition network $\phi(\cdot)$ to extract face-level features and a common embedding space, which is trained to recognize faces of the same identity across the two domains.

To project the representations of artistic and real images in a common semantic space, we perform a non-symmetric linear projection, followed by a ℓ_2-normalization step, so that the embedding space lies on the ℓ_2 unit ball. When projecting, we also employ a residual connection which was observed to slightly improve residual performance during our preliminary experiments. Formally:

$$f_a(I^A, \mathbf{w}_a, \mathbf{w}_\phi) = \ell_{2,norm}(\mathbf{w}_a^\intercal(1 + \phi(I^A, \mathbf{w}_\phi))) \tag{A.16}$$

$$f_r(I^R, \mathbf{w}_r, \mathbf{w}_\phi) = \ell_{2,norm}(\mathbf{w}_r^\intercal(1 + \phi(I^R, \mathbf{w}_\phi))), \tag{A.17}$$

where $\ell_{2,norm}$ is the ℓ_2 normalization function. Being D_ψ the output dimensionality of ϕ and D the dimensionality of the joint embedding space, $\mathbf{w}_r$ and $\mathbf{w}_a$ are $D_\psi \times D$ matrices which are in charge of storing different weights for the artistic and real projection branches. $\mathbf{w}_\phi$ indicates the weights of the face recognition network ϕ.

Artistic and real faces can be compared in the joint embedding space by computing the dot product (*i.e.* the cosine similarity) between their projections, so that the similarity between a (real) query face I^R and an artistic face I^A becomes

$$s_i(I^R, I^A) = f_a(I^A) \cdot f_r(I^R), \tag{A.18}$$

where we drop the dependency on weights for brevity. The utility of the joint embedding space is maximized when it exhibits suitable cross-domain matching properties, *i.e.* when distances in the embedding space correspond to meaningful distances across the artistic and the real domain, and when corresponding pairs are matched in the embedding space. When this is verified to some extent, the

embedding space acts as a bridge between the two domains and makes it possible to retrieve artistic faces given real faces as queries.

In order to learn an embedding space with such properties, we leverage face recognition datasets partially translated into the artistic domain using the strategy outlined in Sec. A.2.1. We train the parameters of the model according to a Hinge triplet ranking loss with maximum violation [34] and margin α, defined as

$$\ell(I^R, I^A) = \max_{\hat{I^A}} \left[\alpha - s(I^R, I^A) + s(I^R, \hat{I^A})\right]_+ + \\ + \max_{\hat{I^R}} \left[\alpha - s(I^R, I^A) + s(\hat{I^R}, I^A)\right]_+, \tag{A.19}$$

where $[x]_+ = \max(0, x)$. In the equation above, (I^R, I^A) is a matching artistic-real pair of faces (such that the identity of the person in I^R is the same of that in I^A), while $\hat{I^R}$ is a negative real face with respect to I^A (such that the person in $\hat{I^R}$ is different from that of I^A), and $\hat{I^A}$ is a negative artistic face with respect to I^R (such that the person in $\hat{I^A}$ is different from that of I^R). The two terms contained in the loss require that the difference in similarity between the matching and the non-matching pair is higher than a margin α: in the first term, this is done by considering a real anchor and matching or non-matching artistic images; in the latter, instead, an artistic image is used as anchor.

Learning from semantic attributes

While preserving people identities across domains is a good proxy objective for retrieving similar faces, the network described so far might be unable to maintain face attributes (*i.e.* hair style, presence of glasses, age) in retrieved elements, as it focuses on face identification features rather than considering properties of the face that might change over time. For this reason, we complement the common embedding with attribute detection capabilities, to ensure that the correct attributes are maintained in retrieved faces.

To this aim, we employ two attribute detection networks, one for the artistic and one for the real domain. We start from a face recognition model and then feed the activations of its last layer to n identical branches, each in charge of predicting the presence of an attribute. In our implementation, each branch is a composition of four fully connected layers, the last having an output size of two. Each branch is then trained, via a categorical cross-entropy loss, to predict the presence of the i-th attribute.

At test time, given predictions from the two attribute networks, we binarize them by tresholding, so to obtain a binary vector with length n for each image, and compute a similarity function by means of their Hamming distance as follows:

$$s_a(I^R, I^A) = 1 - \frac{\mathcal{H}(\gamma(I^R), \gamma(I^A))}{n}, \tag{A.20}$$

where $\gamma(\cdot)$ indicates the binarized prediction from the attribute prediction network, $\mathcal{H}$ the Hamming distance and n the number of attributes.

The final retrieval function is a weighted combination of the similarity computed through cross-domain face recognition and that computed by extracting semantic attributes:

$$s(I^R, I^A) = \frac{1 + s_i(I^R, I^A)}{2} + \lambda s_a(I^R, I^A), \tag{A.21}$$

where $s_i(I^R, I^A)$, resulting from a cosine similarity, is projected back to the same range of $s_a(I^R, I^A)$, *i.e.* $[0, 1]$. For a given λ, the resulting similarity score jointly takes into account the similarity of face traits and the similarity in attributes.

Interacting with user feedback

To foster the interactivity of the application and considering the subjectivity of the task, we also let the user give a feedback on retrieved results. This is, in turn, used to iteratively improve the quality of the retrieval following the preferences expressed by the user. In particular, the user is presented with the first k retrieved artworks and can refine the results by labelling a retrieved item either as positive (when it is more satisfactory than the others) or negative (if completely unsatisfactory).

As our retrieval depends on a face recognition and a face attribute branch, and given that the face attribute branch is easily controllable from the exterior (*e.g.* by asking the user which attributes he prefers to see in retrieved results), we here focus on the face recognition branch of our architecture. We employ two relevance feedback strategies to alter the embeddings of the query and database items. Given an item I^p labeled by the user as positive and an item I^n labelled as negative, we firstly change the position of the query I^q in the embedding space according to the user's feedback, as follows:

$$f_r(I^q) \leftarrow \alpha f_r(I^q) + \beta f_a(I^p) - \gamma f_a(I^n). \tag{A.22}$$

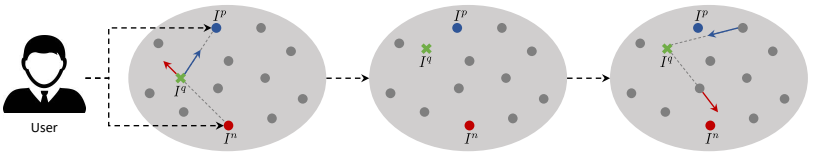


Figure A.9: Schema of the relevance feedback strategy. Given a query I^q, an item labeled by the user as positive I^p and one labeled as negative I^n, we firstly change the position of the query according to the user feedback. Then, we also change the position of the database items (gray dots) employing Feature Space Warping.

Then, we also change the position of the database items employing Feature Space Warping [19, 12]. This consists in moving all data points towards or away the new query vector $f_r(I^q)$ according to their similarities with the user's feedbacks. Formally,

$$\begin{aligned} f_a(I) \leftarrow f_a(I) &+ \eta e^{-c|f_a(I)-f_a(I^p)|}(f_r(I^q) - f_a(I)) \\ &- \eta e^{-c|f_a(I)-f_a(I^n)|}(f_r(I^q) - f_a(I)), \end{aligned} \tag{A.23}$$

where c is a bandwith parameter. The procedure above might be repeated at multiple feedback iterations, until the user is satisfied with at least one of the k retrieved results. A graphical overview of the approach is depicted in Fig. A.9.

In practice, as it will be reported in the experimental section, in most cases the initial retrieved set already contains sufficiently similar images and few feedback iterations are enough to retrieve a significant set of similar faces. Nevertheless the adoption of a relevance feedback strategy is useful for corner cases in which retrieval fails and to accommodate user's preferences.

Providing explanations

Even in the case that retrieved results are satisfactory before the user provides a feedback or imposes any constraint on attributes, it is important to explain why a particular artistic face was selected and presented to the user. This is particularly significant considering that the user might struggle to perceive his own face, and thus the retrieved results, in an objective way.

Following recent works on Explainable AI [2] and techniques on prediction attribution [120, 127, 125, 117], for each retrieved item I^A we provide the user with a saliency map that visually indicates which regions of I^R have mostly

contributed to the final similarity score $s(I^R, I^A)$, as an explanation of why the particular image was selected and presented. Ideally, a pixel in the saliency (or explanation) map $\mathcal{E}$ should be high if the corresponding pixel in I^R has contributed to increase the final similarity, lower otherwise.

To compute an explanation map $\mathcal{E}$, we follow the Integrated Gradients approach [127] by considering the straight-line path from a black baseline image to I^R and computing the gradients of the similarity score at all points along the path. Integrated gradients are obtained by accumulating these gradients. Specifically, a pixel (i, j) in $\mathcal{E}$ is computed as the path integral of the gradients along the straight-line path from 0 to $I^R(i, j)$, *i.e.*

$$\mathcal{E}_{i,j} = I^R_{i,j} \cdot \int_{\alpha=0}^{1} \frac{\partial s(\alpha I^R, I^A)}{\partial I^R_{i,j}} d\alpha. \tag{A.24}$$

Given the structure of our retrieval network, the computation of the partial derivative in Eq. A.24 requires to back-propagate on both the face identity branch of the architecture and the face attribute branch of the network, thus providing an explanation which takes into account the dual nature of the similarity function.

A.2.2 Datasets and implementation details

Datasets

To train and evaluate our model, we use – and in some cases extend – different datasets which contain real photos and portraits of people of known identities. To further validate our proposal, we also collect a large set of paintings containing people faces that can be used in real scenarios in which the corresponding portrait of a person is not known in advance.

Celeb-A [93]. This dataset contains more than $200,000$ face images belonging to $10,177$ different celebrities. Each image is annotated with 40 facial attributes ranging from the gender and hair style of the person to the presence of specific accessories such as eyeglasses, hat, or earrings. Since its size is sufficiently large to train a neural network, we employ this dataset to address the domain shift between artistic and real faces and to train the proposed architecture. To address the domain shift, we create a copy of this dataset where each image is converted into an artistic representation of itself using the style transfer strategy. For each image, we apply the style of a randomly selected painting from WikiArt[2] (*i.e.* different

[2] https://www.wikiart.org/

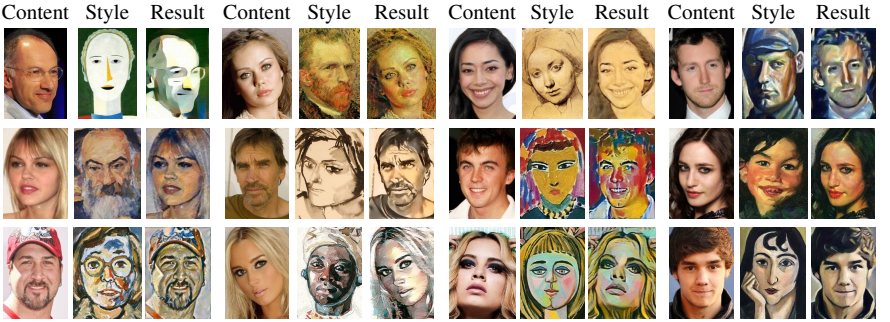


Figure A.10: Sample results of the style transfer strategy for addressing the domain shift between real and artistic faces. Real photos and paintings come from Celeb-A and WikiArt, respectively.

images of the same identity are style-transferred with a different painting) and use the annotations of identity and attributes to train the two branches of our architecture. Sample results of the style transfer performed on Celeb-A are shown in Figure A.10.

WebCaricature [60, 61]. This dataset contains real photos and caricatures of 252 different subjects. For each person, the number of caricatures ranges from 1 to 114, while the number of real photos from 7 to 59. Overall, the dataset contains 6,042 caricatures and 5,974 photos. In our experiments, we use the entire set of caricatures as test set and the real photos as queries to test the effectiveness of our approach in the case of caricatures.

IIIT-D Sketch [9]. It comprises real photos and sketches of 238 different subjects. For each person, a single sketch, drawn by a professional sketch artist, is available. Also for this dataset, we use all 238 sketches as test set and the corresponding real photos as queries, thus testing our solution in the case of sketches.

DeviantArt Faces. This is a dataset that we collected and released as part of this work. It features portraits from the DeviantArt website[3], which contains art images produced by the users, divided in different categories (*e.g.* photography, traditional art, cartoons). On this website, many portraits of well known figures, especially celebrities, are available and can be used for this task. In general, images vary from oil or acrylic painted portraits to sketches, and lack of a photo-realistic quality.

[3] `https://www.deviantart.com/`

To create the dataset, we start from the set of portraits described in [24] containing $1,088$ images, one for each person, collected from DeviantArt. Since only the original image URLs are provided, we discard all identities for which the artistic image is no longer available obtaining a starting set of 617 portraits. We then extend this set of images by collecting new portraits of different people manually downloading them from this website. Overall, we obtain a set of $1,215$ different identities for which a single portrait is collected. For the real counterpart, we download 5 different photos from Google Images for each identity that we manually validate to ensure the identity preservation and avoid errors. Also for this dataset, we use the entire set of portraits as test set and the real photos as queries.

Even though this dataset contains artworks instead of caricatures or sketches, it still cannot be used to fully test our target task, as it deals with a paired setting in which, given a real image, artistic portraits of the same person can be retrieved. Our task, instead, deals with an unpaired setting in which given a real face it is only possible to retrieve similar artistic faces.

WikiArt Faces. To qualitatively validate our solution in a real and unpaired scenario, we also collect a large set of paintings from WikiArt in which at least one face is present. To extract face bounding boxes, we use a deep learning-based face detector [167] that experimentally works well also on less realistic images. Then, we manually validate all detected faces, removing any false positive detection. Overall, we collect $15,116$ artistic faces from paintings of different styles and periods. As previously mentioned, we release both DeviantArt Faces and WikiArt Faces.

Implementation details

Before presenting the experimental evaluations and their corresponding results, we here provide implementation and training details of our face retrieval approach.

Face detection. The first stage of our pipeline consists of the detection of the user face. Similarly, all database images are firstly offline fed through a face detector [167] that extracts a bounding box for each detected face. We apply the same face detector for both artistic and real images as we found face detections to be quite reliable on both domains. In our experiments, we keep all detected faces with a lower edge size of at least 100 pixels. To include the whole head, we extend face bounding boxes by a factor of 0.4.

Style-transferred art proxies. To generate art proxies from real faces, we employ

a pretrained VGG-19 [122] and extract features from layers `conv1_1`, `conv2_1`, `conv3_1`, `conv4_1`, and `conv5_1` to encode the artistic style, and from layer `conv4_2` to encode the content of the real image. The relative weight α of content and style loss is set to 0.02. The generated face I^G is initialized with the realistic face I^R. During the update of I^G, all CNN parameters are kept fixed, and we backpropagate the loss only with respect to the pixels in I^G, using a L-BFGS optimizer [170] and clamping I^G between 0 and 1 at each iteration. We manually check the generated images to ensure that they have sufficient visual quality and repeat the generation with a different artistic image when needed. Noticing that bad visual quality is often associated with an high loss during the generation, we discard a sample and repeat the random choice until the loss is always lower than a threshold. In our experiments, we set this threshold to 40.0 as it generally corresponds to good generated results.

Face embeddings. To extract face feature vectors, we use and compare different face recognition networks, namely: SphereFace [92], VGG-Face [107], LightCNN [155], and VGG-Face-2 [15]. The feature embedding sizes respectively are of 512, 4096, 256, and 2048. For the Light-CNN network, we use its variants composed of 9 and 29 layers. For the VGG-Face-2, instead, we use two different versions of the model: one based on ResNet-50 [52], while the other based on SE-ResNet-50 with squeeze-and-excitation blocks [58]. We employ the implementations provided by the authors, when available.

Attribute prediction. We test the usage of two separate attribute prediction models, one for artistic images and one for real photos, and that of a shared one. In our experiments, we train the two separate models by using real photos and style-transferred images from Celeb-A, each of them annotated with binary attributes. When employing a shared prediction model, instead, we train it on both real and style-transferred images. The network structure consists of $n = 40$ identical branches, one for each facial attribute. Starting from the feature vector coming from a face recognition network, each branch is composed of four fully connected layers having output sizes of 1024, 512, 64, and 2. We use dropout after the first and second layer with a probability of 0.2 and 0.4, respectively. Also for this model, we extract face feature vectors from different face recognition models.

Training details. To train all our models, we employ the Adam optimizer [72]. For face retrieval, we use an initial learning rate of 10^{-5} decreased by a factor of 0.1 when recall metrics on validation set stop improving. When finetuning the whole face embedding model, we instead use a learning rate of 10^{-6}. The margin of the Hinge ranking loss is set to 0.1. For attribute prediction, we employ an

			Celeb-A			WebCaricature			IIIT-D Sketch			DeviantArt Faces		
Face Embedding	**Emb. Size**	**Finetuning**	**R@1**	**R@5**	**R@10**	**R@1**	**R@5**	**R@10**	**R@1**	**R@5**	**R@10**	**R@1**	**R@5**	**R@10**
SphereFace	512	✗	30.2	44.0	50.4	22.9	40.9	49.3	29.9	57.3	67.5	3.0	7.1	10.4
		✓	**33.0**	**46.9**	**52.8**	**26.8**	**45.7**	**54.2**	**31.6**	**62.0**	**69.7**	**3.8**	**9.5**	**13.2**
LightCNN-9L	256	✗	69.2	80.7	83.8	69.0	84.7	88.8	**57.7**	**81.6**	89.7	19.3	31.9	38.7
		✓	**69.6**	**80.8**	**84.2**	**69.3**	**85.0**	**89.0**	**57.7**	81.2	**90.6**	**19.6**	**32.5**	**39.6**
VGG-Face	4096	✗	70.1	82.7	86.2	70.9	87.2	91.6	61.1	86.8	**91.5**	20.5	37.8	46.4
		✓	**77.7**	**88.2**	**90.9**	**75.2**	**89.7**	**93.0**	**61.5**	**87.2**	90.2	**26.5**	**45.3**	**54.3**
LightCNN-29L	256	✗	87.8	92.4	93.6	82.2	91.6	94.1	52.6	75.6	83.8	35.1	51.9	59.4
		✓	**88.6**	**93.2**	**94.1**	**82.5**	**92.5**	**94.8**	**55.1**	**78.6**	**86.3**	**36.3**	**55.1**	**62.9**
VGG-Face-2 (ResNet)	2048	✗	90.5	94.4	95.2	**90.1**	**96.4**	97.6	**66.7**	**88.5**	92.3	45.7	65.4	73.1
		✓	**90.9**	**94.6**	**95.3**	89.6	96.1	**97.7**	63.7	**88.5**	**93.2**	**45.9**	**65.6**	**73.4**
VGG-Face-2 (SE-ResNet)	2048	✗	91.1	94.6	95.3	**91.6**	**97.4**	**98.5**	**60.7**	**87.6**	92.7	47.6	67.0	**74.8**
		✓	**91.5**	**94.8**	**95.5**	91.3	97.2	98.2	58.1	86.3	**93.6**	**47.9**	**67.1**	**74.8**

Table A.4: Face recognition branch performance (Recall@k) on Celeb-A, WebCaricature, IIIT-D Sketch, and DeviantArt Faces.

initial learning rate of 10^{-4} decreased by a factor of 0.8 every 3 epochs. For all our experiments, we use a batch size of 32.

Relevance feedback. The parameters of the relevance feedback algorithms are respectively set to: $\alpha = 0.8$, $\beta = 0.1$, $\gamma = 0.1$, $\eta = 0.5$, $c = 0.8$.

A.2.3 Experimental Results

In the following, we first evaluate the performance of the two branches of our architecture in presence of the artistic domain gap, and then assess the quality of the retrieved results in the final application scenario, also in combination with the user's feedback.

Face retrieval

We evaluate the performance of the embedding space trained on the recognition of face identities, by ablating our approach and removing the face attribute networks. We use real faces as query and artistic or style-transferred images as database items. Table A.4 shows the performances in terms of Recall@k (k=1, 5, 10) when finetuning the whole feature face recognition network and when training only the non-symmetric linear projections with residuals. For completeness, results are reported for all the face recognition backbones we employ and on Celeb-A, WebCaricature, IIIT-D Sketch, and DeviantArt Faces. For fairness, in the case

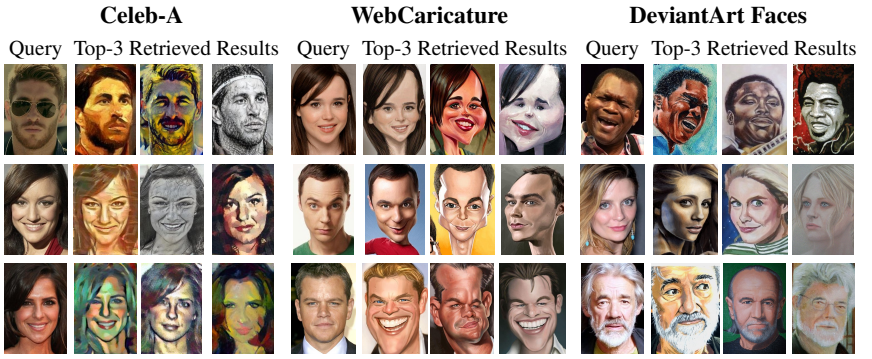


Figure A.11: Top retrieved results from Celeb-A, WebCaricature, and DeviantArt Faces from the face recognition branch ($\lambda = 0$).

of Celeb-A we remove the style-transferred version of the query from the set of retrievable elements.

We observe that our strategy for addressing the domain shift can produce significantly effective results on face recognition (with R@1 greater than 90% on some backbones), and further notice that fine-tuning the backbone is generally beneficial. Analyzing the different backbones, VGG-Face-2 provides the best performance, followed by LightCNN. SphereFace, instead, provides the worse performance on our setting, with a R@1 barely over 30%. As it can be seen, the impact of finetuning the backbone is particularly evident in the case of backbones which have low or medium effectiveness. When comparing the performance of the different datasets, instead, it is noticeable that recalls on IIIT-D Sketch and DeviantArt Faces are generally lower than those on Celeb-A and WebCaricature: this is explained by the smaller average number of relevant items for each query (*i.e.* only one for IIIT-D Sketch and DeviantArt Faces, around 20 for Celeb-A and WebCaricature).

Figure A.11 qualitatively evaluates the performance of the branch by showing the top-3 retrieved results when using queries from Celeb-A, WebCaricature and DeviantArt Faces. As it can be seen, the branch is capable of preserving the identity of the query most of the times, although as expected it sometimes fails to preserve the facial attributes of the query.

Model	Training Set	Accuracy (%)	
		Real photos	Style transfer
VGG-Face	Real	**88.9**	85.2
VGG-Face-2 (ResNet)		88.5	**86.3**
VGG-Face-2 (SE-ResNet)		87.7	85.8
VGG-Face	Style Transfer	**88.7**	**87.2**
VGG-Face-2 (ResNet)		87.9	87.0
VGG-Face-2 (SE-ResNet)		87.1	86.3
VGG-Face	Joint	**88.8**	**87.0**
VGG-Face-2 (ResNet)		88.4	87.0
VGG-Face-2 (SE-ResNet)		87.4	86.4

Table A.5: Facial attribute prediction accuracy on Celeb-A.

Face retrieval using semantic attributes

In Table A.5, we report the accuracy of the attribute prediction branch of our architecture on Celeb-A. We evaluate the performance of the two attribute prediction networks (one trained on real faces, one on style-transferred versions) on both real and artistic images, and test also the usage of a shared attribute prediction network for both real and artistic images. While VGG-Face tends to perform better in almost all settings, we notice that the obtained results are positive in terms of overall accuracy and always greater than 85% for both real and artistic settings and with all considered backbones. Further, we also notice that using a shared backbone does not decrease performances significantly.

Further, in Table A.6 we employ our full architecture and evaluate the role of the two branches by varying the relative weight of the face recognition and the attribute prediction branches. Also in this case we test on Celeb-A, WebCaricature, IIIT-D Sketch, and DeviantArt Faces, and report Recall@1 for face recognition, as well as the attribute accuracy of the first retrieved element with respect to the query. As it can be seen, increasing λ up to 0.2 generally leads to boosting the attribute accuracy of the retrieved elements, without significantly loosing in terms of face recognition, thus obtaining better quality results in terms of the final similarity and validating the appropriateness of combining face recognition and attribute prediction.

To validate the weighted retrieval approach on our target setting (*i.e.* that of a real museum), in Figure A.12 we use queries from the test set of Celeb-A and retrieve artistic faces coming from WikiArt. As it can be noticed, $\lambda = 0.2$ and

	$\lambda = 0.0$		$\lambda = 0.1$		$\lambda = 0.2$		$\lambda = 0.5$		$\lambda = 1.0$		$\lambda = 5.0$		$\lambda = 10.0$	
Face Embedding	**R@1**	**Acc.**	**R@1**	**Acc.**	**R@1**	**Acc.**	**R@1**	**Acc.**	**R@1**	**Acc.**	**R@1**	**Acc.**	**R@1**	**Acc.**
	Celeb-A													
SphereFace	33.0	85.4	34.1	87.7	32.8	88.9	26.0	90.5	17.3	91.3	6.2	91.7	6.1	91.7
LightCNN-9L	69.6	86.5	70.0	87.4	69.5	88.1	63.6	89.3	52.3	90.4	13.5	91.8	9.1	91.9
VGG-Face	77.7	86.8	77.4	87.8	75.7	88.5	68.0	89.6	54.0	90.6	13.6	91.8	9.9	91.8
LightCNN-29	88.6	86.8	88.6	87.6	88.1	88.1	84.2	89.1	74.4	90.0	21.1	91.8	10.8	91.8
VGG-Face-2 (ResNet)	90.9	86.9	90.7	87.6	90.1	88.2	85.5	89.2	73.7	90.2	17.9	91.8	10.6	91.8
VGG-Face-2 (SE-ResNet)	91.5	86.7	91.2	87.6	90.5	88.2	85.7	89.3	73.8	90.3	17.7	91.9	10.7	91.9
	WebCaricature													
SphereFace	26.8	85.0	30.2	89.3	30.6	91.5	26.4	94.6	19.0	96.4	10.9	97.4	10.9	97.4
LightCNN-9L	69.3	86.8	69.9	88.5	69.9	89.7	64.8	92.1	53.0	94.3	17.7	97.4	15.1	97.4
VGG-Face	75.2	87.6	75.5	89.3	73.9	90.5	67.3	92.8	54.3	94.7	19.6	97.3	16.7	97.4
LightCNN-29	82.5	87.1	82.7	88.5	82.6	89.5	77.5	91.7	65.0	93.8	22.5	97.3	16.9	97.4
VGG-Face-2 (ResNet)	89.6	87.5	89.3	88.8	88.2	89.8	81.2	92.0	67.2	94.2	20.6	97.4	17.5	97.4
VGG-Face-2 (SE-ResNet)	91.3	87.5	90.7	88.9	89.8	89.9	83.7	92.1	69.0	94.2	21.5	97.3	17.8	97.4
	IIIT-D Sketch													
SphereFace	31.6	84.0	35.9	84.6	37.6	88.2	31.6	80.6	24.4	92.1	8.5	93.3	8.5	93.3
LightCNN-9L	57.7	84.0	62.0	85.8	62.4	87.0	56.0	88.7	38.9	90.8	11.5	93.2	9.0	93.3
VGG-Face	61.5	85.3	62.0	86.3	62.4	87.1	56.8	88.6	43.2	90.4	9.8	93.2	7.7	93.3
LightCNN-29	55.1	84.5	54.7	85.8	56.4	87.1	50.4	88.7	37.6	90.6	11.5	93.2	8.5	93.3
VGG-Face-2 (ResNet)	63.7	85.0	64.1	86.0	63.2	87.0	52.6	89.2	35.0	91.4	9.4	93.3	8.1	93.3
VGG-Face-2 (SE-ResNet)	58.1	84.9	61.5	86.2	59.8	87.0	51.7	89.0	36.3	91.3	9.8	93.3	8.5	93.3
	DeviantArt Faces													
SphereFace	3.8	84.1	4.8	88.6	5.0	91.0	4.0	94.3	2.7	95.8	1.6	96.5	1.6	96.5
LightCNN-9L	19.6	85.1	20.3	87.6	20.2	89.0	17.4	91.9	11.6	94.2	2.9	96.5	2.4	96.5
VGG-Face	26.5	86.5	26.3	88.2	25.3	89.6	20.2	92.2	13.4	94.3	3.4	96.5	2.7	96.5
LightCNN-29	36.3	85.7	36.6	87.3	35.8	88.5	30.2	91.2	20.0	93.6	4.2	96.5	2.8	96.5
VGG-Face-2 (ResNet)	45.9	86.0	45.6	87.5	43.6	88.8	34.8	91.5	21.5	94.2	3.7	96.5	2.9	96.5
VGG-Face-2 (SE-ResNet)	47.9	86.0	47.3	87.5	45.1	88.7	35.2	91.4	21.8	94.0	3.9	96.5	3.0	96.5

Table A.6: Weighted retrieval results on Celeb-A, WebCaricature, IIIT-D Sketch, and DeviantArt Faces. Results are reported for different λ values in terms of recall and attribute preservation accuracy.

$\lambda = 0.5$ offer the best result in terms of attribute preservation without significant loss in terms of the overall face similarity. Retrieved paintings generally look similar to the corresponding real queries and tend to preserve most of the facial attributes, in addition to face appearance.

Additionally, we quantitatively evaluate how attributes are preserved in retrieved results. In Table A.7, we retrieve elements from WikiArt Faces using our weighted retrieval and queries from the test set of Celeb-A, and measure the attribute accuracy between the query and the top-1 retrieved element. As it can be seen, increasing λ effectively increases the preservation of attributes also in the case of real paintings.

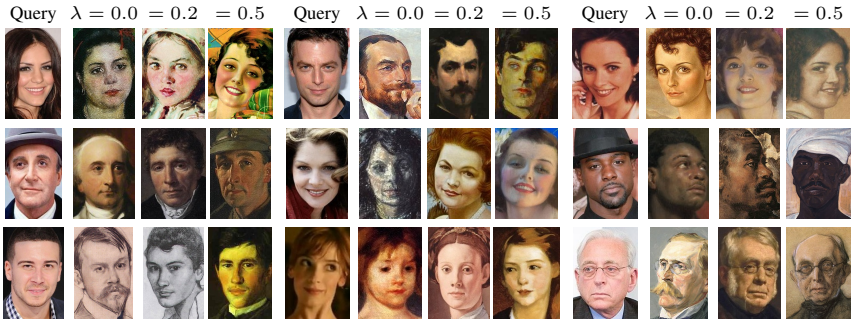


Figure A.12: Top-1 retrieved element from WikiArt Faces using weighted retrieval and different values of λ.

	Accuracy for different λ values						
Face Embedding	**0.0**	**0.1**	**0.2**	**0.5**	**1.0**	**5.0**	**10.0**
SphereFace	83.3	87.3	89.9	94.0	95.9	96.7	96.7
LightCNN-9	83.1	85.8	87.8	91.5	94.3	96.7	96.7
VGG-Face	83.7	86.4	88.4	92.1	94.5	96.6	96.7
LightCNN-29	83.1	85.4	87.3	91.2	94.0	96.6	96.7
VGG-Face-2 (ResNet)	83.1	85.5	87.6	92.0	94.7	96.7	96.7
VGG-Face-2 (SE-ResNet)	83.1	85.3	87.2	91.0	93.7	96.7	96.7

Table A.7: Top-1 attribute accuracy on WikiArt Faces using different values of λ.

Relevance feedback and user interaction

Experimental results provided so far have shown that, when a good face recognition backbone is chosen, the first retrieved element is generally similar enough to the query. However, as the user is presented with the top-k retrieved elements in our application scenario, we also need to assess the quality of the complete ranking. For this reason, we evaluate the Average Precision (AP) of the predicted ranking, and how it changes when the user interacts with the relevance feedback algorithm.

In the following experiments, we use a subset of the Celeb-A test set composed of a single randomly selected image for each identity (*i.e.* 989 query images) and all style-transferred images of the test set as retrievable items (*i.e.* around $18,000$ images). For WebCaricature, we instead perform the experiments on the whole dataset. To simulate user interaction, at each iteration we provide the relevance

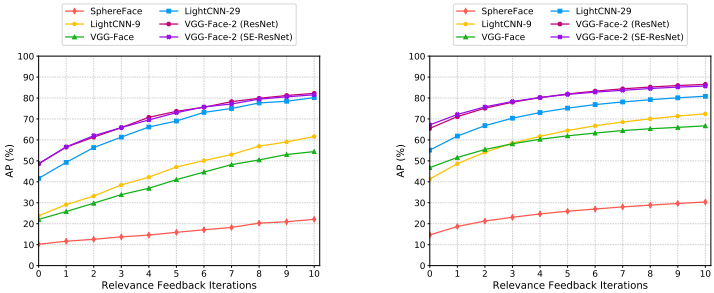
	Parameters					VGG-Face-2 (ResNet)			VGG-Face-2 (SE-ResNet)		
	α	β	γ	η	c	1	5	10	1	5	10
Celeb-A											
QPM	0.0	1.0	0.0	0.0	0.0	56.41	56.20	55.98	59.68	59.64	60.15
FSW	1.0	0.0	0.0	0.5	0.8	56.23	62.74	67.24	**60.02**	64.23	68.32
Ours	0.8	0.1	0.1	0.5	0.8	**56.56**	**73.63**	**82.23**	56.68	**73.00**	**81.53**
WebCaricature											
QPM	0.0	1.0	0.0	0.0	0.0	47.07	46.60	46.79	50.17	50.36	50.72
FSW	1.0	0.0	0.0	0.5	0.8	58.94	62.15	65.67	61.49	64.34	67.43
Ours	0.8	0.1	0.1	0.5	0.8	**71.10**	**81.93**	**86.51**	**72.13**	**81.73**	**85.71**

Table A.8: Ablation study on relevance feedback hyper-parameters.

feedback algorithm with one positive item (randomly selected among images which share the same identity with the query) and one negative item (randomly selected among images which do not share the same identity with the query).

First, we evaluate the proposed relevance feedback model through an ablation study. In Table A.8 we report the Average Precision (AP) of the predicted ranking after 1, 5 and 10 relevance feedback iterations, when employing three different variants of the proposed strategy: (1) Query Point Movement (QPM), where at each iteration a new query is reformulated as the mean of positive feedbacks, that corresponds to $\alpha = 0$, $\beta = 1.0$, $\gamma = 0$, $\eta = 0$; (2) Feature Space Warping (FSW) without changing the query point, corresponding to $\alpha = 1.0$, $\beta = 0$, $\gamma = 0$; (3) the full proposal, with both Query Point Movement and Feature Space Warping. The experiments are conducted on both Celeb-A and WebCaricature datasets and by using the two versions of VGG-Face-2 (*i.e.* ResNet and SE-ResNet). Our complete strategy leads to the best performance according to both datasets and backbones, with an overall AP over 80% after 10 iterations. On the contrary, when employing only one of the relevance feedback components, the final AP is generally lower than 60% and 70% for the QPM and FSW models respectively.

Then, in Figure A.13, we show the AP of different backbones on Celeb-A and WebCaricature, after a variable number of relevance feedback iterations. As it can be seen, the two versions of VGG-Face-2 obtain the best results on both datasets, with an initial AP of 48.5% (ResNet) and 48.6% (SE-ResNet) on Celeb-A, and 65.5% (ResNet) and 67.2% (SE-ResNet) on WebCaricature. In both cases, the initial AP is further increased by 15 points after four relevance feedback iterations, reaching more than 70% and 80% on Celeb-A and WebCaricature respectively.



(a) Celeb-A (b) WebCaricature

Figure A.13: Relevance feedback evaluation with different backbones on Celeb-A and WebCaricature ($\lambda = 0.2$).

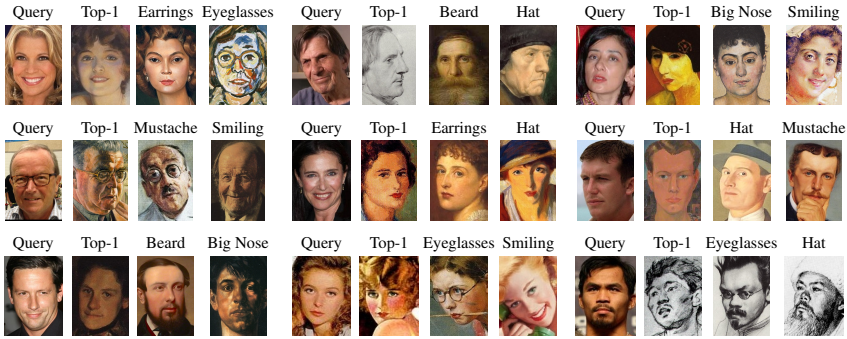


Figure A.14: Weighted retrieval results with attribute constraints ($\lambda = 0.2$).

Additionally, in Figure A.14 we show some qualitative examples of using attributes as a further constraint to retrieval. In this case, the user is presented with the list of all possible attributes and can constraint the retrieval so that the retrieved set contains samples with a given attribute.

Appendix B

List of publications

1. Matteo Tomei, Lorenzo Baraldi, Marcella Cornia, and Rita Cucchiara. What was Monet seeing while painting? Translating artworks to photo-realistic images. In *Proceedings of the European Conference on Computer Vision Workshops*, 2018.

2. Matteo Tomei, Marcella Cornia, Lorenzo Baraldi, and Rita Cucchiara. Art2real: Unfolding the reality of artworks via semantically-aware image-to-image translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

3. Matteo Tomei, Marcella Cornia, Lorenzo Baraldi, and Rita Cucchiara. Image-to-image translation to unfold the reality of artworks: an empirical analysis. In *Proceedings of the International Conference on Image Analysis and Processing*. Springer, 2019.

4. Matteo Tomei, Lorenzo Baraldi, Simone Calderara, Simone Bronzin, and Rita Cucchiara. Rms-net: Regression and masking for soccer event spotting. In *Proceedings of the International Conference on Pattern Recognition*. IEEE, 2020.

5. Matteo Tomei, Lorenzo Baraldi, Simone Calderara, Simone Bronzin, and Rita Cucchiara. Video action detection by learning graph-based spatio-temporal interactions. *Computer Vision and Image Understanding*, 206:103187, 2021.

6. Matteo Tomei, Lorenzo Baraldi, Simone Bronzin, and Rita Cucchiara. Estimating (and fixing) the effect of face obfuscation in video recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2021.

7. Matteo Tomei, Lorenzo Baraldi, Giuseppe Fiameni, Simone Bronzin, and Rita Cucchiara. A computational approach for progressive architecture shrinkage in action recognition. *Software: Practice and Experience*, 2021.

8. Marcella Cornia, Matteo Tomei, Lorenzo Baraldi, and Rita Cucchiara. Matching faces and attributes between the artistic and the real domain: the personart approach. *ACM Transactions on Multimedia Computing, Communications and Applications*, 2021.

Bibliography

[1] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 109

[2] Sule Anjomshoae, Amro Najjar, Davide Calvaresi, and Kary Främling. Explainable agents and robots: Results from a systematic literature review. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems*, 2019. 120

[3] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. *arXiv preprint arXiv:2103.15691*, 2021. 11, 73

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 25

[5] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Action classification in soccer videos with long short-term memory recurrent neural networks. In *International Conference on Artificial Neural Networks*, pages 154–159. Springer, 2010. 15

[6] Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. Frozen in time: A joint video and image encoder for end-to-end retrieval. *arXiv preprint arXiv:2104.00650*, 2021. 11

[7] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning*, 2021. 11

[8] Shweta Bhardwaj, Mukundhan Srinivasan, and Mitesh M Khapra. Efficient video classification using fewer frames. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 12

[9] Himanshu S Bhatt, Samarth Bharadwaj, Richa Singh, and Mayank Vatsa. Memetically optimized MCWLD for matching sketches with digital face images. *IEEE Trans. on Information Forensics and Security*, 7(5):1522–1535, 2012. 122

[10] Hakan Bilen, Basura Fernando, Efstratios Gavves, Andrea Vedaldi, and Stephen Gould. Dynamic image networks for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 16

[11] Aaron F. Bobick and James W. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(3), 2001. 17

[12] Daniele Borghesani, Costantino Grana, and Rita Cucchiara. Miniature illustrations retrieval and innovative interaction for digital illuminated manuscripts. *Multimedia Systems*, 20(1):65–79, 2014. 120

[13] William Brendel and Sinisa Todorovic. Learning spatiotemporal graphs of human activities. In *Proceedings of the International Conference on Computer Vision*, 2011. 14

[14] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 13

[15] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. VGGFace2: A Dataset for Recognising Faces Across Pose and Age. In *Proceedings of the IEEE International Conference on Automatic Face & Gesture Recognition*, 2018. 124

[16] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proceedings of the European Conference on Computer Vision*, 2020. 43

[17] Joao Carreira, Viorica Patraucean, Laurent Mazare, Andrew Zisserman, and Simon Osindero. Massively parallel video networks. In *Proceedings of the European Conference on Computer Vision*, 2018. 16

[18] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 10, 26, 29, 30, 36, 88

[19] Yao-Jen Chang, Keisuke Kamataki, and Tsuhan Chen. Mean shift feature space warping for relevance feedback. In *Proceedings of the International Conference on Image Processing*, 2009. 120

[20] Yunpeng Chen, Haoqi Fan, Bing Xu, Zhicheng Yan, Yannis Kalantidis, Marcus Rohrbach, Shuicheng Yan, and Jiashi Feng. Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. In *Proceedings of the International Conference on Computer Vision*, 2019. 88

[21] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. Multi-fiber networks for video recognition. In *Proceedings of the European Conference on Computer Vision*, 2018. 16, 88

[22] Anthony Cioppa, Adrien Deliege, Silvio Giancola, Bernard Ghanem, Marc Van Droogenbroeck, Rikke Gade, and Thomas B Moeslund. A context-aware loss function for action spotting in soccer videos. In *cvpr*, 2020. 15, 39, 43, 47

[23] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *Proceedings of the International Conference on Learning Representations*, 2016. 22

[24] Elliot J Crowley, Omkar M Parkhi, and Andrew Zisserman. Face Painting: Querying Art with Photos. In *Proceedings of the British Machine Vision Conference*, 2015. 114, 123

[25] Ji Dai, Behrouz Saghafi, Jonathan Wu, Janusz Konrad, and Prakash Ishwar. Towards privacy-preserving recognition of human activities. In *Proceedings of the International Conference on Image Processing*. IEEE, 2015. 12

[26] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in Neural Information Processing Systems*, 2016. 14

[27] Adrien Deliege, Anthony Cioppa, Silvio Giancola, Meisam J Seikavandi, Jacob V Dueholm, Kamal Nasrollahi, Bernard Ghanem, Thomas B Moeslund, and Marc Van Droogenbroeck. Soccernet-v2: A dataset and benchmarks for holistic understanding of broadcast soccer videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2021. 15, 54

[28] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 10, 12, 26, 45, 54, 109

[29] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 40, 43

[30] Ali Diba, Mohsen Fayyaz, Vivek Sharma, M Mahdi Arzani, Rahman Yousefzadeh, Juergen Gall, and Luc Van Gool. Spatio-temporal channel correlation networks for action classification. In *Proceedings of the European Conference on Computer Vision*, 2018. 16

[31] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations*, 2021. 11

[32] Victor Escorcia, Cuong D Dao, Mihir Jain, Bernard Ghanem, and Cees Snoek. Guess where? actor-supervision for spatiotemporal action localization. *Computer Vision and Image Understanding*, 192, 2020. 13

[33] Victor Escorcia and Juan Niebles. Spatio-temporal human-object interactions for action recognition in videos. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2013. 13

[34] Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. VSE++: Improving visual-semantic embeddings with hard negatives. In *Proceedings of the British Machine Vision Conference*, 2018. 118

[35] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 16, 73, 77, 78, 87

[36] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the International Conference on Computer Vision*, 2019. 10, 13, 17, 26, 27, 29, 30, 35, 36, 66, 70, 71, 73, 75, 77, 82, 83, 84, 85, 88, 90, 93

[37] Christoph Feichtenhofer, Axel Pinz, and Richard Wildes. Spatiotemporal residual networks for video action recognition. In *Advances in Neural Information Processing Systems*, 2016. 10

[38] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 10

[39] Yushu Feng, Huan Wang, Haoji Roland Hu, Lu Yu, Wei Wang, and Shiyan Wang. Triplet distillation for deep face recognition. In *Proceedings of the International Conference on Image Processing*. IEEE, 2020. 12, 65

[40] Nuno C Garcia, Sarah Adel Bargal, Vitaly Ablavsky, Pietro Morerio, Vittorio Murino, and Stan Sclaroff. Dmcl: Distillation multiple choice learning for multimodal action recognition. *arXiv preprint arXiv:1912.10982*, 2019. 12

[41] Nuno C Garcia, Pietro Morerio, and Vittorio Murino. Modality distillation with multiple stream networks for action recognition. In *Proceedings of the European Conference on Computer Vision*, 2018. 12

[42] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015. 107

[43] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 116

[44] Silvio Giancola, Mohieddine Amine, Tarek Dghaily, and Bernard Ghanem. Soccernet: A scalable dataset for action spotting in soccer videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018. 15, 39, 41, 45, 46, 47, 48, 49, 54

[45] Rohit Girdhar, João Carreira, Carl Doersch, and Andrew Zisserman. A better baseline for ava. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018. 29, 35

[46] Rohit Girdhar, Joao Carreira, Carl Doersch, and Andrew Zisserman. Video action transformer network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 13, 19, 29, 35

[47] Georgia Gkioxari and Jitendra Malik. Finding action tubes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 13, 31, 32

[48] Chunhui Gu, Chen Sun, David A Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, et al. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 13, 19, 25, 26, 29, 35

[49] Abhinav Gupta and Larry S Davis. Objects in action: An approach for combining action understanding and object perception. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007. 13

[50] Abhinav Gupta, Aniruddha Kembhavi, and Larry S Davis. Observing human-object interactions: Using spatial and functional compatibility for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(10), 2009. 13

[51] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the International Conference on Computer Vision*, 2017. 26, 59, 60, 111

[52] 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*, 2016. 15, 26, 45, 54, 67, 83, 109, 111, 124

[53] Byeongho Heo, Jeesoo Kim, Sangdoo Yun, Hyojin Park, Nojun Kwak, and Jin Young Choi. A comprehensive overhaul of feature distillation. In *Proceedings of the International Conference on Computer Vision*, 2019. 12, 80, 91, 92

[54] Samitha Herath, Mehrtash Harandi, and Fatih Porikli. Going deeper into action recognition: A survey. *Image and vision computing*, 60:4–21, 2017. 17

[55] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a Nash equilibrium. *Advances in Neural Information Processing Systems*, 2017. 106, 107

[56] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *Proceedings of Neural Information Processing Systems Workshops*, 2015. 12, 64, 79

[57] Rui Hou, Chen Chen, and Mubarak Shah. Tube convolutional neural network (t-cnn) for action detection in videos. In *Proceedings of the International Conference on Computer Vision*, 2017. 13

[58] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 124

[59] Ronghang Hu, Piotr Dollár, Kaiming He, Trevor Darrell, and Ross Girshick. Learning to Segment Every Thing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 101, 102, 109

[60] Jing Huo, Yang Gao, Yinghuan Shi, and Hujun Yin. Variation robust cross-modal metric learning for caricature recognition. In *Proceedings of the ACM International Conference on Multimedia Workshops*, 2017. 122

[61] Jing Huo, Wenbin Li, Yinghuan Shi, Yang Gao, and Hujun Yin. Webcaricature: a benchmark for caricature face recognition. In *Proceedings of the British Machine Vision Conference*, 2018. 122

[62] Noureldien Hussein, Efstratios Gavves, and Arnold WM Smeulders. Timeception for complex action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 11

[63] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 106

[64] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 14

[65] Hueihan Jhuang, Juergen Gall, Silvia Zuffi, Cordelia Schmid, and Michael J Black. Towards understanding action recognition. In *Proceedings of the International Conference on Computer Vision*, 2013. 13, 19, 25, 31

[66] Haohao Jiang, Yao Lu, and Jing Xue. Automatic soccer video event detection based on a deep neural network combined cnn and rnn. In *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2016. 15

[67] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734*, 2017. 104

[68] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of the European Conference on Computer Vision*, 2016. 106

[69] Vicky Kalogeiton, Philippe Weinzaepfel, Vittorio Ferrari, and Cordelia Schmid. Action tubelet detector for spatio-temporal action localization. In *Proceedings of the International Conference on Computer Vision*, 2017. 13

[70] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 26, 58, 59, 60, 66, 75, 81

[71] Kyungyul Kim, ByeongMoon Ji, Doyoung Yoon, and Sangheum Hwang. Self-knowledge distillation: A simple way for better generalization. *arXiv preprint arXiv:2006.12000*, 2020. 12

[72] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2015. 27, 106, 124

[73] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *Proceedings of the International Conference on Learning Representations*, 2017. 14

[74] Okan Köpüklü, Neslihan Kose, Ahmet Gunduz, and Gerhard Rigoll. Resource efficient 3d convolutional neural networks. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019. 15

[75] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73, 2017. 26, 109, 111

[76] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *Proceedings of the International Conference on Computer Vision*, 2011. 75, 82

[77] Ivan Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2-3):107–123, 2005. 17

[78] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 106

[79] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Kumar Singh, and Ming-Hsuan Yang. Diverse Image-to-Image Translation via Disentangled Representations. In *Proceedings of the European Conference on Computer Vision*, 2018. 107, 108, 109, 110, 111, 112

[80] Chao Li, Qiaoyong Zhong, Di Xie, and Shiliang Pu. Collaborative spatiotemporal feature learning for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 11

[81] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *Proceedings of the European Conference on Computer Vision*, 2016. 106

[82] Dong Li, Zhaofan Qiu, Qi Dai, Ting Yao, and Tao Mei. Recurrent tubelet proposal and recognition networks for action detection. In *Proceedings of the European Conference on Computer Vision*, 2018. 29

[83] Jian Li, Yabiao Wang, Changan Wang, Ying Tai, Jianjun Qian, Jian Yang, Chengjie Wang, Jilin Li, and Feiyue Huang. Dsfd: Dual shot face detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 59, 60

[84] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the International Conference on Computer Vision*, 2019. 10, 16, 73, 88

[85] Tianwei Lin, Xiao Liu, Xin Li, Errui Ding, and Shilei Wen. Bmn: Boundary-matching network for temporal action proposal generation. In *Proceedings of the International Conference on Computer Vision*, 2019. 13

[86] Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang, and Ming Yang. Bsn: Boundary sensitive network for temporal action proposal generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 13

[87] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 26

[88] 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 *Proceedings of the European Conference on Computer Vision*, 2014. 26, 111

[89] Zhijie Lin, Zhou Zhao, Zhu Zhang, Zijian Zhang, and Deng Cai. Moment retrieval via cross-modal interaction networks with query reconstruction. *IEEE Transactions on Image Processing*, 29, 2020. 57

[90] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*, 2017. 107, 108, 109, 110, 111, 112

[91] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision*, 2016. 13

[92] 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. 124

[93] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep Learning Face Attributes in the Wild. In *Proceedings of the International Conference on Computer Vision*, 2015. 106, 121

[94] Chenxu Luo and Alan L Yuille. Grouped spatial-temporal aggregation for efficient action recognition. In *Proceedings of the International Conference on Computer Vision*, 2019. 16

[95] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008. 111

[96] Roey Mechrez, Itamar Talmi, Firas Shama, and Lihi Zelnik-Manor. Learning to Maintain Natural Image Statistics. *arXiv preprint arXiv:1803.04626*, 2018. 104

[97] Yue Meng, Chung-Ching Lin, Rameswar Panda, Prasanna Sattigeri, Leonid Karlinsky, Aude Oliva, Kate Saenko, and Rogerio Feris. Ar-net: Adaptive frame resolution for efficient action recognition. In *Proceedings of the European Conference on Computer Vision*, 2020. 15

[98] Pascal Mettes and Cees GM Snoek. Spatial-aware object embeddings for zero-shot localization and classification of actions. In *Proceedings of the International Conference on Computer Vision*, 2017. 13

[99] Pascal Mettes and Cees GM Snoek. Pointly-supervised action localization. *International Journal of Computer Vision*, 127(3):263–281, 2019. 13

[100] Pascal Mettes, Jan C Van Gemert, and Cees GM Snoek. Spot on: Action localization from pointly-supervised proposals. In *Proceedings of the European Conference on Computer Vision*. Springer, 2016. 13

[101] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *Proceedings of the Conference on Artificial Intelligence*, 2020. 12

[102] Hossein Mobahi, Mehrdad Farajtabar, and Peter L Bartlett. Self-distillation amplifies regularization in hilbert space. *Advances in Neural Information Processing Systems*, 2020. 70

[103] Daniel Neimark, Omri Bar, Maya Zohar, and Dotan Asselmann. Video transformer network. *arXiv preprint arXiv:2102.00719*, 2021. 11

[104] Junting Pan, Siyu Chen, Mike Zheng Shou, Yu Liu, Jing Shao, and Hongsheng Li. Actor-context-actor relation network for spatio-temporal action localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 14

[105] Luca Pappalardo, Paolo Cintia, Alessio Rossi, Emanuele Massucco, Paolo Ferragina, Dino Pedreschi, and Fosca Giannotti. A public data set of spatio-temporal match events in soccer competitions. *Scientific data*, 6(1):1–15, 2019. 14

[106] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 12, 65

[107] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep Face Recognition. In *Proceedings of the British Machine Vision Conference*, 2015. 124

[108] AJ Piergiovanni and Michael S Ryoo. Avid dataset: Anonymized videos from diverse countries. *Advances in Neural Information Processing Systems*, 2020. 12

[109] Alessandro Prest, Vittorio Ferrari, and Cordelia Schmid. Explicit modeling of human-object interactions in realistic videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(4), 2012. 13

[110] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *Proceedings of the International Conference on Computer Vision*, 2017. 10

[111] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015. 13, 26, 41

[112] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017. 109, 111

[113] Zhongzheng Ren, Yong Jae Lee, and Michael S Ryoo. Learning to anonymize faces for privacy preserving action detection. In *Proceedings of the european conference on computer vision (ECCV)*, 2018. 12

[114] Olav A Norgård Rongved, Steven A Hicks, Vajira Thambawita, Håkon K Stensland, Evi Zouganeli, Dag Johansen, Michael A Riegler, and Pål Halvorsen. Real-time detection of events in soccer videos using 3d convolutional neural networks. In *2020 IEEE International Symposium on Multimedia (ISM)*, pages 135–144. IEEE, 2020. 15

[115] Michael Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. Tokenlearner: Adaptive space-time tokenization for videos. *Advances in Neural Information Processing Systems*, 2021. 34

[116] Suman Saha, Gurkirt Singh, and Fabio Cuzzolin. Amtnet: Action-micro-tube regression by end-to-end trainable deep architecture. In *Proceedings of the International Conference on Computer Vision*, 2017. 13, 31, 32

[117] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the International Conference on Computer Vision*, 2017. 120

[118] Gilad Sharir, Asaf Noy, and Lihi Zelnik-Manor. An image is worth 16x16 words, what is a video worth? *arXiv preprint arXiv:2103.13915*, 2021. 11

[119] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *Proceedings of the European Conference on Computer Vision*, 2016. 13

[120] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Proceedings of the International Conference on Learning Representations Workshops*, 2014. 120

[121] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, 2014. 10, 57, 73

[122] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 47, 124

[123] Khurram Soomro and Amir R Zamir. Action recognition in realistic sports videos. In *Computer vision in sports*, pages 181–208. Springer, 2014. 13

[124] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 13, 19, 26, 31, 75, 82

[125] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. In *Proceedings of the International Conference on Learning Representations Workshops*, 2015. 120

[126] Chen Sun, Abhinav Shrivastava, Carl Vondrick, Kevin Murphy, Rahul Sukthankar, and Cordelia Schmid. Actor-centric relation network. In *Proceedings of the European Conference on Computer Vision*, 2018. 17, 29, 35

[127] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the International Conference on Machine Learning*, 2017. 120, 121

[128] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 107

[129] Fida Mohammad Thoker and Juergen Gall. Cross-modal knowledge distillation for action recognition. In *Proceedings of the International Conference on Image Processing*, 2019. 12

[130] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. *arXiv preprint arXiv:1910.10699*, 2019. 12, 65

[131] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the International Conference on Computer Vision*, 2015. 10, 73

[132] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *Proceedings of the International Conference on Computer Vision*, 2019. 15, 77, 88

[133] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 10, 17, 66, 67, 68, 69, 70, 71, 73, 75, 77, 82, 84, 87, 88, 90

[134] Grigorios Tsagkatakis, Mustafa Jaber, and Panagiotis Tsakalides. Goal!! event detection in sports video. *Electronic Imaging*, 2017(16):15–20, 2017. 15

[135] Ries Uittenbogaard, Clint Sebastian, Julien Vijverberg, Bas Boom, Dariu M Gavrila, et al. Privacy protection in street-view panoramas using depth and multi-view imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 12

[136] Oytun Ulutan, Swati Rallapalli, Mudhakar Srivatsa, Carlos Torres, and BS Manjunath. Actor conditioned attention maps for video action detection. In *The IEEE Winter Conference on Applications of Computer Vision*, 2020. 13, 17, 18, 19, 29

[137] Bastien Vanderplaetse and Stephane Dupont. Improved soccer action spotting using both audio and video streams. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2020. 15, 47

[138] Gül Varol, Ivan Laptev, and Cordelia Schmid. Long-term temporal convolutions for action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6):1510–1517, 2017. 10

[139] Leonid Nisonovich Vaserstein. Markov processes over denumerable products of spaces, describing large systems of automata. *Problemy Peredachi Informatsii*, 5(3):64–72, 1969. 107

[140] 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. 11, 14, 34

[141] Kanav Vats, Mehrnaz Fani, Pascale Walters, David A Clausi, and John Zelek. Event detection in coarsely annotated sports videos via parallel multi-receptive field 1d convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2020. 15, 39, 47, 49

[142] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *Proceedings of the International Conference on Learning Representations*, 2018. 14, 21

[143] Roberto Vezzani, Massimo Piccardi, and Rita Cucchiara. An efficient bayesian framework for on-line action recognition. In *2009 16th IEEE International Conference on Image Processing*. IEEE, 2009. 17

[144] Chen Wang, Hui Liu, et al. Comprehensive soccer video understanding: Towards human-comparable video understanding system in constrained environment. *arXiv preprint arXiv:1912.04465*, 2019. 14

[145] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *Proceedings of the European Conference on Computer Vision*, 2016. 10

[146] Lin Wang and Kuk-Jin Yoon. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *arXiv preprint arXiv:2004.05937*, 2020. 12

[147] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 11, 29, 33, 73, 77, 88

[148] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *Proceedings of the European Conference on Computer Vision*, 2018. 14, 19

[149] Zihao W Wang, Vibhav Vineet, Francesco Pittaluga, Sudipta N Sinha, Oliver Cossairt, and Sing Bing Kang. Privacy-preserving action recognition using coded aperture videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019. 12

[150] Stephen J Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015. 77

[151] Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krahenbuhl, and Ross Girshick. Long-term feature banks for detailed video understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 13, 17, 19, 26, 27, 29, 30, 33, 35, 36

[152] Chao-Yuan Wu, Ross Girshick, Kaiming He, Christoph Feichtenhofer, and Philipp Krahenbuhl. A multigrid method for efficiently training video models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 16

[153] Chao-Yuan Wu and Philipp Krahenbuhl. Towards long-form video understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 11, 73

[154] Chao-Yuan Wu, Manzil Zaheer, Hexiang Hu, R Manmatha, Alexander J Smola, and Philipp Krähenbühl. Compressed video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 16

[155] Xiang Wu, Ran He, Zhenan Sun, and Tieniu Tan. A Light CNN for Deep Face Representation with Noisy Labels. *IEEE Transactions on Information Forensics and Security*, 13(11):2884–2896, 2018. 124

[156] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 26

[157] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision*, 2018. 11, 15, 73

[158] Huijuan Xu, Abir Das, and Kate Saenko. R-c3d: Region convolutional 3d network for temporal activity detection. In *Proceedings of the International Conference on Computer Vision*, 2017. 13

[159] Kaiyu Yang, Jacqueline Yau, Li Fei-Fei, Jia Deng, and Olga Russakovsky. A study of face obfuscation in imagenet. *arXiv preprint arXiv:2103.06191*, 2021. 12, 57

[160] Xitong Yang, Xiaodong Yang, Ming-Yu Liu, Fanyi Xiao, Larry S Davis, and Jan Kautz. Step: Spatio-temporal progressive learning for video action detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 17, 29

[161] Junqing Yu, Aiping Lei, Zikai Song, Tingting Wang, Hengyou Cai, and Na Feng. Comprehensive dataset of broadcast soccer videos. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, 2018. 14

[162] Li Yuan, Francis EH Tay, Guilin Li, Tao Wang, and Jiashi Feng. Revisiting knowledge distillation via label smoothing regularization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 12

[163] Kaiyu Yue, Jiangfan Deng, and Feng Zhou. Matching guided distillation. In *Proceedings of the European Conference on Computer Vision*, 2020. 12

[164] Sukmin Yun, Jongjin Park, Kimin Lee, and Jinwoo Shin. Regularizing class-wise predictions via self-knowledge distillation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 70

[165] Xuefan Zha, Wentao Zhu, Tingxun Lv, Sen Yang, and Ji Liu. Shifted chunk transformer for spatio-temporal representational learning. *arXiv preprint arXiv:2108.11575*, 2021. 11

[166] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the International Conference on Computer Vision*, 2019. 70

[167] Shifeng Zhang, Xiangyu Zhu, Zhen Lei, Hailin Shi, Xiaobo Wang, and Stan Z Li. S3fd: Single shot scale-invariant face detector. In *Proceedings of the International Conference on Computer Vision*, 2017. 123

[168] Shiwen Zhang, Sheng Guo, Weilin Huang, Matthew R Scott, and Limin Wang. V4d: 4d convolutional neural networks for video-level representation learning. *arXiv preprint arXiv:2002.07442*, 2020. 11

[169] Yubo Zhang, Pavel Tokmakov, Martial Hebert, and Cordelia Schmid. A structured model for action detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 14, 19, 29, 34, 35

[170] Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997. 124

[171] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the International Conference on Computer Vision*, 2017. 100, 105, 107, 108, 109, 110, 111, 112

[172] Yi Zhu, Xinyu Li, Chunhui Liu, Mohammadreza Zolfaghari, Yuanjun Xiong, Chongruo Wu, Zhi Zhang, Joseph Tighe, R Manmatha, and Mu Li. A comprehensive study of deep video action recognition. *arXiv preprint arXiv:2012.06567*, 2020. 9

[173] Mohammadreza Zolfaghari, Kamaljeet Singh, and Thomas Brox. Eco: Efficient convolutional network for online video understanding. In *Proceedings of the European Conference on Computer Vision*, 2018. 16