What, *When*, and *Where* Exactly? Human Activity Detection in Untrimmed Videos Using Deep Learning

by

Md Atiqur Rahman

A thesis submitted to the Faculty of Engineering of
the University of Ottawa
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science
School of Electrical Engineering and Computer Science
University of Ottawa

Abstract

Over the past decade, there has been an explosion in the volume of video data, including internet videos and surveillance camera footage. These videos often feature extended durations with unedited content, predominantly filled with background clutter, while the relevant activities of interest occupy only a small portion of the footage. Consequently, there is a compelling need for advanced processing techniques to automatically analyze this vast reservoir of video data, specifically with the goal of identifying the segments that contain the events of interest. Given that humans are the primary subjects in these videos, comprehending human activities plays a pivotal role in automated video analysis.

This thesis seeks to tackle the challenge of detecting human activities from untrimmed videos, aiming to classify and pinpoint these activities both in their spatial and temporal dimensions. To achieve this, we propose a modular approach. We begin by developing a temporal activity detection framework, and then progressively extend the framework to support activity detection in the spatio-temporal dimension.

To perform temporal activity detection, we introduce an end-to-end trainable deep

learning model leveraging 3D convolutions. Additionally, we propose a novel and adaptable fusion strategy to combine both the appearance and motion information extracted from a video, using RGB and optical flow frames. Importantly, we incorporate the learning of this fusion strategy into the activity detection framework.

Building upon the temporal activity detection framework, we extend it by incorporating a spatial localization module to enable activity detection both in space and time in a holistic end-to-end manner. To accomplish this, we leverage shared spatio-temporal feature maps to jointly optimize both spatial and temporal localization of activities, thus making the entire pipeline more effective and efficient.

Finally, we introduce several novel techniques for modeling actor motion, specifically designed for efficient activity recognition. This is achieved by harnessing 2D pose information extracted from video frames and then representing human motion through bone movement, bone orientation, and body joint positions.

Our experimental evaluations, conducted using benchmark datasets, showcase the effectiveness of the proposed temporal and spatio-temporal activity detection methods when compared to the current state-of-the-art methods. Moreover, the proposed motion representations excel in both performance and computational efficiency. Ultimately, this research shall pave the way forward towards imbuing computers with social visual intelligence, enabling them to comprehend human activities in any given time and space, opening up exciting possibilities for the future.

List of Publications

The research conducted in this thesis has led to the following publications.

1. **M. A. Rahman** and R. Laganière, "Spatio-temporal activity detection via joint optimization of spatial and temporal localization," accepted in *IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW), USA*, 2024.

2. **M. A. Rahman** and R. Laganière, "Mid-level fusion for end-to-end temporal activity detection in untrimmed videos," in *31st British Machine Vision Conference (BMVC2020), UK, Sept. 7–10*, 2020.

3. **M. A. Rahman** and R. Laganière, "Single-stage end-to-end temporal activity detection in untrimmed videos," in *17th Conference on Computer and Robot Vision (CRV2020), Ottawa, Canada, May 13–15*, pp. 206–213, IEEE, 2020.

4. **M. A. Rahman**, P. Kapoor, R. Laganière, D. Laroche, C. Zhu, X. Xu, and A. Osman Ors, "Deep people detection: A comparative study of SSD and LSTM-decoder," in *15th Conference on Computer and Robot Vision (CRV)*, pp. 305–312, IEEE, 2018.

The following publication was made during the PhD program, but does not form part of this thesis.

1. **M. A. Rahman**, T. Rahman, R. Laganière, N. Mohammed, and Y. Wang, "Membership inference attack against differentially private deep learning model," *Transactions on Data Privacy*, vol. 11, issue. 1, pp. 61–69, 2018.

Acknowledgements

All praises and thanks are due to the Almighty God, the most Beneficent, the most Merciful. This thesis would not have become a reality without His blessings and guidance.

First and foremost, I would like to express my profound gratitude to my supervisor, Prof. Robert Laganière. Throughout this arduous journey, Prof. Robert's relentless support, kindness, and guidance have been instrumental in shaping the direction of this PhD work. I am deeply thankful for his unwavering belief in my abilities, which has been a constant source of motivation and empowerment. Working under his guidance has been a true privilege, and I feel truly fortunate.

I owe a debt of gratitude to the members of the VIVA lab, who have been an integral part of my academic journey. I extend a special thanks to Rytis Verbickas, not only for his tireless efforts in maintaining the lab's complex computing resources but also for the numerous conversations we had that extended beyond the confines of our academic pursuits.

I would like to take this opportunity to acknowledge the generous financial support provided by the University of Ottawa. I also extend my sincere appreciation to Wrnch for

their valuable collaboration in the research presented in Chapter 5 of this thesis.

Last but certainly not least, I extend my deepest gratitude to my parents, and my wife. Their unwavering love, support, and understanding have been the foundation upon which I have built my academic pursuits. Their sacrifices, encouragement, and belief in me have been a constant source of strength and motivation. I am indebted to them for their enduring faith in my abilities and their unyielding support throughout this challenging journey.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The widespread availability of affordable video recording devices such as smartphones and surveillance cameras, along with the explosive growth of social media platforms such as YouTube, TikTok, Facebook, X (formerly Twitter), and Instagram, has led to an enormous daily accumulation of video data. For example, YouTube alone sees over 30,000 hours of new video contents uploaded every hour [1]. According to a report by Cisco, video traffic was projected to make up 82% of global internet traffic by the end of 2022, up from 75% in 2017 [2].

As this trend continues, the automatic analysis of video contents becomes increasingly important for efficient and timely processing of these yottabyte amount of video data. No surprise, the most prevalent and intriguing subjects in such realistic videos are humans. Consequently, gaining a comprehensive understanding of what humans are doing in a video

plays a pivotal role in automated video analysis.

Automated understanding of human activities in videos has received considerable research attention mainly in the setting of activity recognition [3–9], which typically involves classifying activities in short, trimmed video clips that are pre-segmented to precisely surround the activity. However, this is an artificial task, since real-world videos (e.g., web videos, surveillance videos, videos captured with smartphone cameras) are not trimmed, rather long and temporally untrimmed with the activities of interest buried under temporally cluttered background. In such untrimmed and unconstrained videos, it is common to find multiple instances of the same activity or different activities occurring at various points throughout the video. In some extreme cases, these videos might not even contain the specific activities of interest. Therefore, in this more realistic scenario, the concept of human activity understanding naturally expands beyond mere activity classification to encompass activity detection.

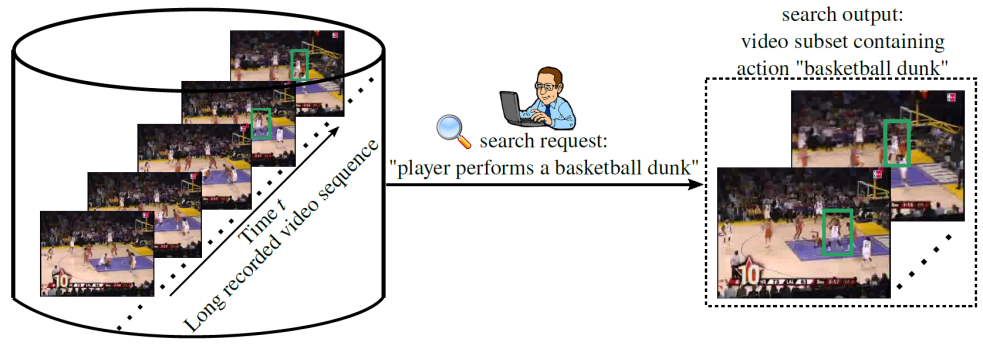
Activity detection entails identifying and pinpointing human activities within a video, essentially giving rise to two related tasks: temporal activity detection (TAD), and spatio-temporal activity detection (STAD). In the context of TAD, given a temporally untrimmed long video sequence, the objective is to localize the activities in time (i.e., determine the start and end time of each activity), and classify them (i.e., determine what activities are happening). In contrast, STAD goes one step further, and requires determining the spatial extents of the actors in the individual video frames, in addition to the temporal bounds

and the type of the activities happening in an untrimmed video. Consequently, STAD addresses three aspects of the human activity detection problem in videos – ***what*** activities are happening, ***when*** are they happening (i.e., temporal extent of the activities), and ***where*** exactly (i.e., spatial extent of the actors).

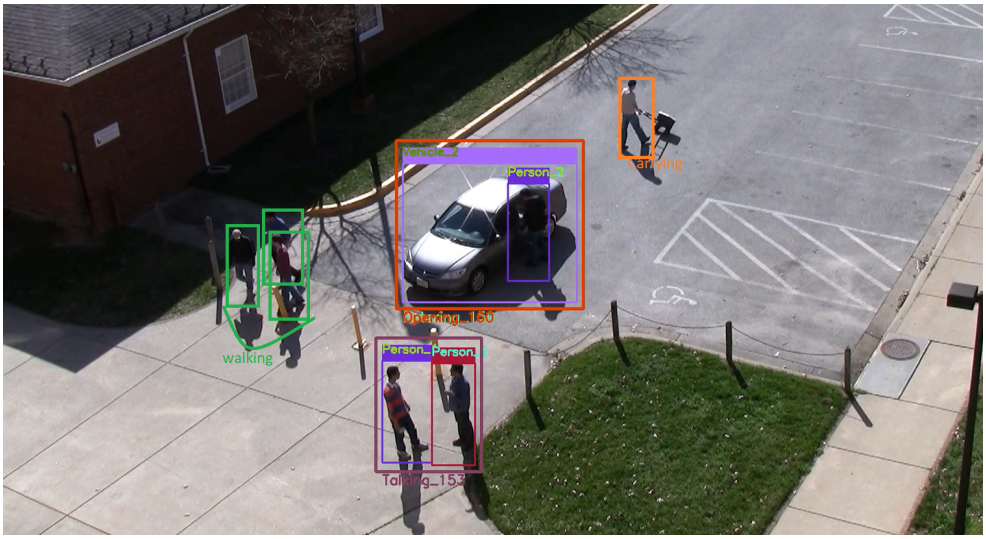
This thesis focuses on the human activity detection problem under the settings of TAD and STAD.

1.1 Motivation

Human activity detection in untrimmed and unconstrained videos has the potential to drive a number of real-world applications. Take the example of video surveillance, where manually sifting through hundreds of hours of recorded security footage in order to find a suspicious activity is prohibitively time-consuming, labour intensive and error-prone. Instead, the ability to automatically locate the activities of interest in these security footage would enable the processing of huge amount video data in a short time without any human intervention. Another application could be semantic video search and retrieval, where given a query activity, the system would automatically return only those video segments that contain the requested activity from an online repository of huge amount of video data. Apart from these applications that are suitable for an off-line setting (i.e., the entire video is available to produce the results), human activity understanding has even more far-reaching implications in an online setting. For instance, consider an autonomous



(a)



(b)

Figure 1.1: Some motivating applications of human activity detection in untrimmed videos. (a) In semantic video search, temporal activity detection (TAD) can automatically analyze huge amount of untrimmed videos and return only the time segments where the searched activity is happening. Image taken from [10]. (b) In automated video surveillance, a smart system can automatically analyze the long security footage and return the time segments along with the spatial detection of the different activities under the setting of spatio-temporal activity detection (STAD). Image source [11].

driving scenario where a self-driving vehicle must not only recognize the activities of the pedestrians in its vicinity (e.g., whether they are standing still, walking, running, or crossing the road) but also predict their future actions. This capability is essential for the vehicle to be able to interact with the pedestrians, and to make optimal decisions regarding its speed and course [12].

With the recent surge of deep learning based machine learning techniques, impressive success has been achieved in a variety of image understanding tasks, including image recognition [13–16], object detection [17–24], and image semantic segmentation [25–28]. However, progress in the realm of video analysis, especially concerning human activity detection in unconstrained videos, has not necessarily paralleled. In addition to the usual challenges encountered in image and video analysis, such as variations in lighting, perspectives, size, and appearance, as well as issues like partial obstruction and background noise, human activity understanding confronts unique difficulties arising from similarities both within and across activity classes. For instance, a single activity can be carried out by different individuals, sometimes even by the same person, using varying patterns and frequencies. Conversely, different activities can share similar execution patterns. Consequently, human activity detection in untrimmed videos remains highly challenging, leaving substantial opportunities for improvement.

In the context of activity detection under the TAD and STAD settings, there is a research gap in the literature as to how best to exploit multi-stream feature representations

of a video using deep learning. Moreover, end-to-end learning of task-specific spatio-temporal features has not received much attention in the literature. Specific to activity detection in the STAD setting, existing methods mostly rely on separate and disjoint pipelines for spatial and temporal localization of activities. This discrete processing pipeline incurs redundant computation. Additionally, it hinders the ability for these methods to apply direct temporal regression on the activity bounds, resulting in poor temporal localization accruacy [29]. Additionally, existing methods mostly resort to optical flow as a primary means for capturing actor motion information. However, computing dense and accurate optical flow is computationally very expensive, thus posing a bottleneck for real-time performance.

In this thesis, we therefore, aim to address the above limitations in activity detection from untrimmed videos and propose efficient solutions to bridge the gap in the effectiveness of the current state-of-the-art.

1.2 Research Questions

In light of the discussions presented above, the particular research questions that we address in this thesis are as follows:

1) How to learn task-specific spatio-temporal features from a video for efficient detection of activities in untrimmed videos?

2) How to best exploit multi-stream feature representations of a video in the context of activity detection?

3) How to perform joint optimization of spatial and temporal localization of activities in a single framework to realize activity detection in the STAD setting efficiently?

4) Can we devise computationally efficient methods that can effectively capture actor motion for the purpose of activity recognition?

1.3 Thesis Objectives

The main objectives of this thesis are to propose efficient solutions for human activity detection in untrimmed videos, both for the TAD and STAD settings, thus addressing the ***what***, ***when***, and ***where*** aspects of the problem. We are particularly inspired by the limitations of the existing methods as discussed in Sec. 1.1, and aim to address those concerns and limitations in this thesis. To this end, the specific objectives of this thesis include the following:

- Develop method for activity detection in the TAD setting that can learn task-specific features from the video while also being capable of efficiently fusing together multi-stream feature representations of a video, all in an end-to-end fashion.

- Devise solution for simultaneously optimizing spatial and temporal localization of activities in order to efficiently perform activity detection in the STAD setting.

- Propose computationally efficient techniques to model actor motion for video activity recognition.

1.4 Contributions

Guided by the research questions listed in Sec. 1.2, this thesis presents the following contributions.

End-to-End TAD Framework: We introduce a single-stage, end-to-end trainable deep learning framework that leverages a two-stream 3D Convolutional Neural Network (CNN) to learn task-specific appearance and motion features from RGB and optical flow frames of the input video. Our proposed TAD framework employs a multi-scale temporal feature hierarchy on top of the appearance and motion feature streams, enabling localization and classification of activities at varying scales in an end-to-end manner.

Mid-level Fusion of Multi-stream Feature Representations: We systematically investigate how and where to fuse the appearance and motion information of a video in order to gain the best of both worlds in the context of activity detection in the TAD setting. This exploration leads us to propose a novel and adaptable fusion strategy to combine the two streams of information in the middle of the processing pipeline, allowing the two streams of information to interact with each other. Importantly, we incorporate the learning of this fusion strategy into the proposed TAD framework in order to capture

the full correspondence between the two modalities, resulting in promising results on the TAD task.

STAD Framework for Joint Spatial and Temporal Localization of Activities: Building upon the proposed TAD framework, we extend it by incorporating a spatial localization module to enable activity detection both in space and time in a holistic end-to-end manner. To accomplish this, we leverage shared spatio-temporal feature maps to jointly optimize both spatial and temporal localization of activities, thus making the entire pipeline more effective and efficient. As an extension of the TAD framework, the proposed STAD framework enjoys direct regression on the temporal bounds of the activities which helps improve the temporal localization accuracy.

Novel Motion Representations for Actors: We introduce novel techniques for modeling actor motion in videos that are computationally efficient. This is achieved by harnessing 2D pose information extracted from video frames and then representing human motion through bone movement, bone orientation, and body joint positions. Our best performing method excels in both computational efficiency and effectiveness for video activity recognition.

1.5 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 reviews recent literature related to activity detection in both the TAD and STAD settings. It also discusses methods proposed for human pose-based activity recognition. Chapter 3 delves into the details of our proposed end-to-end trainable framework for activity detection in the TAD setting. Additionally, we explain our novel mid-level fusion method for combining the appearance and motion information of a video based on deep feature representations in the TAD context. Chapter 4 explores the extension of activity detection from the TAD setting to the STAD setting. We present a joint framework designed to localize activities in both space and time. Chapter 5 investigates the feasibility and effectiveness of incorporating human pose information for modeling actor motion as a computationally efficient alternative to using optical flow for video activity recognition. Finally, Chapter 6 concludes this thesis with a summary of the key points discussed in the thesis, followed by an exploration of its limitations. Additionally, it highlights potential future research directions and application areas for the methods proposed in this thesis.

Chapter 2

Related Work

In this chapter, we review the recent literature that is relevant to the work proposed in this thesis. To facilitate the literature review, we group the related works based on the objectives of this thesis as mentioned in Sec. 1.3. Consequently, we provide an overview of the literature on the following domains: activity detection in the TAD setting, multi-stream feature fusion for video activity recognition, activity detection in the STAD setting, and human pose for activity recognition.

2.1 Temporal Activity Detection (TAD)

Temporal activity detection (TAD) aims at classifying and temporally localizing the activities in an untrimmed video. Due to its close resemblance to the object detection task, which aims to classify and spatially localize objects in a 2D image, recent approaches to TAD,

in many ways, have paralleled the advances in object detection, hence can be classified as two-stage approach or single-stage approach. The recent related works on both approaches are discussed below.

2.1.1 Two-stage Approach for Temporal Activity Detection

Two-stage approaches for TAD work in two separate stages, where the first stage generates a sparse set of class agnostic temporal segments as activity proposals, and the second stage classifies the temporal segments into specific activity classes while further refining the temporal bounds of the proposals. Earlier methods (e.g., [30–33]) could not offer end-to-end training due to the proposals being generated using an external proposal generation mechanism leading to poor computational efficiency. Moreover, the proposals were typically selected using a fixed sliding window, thereby resulting in poor localization accuracy.

Inspired by the impressive success of the state-of-the-art (SOTA) two-stage object detectors, such as Fast R-CNN [34] and Faster-RCNN [17], several attempts have been made [35–39] to directly apply these methods to the TAD problem. Among these, R-C3D [39] closely followed the original Faster-RCNN architecture and achieved superior performance by applying C3D [40], a 3D CNN, for learning task-specific spatio-temporal features of a video in an end-to-end manner. CMS-RC3D [41] later improved upon R-C3D by applying a temporal feature pyramid to represent activities at different temporal scales.

However, the performance of R-C3D and its variants is limited by the short temporal footprint of the feature extractor 3D-CNN (i.e., C3D [40]). Moreover, naively applying Faster-RCNN to TAD causes misalignment between the temporal spans of the predefined anchors and the receptive fields of the feature maps. These issues were later addressed in TAL-net [42] which is based on a pretrained two-stream 3D CNN called I3D that enjoys comparatively longer temporal footprint. TAL-net leverages I3D to extract spatio-temporal features and emplys a multi-tower temporal convolutional network to perform receptive field alignment of the anchor segments. These improvements earned TAL-net state-of-the-art results on the THUMOS'14 [43] benchmark. However, TAL-net could not offer end-to-end learning of the task-specific spatio-temporal features as it is based on pre-trained off-the-shelf features. Moreover, it uses simple *late fuison* approach for fusing the two-streams which is not efficient in the settings of TAD as discussed in Chapter 3.

2.1.2 Single-stage Approach for Temporal Activity Detection

Although highly performant, the two-stage methods for TAD are comparatively slower as they need to make two separate passes over the input. Single-stage approaches, on the other hand, can manage to generate activity proposals as well as classify the proposals in a single step. Along this line, a number of works [44–49] leveraged Recurrent Neural Networks (RNNs) to encode a sequence of frames or video snippet features extracted using 3D CNN (e.g., [40, 50]) followed by frame-level predictions. However, such sequential prediction is

often time-consuming for long untrimmed videos.

Motivated by the impressive speed and accuracy of the leading single-stage object detectors, such as SSD [18] and YOLO [51, 52], recent single-stage approaches for TAD (e.g., SSAD [53], GTAN [54]) employ a multi-scale temporal 1D feature pyramid based on a set of anchors of predefined or learnable scales to directly predict activity classes and regress temporal bounds at varying temporal scales. Decouple-SSAD [55] improves upon SSAD by having two separate branches, one for classification and the other for localization, and achieves state-of-the-art performance on THUMOS'14 benchmark [43]. However, none of these approaches could afford end-to-end training as they are based on feature extraction and aggregation from multiple different 2D, or 3D CNNs. Our proposed approach for TAD, in contrast, performs activity detection based on task-specific spatio-temporal features that are learned via a single-stage 3D CNN end-to-end, resulting in impressive results.

2.2 Multi-stream Feature Fusion for Activity Recognition

In order to provide a robust video representation with a view to improving the activity recognition performance, the pioneering work called two-stream CNN [3] first proposed the idea of having two separate processing streams – an spatial stream to capture the appearance information of the video from the RGB frames, and a temporal stream to

capture the motion information from optical flow frames. These two streams are finally combined together very late in the processing pipeline using element-wise averaging. This approach is commonly known as *late fusion*. Most state-of-the-art methods for video activity recognition (e.g., [4, 6–8, 56]) and temporal activity detection (e.g., [42, 55]) mainly follow this route and perform *late fusion* of the two streams. However, there are only a handful of works that demonstrated optimal results using *mid-level fusion* approach that combines the two streams at an appropriate level in the middle of the processing pipeline rather than very late (hence the name *mid-level fusion*). For example, Feichtenhofer *et al.* proposed *mid-level fusion* via concatenation of the two streams followed by convolution [57], or element-wise multiplication followed by residual connections [58]. EPIC-Fusion [59], on the other hand, combined audio, appearance, and motion information via concatenation followed by fully-connected layer for the purpose of ego-centric action recognition. However, none of these works explored any sophisticated fusion strategies. We therefore, explore more sophisticated *mid-level fusion* strategies, while investigating where in the feature abstractions level such fusion would be appropriate in the context of TAD.

2.3 Spatio-Temporal Activity Detection

Unlike temporal activity detection (TAD), spatio-temporal activity detection (STAD) requires localizing activities both in spatial and temporal dimensions, and as such, is significantly more challenging than TAD. Some of the early approaches (e.g., [60, 61])

attempted to extend the unsupervised 2D region proposal generation algorithms, (e.g., Selective Search [62], Prime Object Proposal [63]) to their 3D counterparts with a view to generating supervoxels which are then merged together based on color, texture, or motion information to form activity tubes. These activity tubes are then encoded with dense trajectories [64] followed by classification. Soomro *et al.* [65], on the other hand, used video segmentation to generate supervoxels which are then encoded using bag-of-visual-words on improved dense trajectory features [66] followed by classification using SVM. However, such supervoxel based approaches suffer from very poor temporal localization, as the supervoxels may span over very long intervals.

Recent approaches address the STAD problem mainly in two separate stages – first, perform actor detection in individual video frames, then localize the activities in time based on the frame-based detections. Along this line, a number of methods [67–72] used complex dynamic optimization to link the frame-level detections. Weinzaepfl *et al.* [73] applied temporal sliding stage over the frame-level detections to realize temporal localization. However, due to frame-based detections, these methods fail to fully capture the temporal structure of the activities. To overcome this issue, later methods [74–77] introduced clip-level detection on short video snippets. They aim to regress activity tubelets within these clips and subsequently link them to achieve temporal localization. However, these methods have limitations, especially when dealing with complex and prolonged activities. Their effectiveness is hindered by the need for optimizing tubelet linking and their inability

to incorporate long-term temporal contexts. In contrast, our proposed method to address the STAD problem (Chap. 4) avoids the need for tubelet linking altogether by directly regressing activity over time, thus resulting in impressive performance.

Several recent methods have demonstrated superior ability in modeling longer temporal context, thus achieving better performance on the STAD task. Notable ones include TACNet [78], which proposed a temporal context detector to extract long-term contextual information and a transition-aware classifier to further distinguish the ambiguous states from real activity sequences. The Spatio-TEmporal Progressive (STEP) [79] action detector, on the other hand, progressively processes longer sequences and adaptively extends proposals to follow the action movement.

There have been some recent works that have attempted to explicitly capture the relationships between actor, surrounding context and other objects to solve the STAD problem. To this end, Context-Aware RCNN [80] used the background information to model interactions between the persons in the frame and the context. Pan *et al.* [81] proposed Actor-Context-Actor Relation Network (ACAR-Net), incorporating a novel high-order relation reasoning operator and an actor-context feature bank. This enables the reasoning of indirect relations for spatio-temporal action localization. Ni *et al.* [82] emphasized on the identity details of actors by a Hierarchical Graph Neural Network (HGNN) to effectively model long-term relationships within the same identity and across different identities.

However, none of the above methods perform spatial and temporal regression in a single network, rather by two separate processing pipelines, thereby incurring redundant computations. Moreover, the disjoint optimization of the two tasks possibly leads to sub-optimal results. A very recent work called STAR [29] addressed this issue and proposed an end-to-end two-stage pipeline for joint optimization of spatial and temporal detection in a single network. Moreover, unlike the previous methods, STAR is capable of providing spatial detection that enclose both the actor(s) and the object(s) that the actor(s) interact(s) with during an activity.

Our proposed approach to address the STAD problem, as detailed in Sec. 4.3, is closely aligned with STAR [29] in terms of performing simultaneous spatial and temporal regression of activities. However, unlike STAR, we propose a single-stage end-to-end pipeline that employs multi-scale architecture for both spatial and temporal localization to be able to handle activities at varying scales. Moreover, STAR extracts tube features from the whole scene rather than the tube-specific scene contexts during the temporal detection stage. This is likely to lead poor temporal localization for unrelated activities, mainly due to the confluence of the features from the whole scene. Furthermore, the simple heuristic used by STAR to form activity tubes is susceptible to failure in complex multi-actor scenarios. In our proposed approach, we aim to address these limitations.

2.4 Human Pose-based Activity Recognition

Human pose refers to the specific configuration of the body parts. Due to the discriminative nature of body pose, literature on activity recognition (e.g., [83–86]) has been actively tapping into the human pose information for representing activity dynamics. Though previous methods used to rely on hand-crafted features (e.g., relative joint locations, or covariance matrix of joint locations [87]), recent approaches have mostly leveraged features learned via deep networks mainly due to their superior performance and generalization ability.

Since activity recognition requires modeling temporal contexts, RNNs have been the primary choice for human pose based activity recognition. Along this line, Du *et al.* [88] proposed a bi-directional LSTM [89] model for skeleton-based activity recognition by splitting the skeleton into anatomically-relevant parts (e.g., legs, arms, torso etc.). Each layer of the bi-directional LSTM learns to generate specialized features on a specific body part. Along the same line, Shahroudy *et al.* proposed partware LSTMs [90] to separate the LSTM memory cells into part-based sub-cells while allowing the network to learn long-range representation for individual body parts. Originally introduced for images, multi-dimensional LSTM [91], comprising of the time dimension and the topological traversal of the joints, has been applied on pose sequences in [92].

Though LSTMs have been proved successful for temporal modeling, it has the

disadvantage of high-computational demands which is further aggravated by its inability to parallelize computations as it is inherently sequential in nature. Therefore, there have been approaches based on CNNs proposed to handle pose sequences (e.g., [93–95]). Such approaches require a 3D tensor as input, and therefore encode the pose sequences as a trajectory [94], or into RGB-like images [95].

However, recent methods adopt a multi-stream architecture combining human pose information with appearance information extracted from RGB frames. This is due to the fact that pose information alone is not able to capture the important contextual cues around the actors/body parts that often play a crucial role in distinguishing one activity from the other (e.g., brushing hair vs. brushing teeth). To this end, Song *et al.* [96] proposed an end-to-end spatial and temporal attention model for activity recognition. The spatial model processes RGB frames to capture appearance information while the temporal model is based on human skeletons. Zolfaghari *et al.* [97], on the other hand, proposed a three-stream model architecture to integrate pose information from human 3D skeletons, motion information from pre-computed optical flows, and appearance information from raw RGB frames. The authors used a Markov chain model to successively integrate the different cues. Wang *et al.* [98] attempted to directly model the human-object interactions involved in an activity by proposing a two-stream relational network (PSRN). PSRN models the temporal dynamics using 2D human pose sequences directly extracted from raw videos while activity-related objects are modeled from randomly sampled video frames.

Another two-stream model is proposed in [99] where the RGB stream employs a soft-attention mechanism to guide the learning of spatio-temporal features from RGB frames conditioned on pose features. The pose stream learns the pose features by a convolutional model.

While the above discussed methods have mainly utilized 2D pose information as input to subsequent processing pipelines, our proposed approach, as discussed in Chap. 5, focuses on building more robust representation of human motion and structure based on raw 2D pose data.

Chapter 3

End-to-End Temporal Activity Detection using Mid-level Fusion

3.1 Introduction

In this chapter, we present an end-to-end deep learning framework to address the temporal activity detection (TAD) problem in untrimmed videos. Additionally, we propose a novel fusion strategy for combining multi-stream feature representations of a video with a view to improving the temporal localization accuracy of the proposed TAD framework.

As reviewed in Chapter 2, the existing state-of-the-art (SOTA) methods for TAD (e.g., [42, 49, 53–55]) are based on either two-stage or single-stage pipeline. However, most of them do not learn task-specific spatio-temporal feature representations of a video end-to-

end, but rather extract deep features from short video snippets using pre-trained 2D CNNs (e.g., [3,4]), or 3D CNNs (e.g., [6,40]), followed by complex feature aggregation to provide the temporal modeling. Such off-the-shelf feature representations are not optimal for temporal localization of activities in diverse video domains, especially in the context of TAD involving unconstrained and untrimmed videos. The handful of works ([39,41,100]) that attempted to provide an end-to-end framework for TAD, are all based on the two-stage pipeline requiring multiple passes over the input, and are therefore, computationally more expensive than their single-stage counterparts. Besides, all of these methods rely on relatively shallow 3D CNNs (e.g., [40]) having limited temporal footprint to extract spatio-temporal features which essentially leads to their poor performance [6].

Moreover, the most effective techniques for Temporal Action Detection (TAD) primarily employ a two-stream architecture (e.g., [3]) that consists of two parallel processing streams: a spatial stream for capturing appearance information and a motion stream for modeling motion information. To integrate the information from both streams, these methods predominantly employ the *late fusion* approach which simply averages the *logits* (i.e., the raw output of the penultimate layer in a deep network) from the two processing streams at the end of the processing pipeline, hence referred to as *late fusion*. However, limited body of research (e.g., [57–59]) has demonstrated superior results in video activity recognition by using *mid-level fusion* that combines the appearance and motion streams in the middle of the processing pipeline, hence the name *mid-level fusion*.

Therefore, further research is warranted into investigating the efficacy of the same in the context of temporal activity detection.

Furthermore, *mid-level fusion* has been studied mainly based on simple fusion strategies (e.g., *sum*, *max*, or *convolution*) which may not be optimal for modeling long-range temporal relationship in the context of TAD. Therefore, it would be interesting to explore different *mid-level fusion* strategies so as to best exploit the multi-stream feature representations of a video in the context of TAD.

Motivated by the above limitations, in this chapter, we propose a single-stage and end-to-end trainable deep learning framework for activity detection in the TAD setting, which we call the TAD framework. Our proposed framework leverages multi-stream cues of a video based on a novel sophisticated *mid-level fusion* method. Drawing inspiration from the single-stage object detection methods (e.g., [18, 51]), we first build a multi-scale temporal feature pyramid atop a two-stream 3D CNN to learn task-specific appearance and motion features of a video in an end-to-end manner. Capitalizing the two-stream deep feature representations, we then systematically investigate the effectiveness of several sophisticated *mid-level fusion* methods at different levels in the feature abstractions with a view to finding the optimal strategy to fuse such multi-stream information. This investigation eventually leads us to propose a new *mid-level fusion* approach based on efficient bilinear pooling operation [101]. Finally, a set of 1D temporal convolutional filters are learned on top of the fused features to directly regress on the temporal boundaries and generate class predictions for the different

activities present in the input video.

Contributions: Our contributions in this chapter are three fold:

1. We propose a single-stage End-to-End TAD framework.

2. We investigate effective ways to fuse multi-stream feature representations of a video in the context of TAD.

3. We propose a novel *mid-level fusion* strategy to improve the accuracy of activity detection in the TAD setting.

4. Finally, we set new state-of-the-art results on THUMOS'14 and MEXaction2 benchmarks for temporal activity detection.

Related Publications: The methods and results presented in this chapter have already appeared in the following publications.

- **M. A. Rahman** and R. Laganière, "Mid-level fusion for end-to-end temporal activity detection in untrimmed videos," in *31st British Machine Vision Conference (BMVC2020), UK, Sept. 7–10*, 2020.

- **M. A. Rahman** and R. Laganière, "Single-stage end-to-end temporal activity detection in untrimmed videos," in *17th Conference on Computer and Robot Vision (CRV2020), Ottawa, Canada, May 13–15*, pp. 206–213, IEEE, 2020.

3.2 Problem Statement

We formulate the problem definition for TAD as follows. Given a temporally untrimmed long video sequence $\mathcal{I} = \{i_1, i_2, \ldots, i_T\}$ containig T frames, the goal of the TAD task is to output a set $\Psi = \{\psi_1, \psi_2, \ldots, \psi_N\}$ of predicted temporal activity segments such that each activity segment $\psi_n = (\phi^n_{start}, \phi^n_{end}, c^n, p^n)$ has associated with it the activity start and end times ϕ^n_{start} and ϕ^n_{end} respectively, the predicted activity category $c^n \in \{1, \ldots, C\}$, as well as the prediction confidence score p^n, where C denotes the total number of activity categories. Under this problem formulation, we assume that the annotations Ψ are available during the training time, whereas, we need to predict the same during inference.

3.3 Proposed Approach

Our primary objective is to design a single-stage and end-to-end trainable temporal activity detection framework. To this end, we build on a two-stream architecture [3] – a spatial stream to model the appearance information from RGB frames, and a temporal stream to capture the motion contexts from pre-computed optical flow frames. This leads to the second objective of this work, which is to find the optimal strategy to fuse such multi-stream information extracted from a video in the context of TAD. We first explain our baseline TAD framework that is based on the *late fusion* approach in Sec. 3.3.1. We then explore several sophisticated *mid-level fusion* methods in Sec. 3.3.2. Finally, an exploration of the different

feature abstraction levels where such *mid-level fusion* could be applied is presented in Sec. 3.3.3.

3.3.1 Baseline End-to-End TAD Framework

We are inspired by the working principle of the single-stage object detectors (e.g., [18, 51]) that turns an image classification network into an object detection network. In a similar vein, we build upon a state-of-the-art activity recognition 3D CNN and transform it into an end-to-end TAD framework. Figure 3.1 shows the overall architecture of the baseline TAD framework that comprises two separate branches: one for the appearance stream, the other for the motion stream. Each branch has the same architecture and is composed of three key components – a 3D CNN feature extractor, a multi-scale temporal feature hierarchy and a set of prediction layers to generate predictions on activity segments.

3.3.1.1 3D CNN Feature Extractor:

The input to our network is a pair of video sequences $(\mathcal{I}_{\text{rgb}}, \mathcal{I}_{\text{flow}}) \in \mathbb{R}^{T\times H\times W\times C}$ consisting of T number of RGB and FLOW frames, where H, W, and C represent the the height, width and channel dimensions of each frame, respectively. $(\mathcal{I}_{\text{rgb}}, \mathcal{I}_{\text{flow}})$ are first passed through a two-stream 3D CNN to learn task-specific spatio-temporal feature representations of the video sequence end-to-end. The architecture of the 3D CNN is adopted from the state-of-the-art video activity recognition network called S-3DG [8]. We extract two-stream 3D

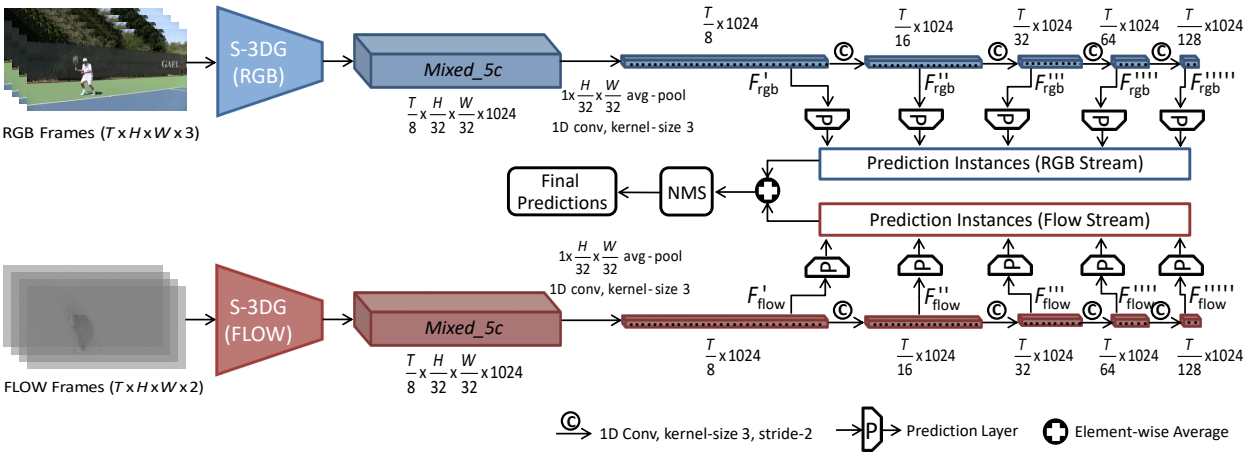


Figure 3.1: Architecture of the baseline end-to-end TAD framework. We perform activity detection on a pair of video sequences of RGB and FLOW frames each of length T. Both clips are first passed through a two-stream feature extractor 3D CNN, followed by a two-stream temporal feature hierarchy to generate multi-scale and multi-stream temporal feature maps. A set of 1D temporal convolutional filters are learned to generate predictions on activity segments. The baseline framework fuses the predictions from both streams using *late fusion* approach via element-wise averaging. Finally, the fused predictions are post-processed using Non-Maximum Suppression (NMS).

feature maps $(\mathcal{F}_{\text{rgb}}, \mathcal{F}_{\text{flow}}) \in \mathbb{R}^{\frac{T}{8}\times\frac{H}{32}\times\frac{W}{32}\times 1024}$ from the *Mixed_5c* block of S-3DG, and pass them through *average pooling* operation to completely collapse the spatial dimensions of the feature maps. Finally, we apply 1D temporal convolutions (kernel size 3, stride 1, padding 1) to further extend the temporal receptive field and produce temporal-only two-stream feature maps $(\mathcal{F}'_{\text{rgb}}, \mathcal{F}'_{\text{flow}}) \in \mathbb{R}^{\frac{T}{8}\times 1024}$. These temporal-only two-stream feature maps serve as the base feature maps to perform activity detection. The feature extractor 3D CNN being *fully-convolutional*, the length of the video sequences T can be arbitrarily long and is only constrained by the amount of physical memory.

3.3.1.2 Multi-scale Temporal Feature Hierarchy

Following the work [53], we build a multi-scale temporal feature hierarchy on top of the 3D CNN feature extractor. To this end, we create a cascade of four 1D temporal convolutional layers for each stream, where each layer consists of 1024 filters each having a kernel size of 3 and temporal strides of 2. This sequential arrangement thus generates a two-stream temporal feature hierarchy that progressively decreases the temporal resolution while increasing the temporal receptive field of the feature maps, thereby allowing activity contexts to be captured at varying temporal scales. To be specific, starting from the base feature maps $(\mathcal{F}^{'}_{\text{rgb}}, \mathcal{F}^{'}_{\text{flow}}) \in \mathbb{R}^{\frac{T}{8}\times 1024}$, the two-stream feature hierarchy produces four temporal-only feature maps, namely, $(\mathcal{F}^{''}_{\text{rgb}}, \mathcal{F}^{''}_{\text{flow}}) \in \mathbb{R}^{\frac{T}{16}\times 1024}$; $\cdots\cdots$; $(\mathcal{F}^{'''''}_{\text{rgb}}, \mathcal{F}^{'''''}_{\text{flow}}) \in \mathbb{R}^{\frac{T}{128}\times 1024}$. Together, the base features maps (i.e., $(\mathcal{F}^{'}_{\text{rgb}}, \mathcal{F}^{'}_{\text{flow}})$) and these four feature maps create a two-stream multi-scale feature hierarchy $(\mathcal{F}^{MS}_{\text{rgb}}, \mathcal{F}^{MS}_{\text{flow}})$; such that, $\mathcal{F}^{MS}_{\text{rgb}} = \{\mathcal{F}^{'}_{\text{rgb}}, \cdots\cdots, \mathcal{F}^{'''''}_{\text{rgb}}\}$ and $\mathcal{F}^{MS}_{\text{flow}} = \{\mathcal{F}^{'}_{\text{flow}}, \cdots\cdots, \mathcal{F}^{'''''}_{\text{flow}}\}$. This multi-scale feature hierarchy is used as input to the prediction layers as detailed in the next section.

3.3.1.3 Prediction Layers

In the baseline TAD framework, we make predictions on each feature map belonging to the two-stream multi-scale feature hierarchy $(\mathcal{F}^{MS}_{\text{rgb}}, \mathcal{F}^{MS}_{\text{flow}})$. Each prediction layer consists of a set of 1D convolutional filters each with kernel size 3 and stride 1. Following [53], we

associate a set of *default temporal segments* with each temporal location in a feature map. Each *default temporal segment* at a temporal location has the same default center location, but different scale ratios $s_k \in \{s_1, s_2, \cdots, s_K\}$. Therefore, K activity segments are predicted relative to the *default temporal segments* at each temporal location. For example, each temporal location $i^{'} \in [0, \frac{T}{8})$ in $\mathcal{F}^{'} \in \mathbb{R}^{\frac{T}{8} \times 1024}$ has a base temporal scale of $\frac{1}{T/8}$. Therefore, the *default temporal segment* with scale ratio s_k at $i^{'}$ has the default center and default width as $d_c = \frac{i^{'}+0.5}{T/8}$ and $d_w = s_k \cdot \frac{1}{T/8}$, respectively. Each prediction layer, therefore, employs a total of $K(C+3)$ filters at each temporal location to generate the following predictions:

(i) class scores $\{p_r\}_{r=1}^{C}$ over C activity classes including the *background* class;

(ii) the center and width offsets Δ_c and Δ_w relative to d_c and d_w, respectively; and

(iii) an overlap score p_{ov} indicating the overlap of the *default temporal segment* with the closest ground-truth segment, which is later passed through *sigmoid* function to produce a confidence value in the range $[0, 1]$.

Following [53], the center and width offsets are used to compute the actual center ϕ_c and actual width ϕ_w of the predicted activity segments, whereas, ϕ_c and ϕ_w are used to compute the activity start time ϕ_{start} and activity end time ϕ_{end} as shown in the following equation.

$$\phi_c = d_c + \alpha_1 d_w \Delta_c \;\text{ and }\; \phi_w = d_w \exp(\alpha_2 \Delta_w) \tag{3.1}$$

$$\phi_{start} = \phi_c - \frac{\phi_w}{2} \;\text{ and }\; \phi_{end} = \phi_c + \frac{\phi_w}{2} \tag{3.2}$$

where, α_1 and α_2 are hyper-parameters used to control the effect of Δ_c and Δ_w, respectively.

3.3.1.4 Fusion of the Two Streams

In the baseline TAD framework, we use *late fusion* approach to combine the individual predictions of the two streams via element-wise averaging. Final predictions are generated by passing the combined predictions through Non-Maximum Suppression (NMS) technique.

3.3.2 *How* to Fuse the Two Streams

As aptly reasoned in [102], to discriminate between activities having similar motion or appearance pattern (e.g., *'brushing teeth'* vs. *'brushing hair'*), it is necessary for the appearance and motion streams to interact earlier in the network, rather than fusing them late. We, therefore, propose *mid-level* fusion in this work. However, the existing works (e.g., [57–59]) use simple straightforward methods (e.g., *sum*, *max*, or *convolution*) for *mid-level* fusion that fail to capture the full correspondence between the different input modalities, resulting in poor performance as demonstrated in our experiments. To this end, we are inspired by the latest advancement in *Visual Question Answering (VQA)* that makes efficient use of bilinear pooling based methods (e.g., [103, 104]) to allow for high-level interactions between visual and linguistic modalities while keeping the dimensionality of the resultant feature computationally feasible. We first discuss the traditional *mid-level fusion* strategies, followed by the efficient bilinear pooling based methods. Afterwards, we

propose a novel *mid-level fusion* method. For the purpose of the following discussion, we assume that $\mathcal{F}_{\text{rgb}} \in \mathbb{R}^{T \times C}$, $\mathcal{F}_{\text{flow}} \in \mathbb{R}^{T \times C'}$ represent two input feature maps that need to be fused, where T denotes the temporal dimension, and C and C' represent the channel dimensions for the input feature maps, respectively.

3.3.2.1 Sum, Max, and Convolution

The *sum, max* and *convolution* fusion can be defined as follows:

$$\mathcal{F}_{\text{sum}} = \mathcal{F}_{\text{rgb}} \oplus \mathcal{F}_{\text{flow}} \tag{3.3}$$

$$\mathcal{F}_{\text{max}} = \max(\mathcal{F}_{\text{rgb}}, \mathcal{F}_{\text{flow}}) \tag{3.4}$$

$$\mathcal{F}_{\text{conv}} = (\mathcal{F}_{\text{rgb}} \parallel \mathcal{F}_{\text{flow}}) * f + b \tag{3.5}$$

where $\mathcal{F}_{\text{sum}} \in \mathbb{R}^{T \times C}$, $\mathcal{F}_{\text{max}} \in \mathbb{R}^{T \times C}$, and $\mathcal{F}_{\text{conv}} \in \mathbb{R}^{T \times C^{\text{conv}}}$ are the fused feature maps, $f \in \mathbb{R}^{1 \times (C+C') \times C^{\text{conv}}}$ is a filter bank of C^{conv} filters where each filter employs 1D convolutions with channel dimension of $(C + C')$, and $b \in \mathbb{R}^{C^{\text{conv}}}$ are the biases. Here, $\oplus$ represents element-wise addition, while $\parallel$ and $*$ represent concatenation and convolution across the channel dimension, respectively. We set $C^{\text{conv}} = C = C'$.

3.3.2.2 Multi-modal Low-rank Bilinear Pooling

Multi-modal Low-rank Bilinear Pooling (MLB) [103] operation is designed to reduce the computational complexity of the bilinear pooling operation by factoring the

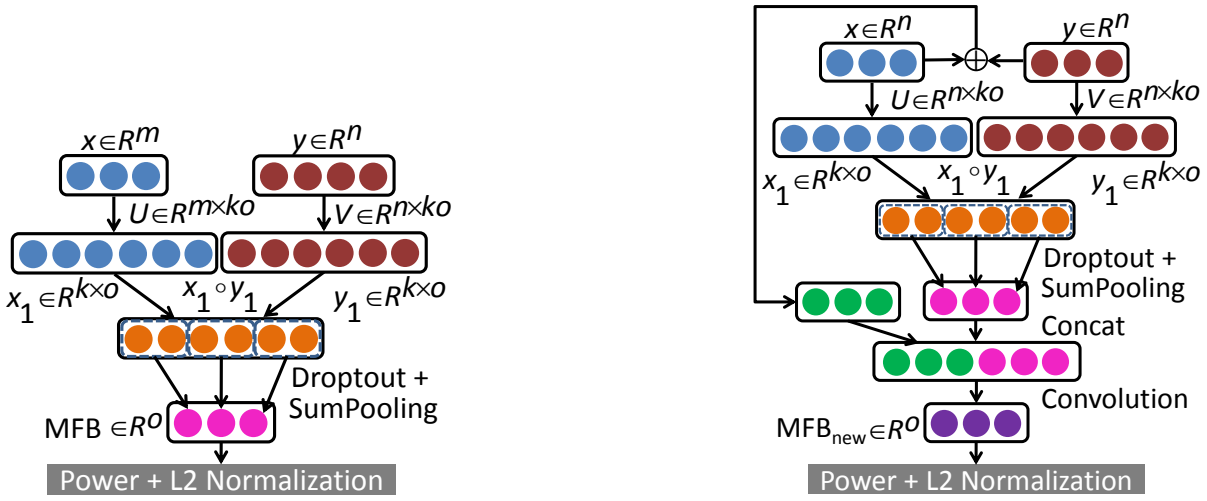


Figure 3.2: (a) *MFB* [104] fusion. (b) Proposed fusion method *MFB_new*.

three-dimensional weight tensor of the bilinear pooling operation into three two-dimensional weight tensors. To this end, MLB first non-linearly projects the input feature maps into a common embedding space where they are fused via element-wise multiplication, which is then followed by a linear projection as follows:

$$\mathcal{F}_{\text{MLB}} = (\sigma(\mathcal{F}_{\text{rgb}}U) \odot \sigma(\mathcal{F}_{\text{flow}}V))P \tag{3.6}$$

where, $\mathcal{F}_{\text{MLB}} \in \mathbb{R}^{T \times C^{\text{MLB}}}$. The three weight tensors are denoted by $U \in \mathbb{R}^{C \times D}$, $V \in \mathbb{R}^{C' \times D}$, and $P \in \mathbb{R}^{D \times C^{\text{MLB}}}$, and the dimensionality of the common embedding space is denoted by $D = min(C, C')$. Here, $\odot$ and σ represent element-wise multiplication and non-linear activation function, respectively. We set $C^{\text{MLB}} = C = C'$.

3.3.2.3 Multi-modal Factorized Bilinear Pooling

Multi-modal Factorized Bilinear Pooling (MFB) [104] provides an improvement over MLB. As shown in Figure 3.2(a), *MFB* first projects the input feature maps into a higher dimensional space, then fuses them via element-wise multiplication followed by *Dropout* and *SumPooling* operation as follows:

$$\mathcal{F}_{MFB} = SumPooling(Dropout(\mathcal{F}_{\text{rgb}}U \odot \mathcal{F}_{\text{flow}}V), J) \tag{3.7}$$

where, $\mathcal{F}_{MFB} \in \mathbb{R}^{T \times C^{MFB}}$; $U \in \mathbb{R}^{C \times D}, V \in \mathbb{R}^{C^{'} \times D}$ are the weight tensors, $D = J \times C^{MFB}$ is the dimensionality of the embedding space. Here, J is the window-size for *SumPooling* and $\odot$ represents element-wise multiplication. We set $C^{MFB} = C = C^{'}$. The output of the *MFB* fusion are passed through power normalization ($z \leftarrow sign(z)|\ z\ |^{0.5}$), followed by l_2-normalization ($z \leftarrow z/\|z\|$) to avoid unsatisfactory local minima during training, as explained in [104].

3.3.2.4 Proposed Fusion Method

We build on the *MFB* fusion. However, unlike *MFB*, we non-linearly project the input feature maps into the higher dimensional space. We, further, boost the fused feature map by having it convolved with the element-wise summation of the original feature maps as shown in Figure 3.2(b). This skip connection essentially acts as a residual connection as it allows

the original input feature maps to directly interact with the transformed fused feature map, thereby providing better representation capacity than *MFB*. The motivation for using such a residual connection stems from the fact that the fusion operator is used in multiple layers in a deep two-stream 3D CNN (i.e., S-3DG), and residual connections are known to help better optimize such deep networks [105]. We denote the proposed fusion approach as *MFB_new* which is formulated as follows:

$$z = \mathit{SumPooling}(\mathit{Dropout}(\sigma(\mathcal{F}_{\text{rgb}}U) \odot \sigma(\mathcal{F}_{\text{flow}}V)), J) \tag{3.8}$$

$$\mathcal{F}_{\mathit{MFB\_new}} = ((\mathcal{F}_{\text{rgb}} \oplus \mathcal{F}_{\text{flow}}) \parallel z) * f + b \tag{3.9}$$

where, $\mathcal{F}_{\mathit{MFB\_new}} \in \mathbb{R}^{T \times C^{\text{new}}}$; and U, V, and J represent the same as in *MFB*, whereas, $f \in \mathbb{R}^{1 \times (C+C^{\text{new}}) \times C^{\text{new}}}$ is a filter bank of C^{new} filters and $b \in \mathbb{R}^{C^{\text{new}}}$ are biases. For the proposed fusion, we set $C^{\text{new}} = C = C'$. Similar to *MFB*, we pass the output of *MFB_new* through the normalization steps as explained above.

3.3.3 *Where* to Fuse the Two Streams

In a two-stream architecture, ideally the two-stream feature maps can be fused at any point in the processing pipeline, provided that the spatial and/or temporal dimensions of the input feature maps match [57]. To this end, we consider *mid-level fusion* at different feature abstraction levels with a view to finding the optimal level to fuse such multi-stream

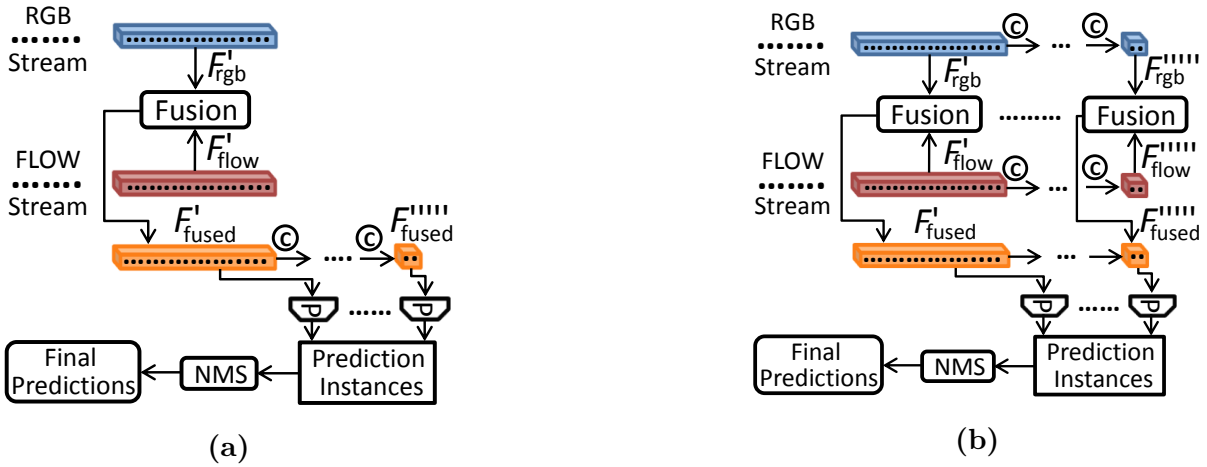


Figure 3.3: (a) Fusion at base feature maps ($\mathcal{F}^{'}_{\text{rgb}}, \mathcal{F}^{'}_{\text{flow}}$); (b) Fusion at the multi-scale feature hierarchies ($\mathcal{F}^{MS}_{\text{rgb}}, \mathcal{F}^{MS}_{\text{flow}}$). Refer to Fig. 3.1 for legends.

information in the context of TAD. In particular, we explore the base feature maps ($\mathcal{F}^{'}_{\text{rgb}}$, $\mathcal{F}^{'}_{\text{flow}}$), as well as the multi-scale feature hierarchies ($\mathcal{F}^{MS}_{\text{rgb}}$, $\mathcal{F}^{MS}_{\text{flow}}$) as the potential candidates for fusion.

3.3.3.1 Fusion at Base Feature Map

Figure 3.3(a) shows how to integrate the RGB and FLOW streams at the base feature map level by fusing $\mathcal{F}^{'}_{\text{rgb}}$ and $\mathcal{F}^{'}_{\text{flow}}$ to produce a fused base feature map $\mathcal{F}^{'}_{\text{fused}}$. This fused feature map is later leveraged to generate a multi-scale temporal feature hierarchy $\mathcal{F}^{MS}_{\text{fused}}$ following the same technique as detailed in Sec. 3.3.1.2. Predictions are made on each feature map in $\mathcal{F}^{MS}_{\text{fused}}$.

3.3.3.2 Fusion at Multi-scale Feature Hierarchy

Figure 3.3(b) shows how to integrate the two-streams at the multi-scale feature hierarchy level by fusing $\mathcal{F}^{MS}_{\text{rgb}}$ and $\mathcal{F}^{MS}_{\text{flow}}$ to produce a fused multi-scale feature hierarchy $\mathcal{F}^{MS}_{\text{fused}}$. Predictions are made on each feature map in $\mathcal{F}^{MS}_{\text{fused}}$.

3.3.4 Training and Inference:

3.3.4.1 Ground-truth Matching and Hard-negative Mining

During training, each of the K activity predictions at each temporal location in a feature map is labeled as *positive* if the temporal Intersection-over-Union ($tIoU$) overlap between the predicted activity segment and any of the ground-truth segments is greater than 0.5, otherwise *negative*. Following [53], we also adopt *hard negative* mining by including the *negative* samples that have an $p_{ov} > 0.5$. To balance the *positive* and *negative* samples, we randomly select negative samples from the remaining pool until we achieve a $1:1$ ratio of *positive* to *negative* samples.

3.3.5 Loss Function

The proposed detection framework is optimized based on a multi-task loss $\mathcal{L}$ which is composed of activity classification loss $\mathcal{L}_{\text{cls}}$, activity localization loss $\mathcal{L}_{\text{loc}}$, and activity

overlap loss $\mathcal{L}_{\text{ov}}$:

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \beta\mathcal{L}_{\text{loc}} + \gamma\mathcal{L}_{\text{ov}} \tag{3.10}$$

where β and γ are hyper-parameters used to trade-off among the three different losses.

The activity classification loss $\mathcal{L}_{\text{cls}}$ tries to optimize the network parameters to learn the activity classes, while the activity localization loss $\mathcal{L}_{\text{loc}}$ helps learn the parameters in order to align the predicted activity segments with their matched ground-truth activity segments. The activity overlap loss $\mathcal{L}_{\text{ov}}$, on the other hand, helps to have a higher overlap between the predicted and the ground-truth activity segments. We used the standard *Softmax* loss for $\mathcal{L}_{\text{cls}}$, and *Smooth L1 loss* S_{L1} [34] for both $\mathcal{L}_{\text{loc}}$ and $\mathcal{L}_{\text{ov}}$. These losses are formulated as follows:

$$\mathcal{L}_{\text{cls}} = \frac{1}{N}\sum_{n=1}^{N}(-\log P_c^n) \quad \text{and} \quad P_c^n = \frac{\exp(p_c^n)}{\sum_{j=1}^{C}\exp(p_j^n)} \tag{3.11}$$

$$\mathcal{L}_{\text{loc}} = \frac{1}{N_{\text{pos}}}\sum_{n=1}^{N_{\text{pos}}}(S_{L1}(\phi_c^n - g_c^n) + S_{L1}(\phi_w^n - g_w^n)) \tag{3.12}$$

$$\mathcal{L}_{\text{ov}} = \frac{1}{N}\sum_{n=1}^{N} S_{L1}(p_{ov}^n - g_{IoU}^n) \tag{3.13}$$

where N is the total number of samples, N_{pos} is the total number of *positive* samples, and $p_j^n, \phi_c^n, \phi_w^n, p_{ov}^n$ represent the n-th predicted segment's prediction score for the j-th class, predicted center, predicted width, and predicted overlap with the corresponding ground-truth segment, respectively. While $c, g_c^n, g_w^n, g_{\text{IoU}}^n$ represent ground-truth activity segment's class

label, center, width, and tIoU overlap with the n-th predicted activity segment, respectively.

3.3.5.1 Inference

During inference, an activity prediction instance is made up as $\psi = \{\phi_{start}, \phi_{end}, c, p\}$, where, $c = argmax(\{p_r\}_{r=1}^{C})$ denotes the final class prediction, and $p = max(\{p_r\}_{r=1}^{C}) \cdot p_{ov}$ is the final prediction confidence score. Finally, we conduct *NMS* on these predictions based on the confidence score p to remove any redundant prediction.

3.4 Experimental Setup

3.4.1 Datasets

We conducted experiments on two challenging temporal activity detection benchmarks, namely THUMOS'14 [43] and MEXaction2 [106]. The details of the datasets are provided below.

THUMOS'14 [43] contains over 22 hours of video spanning over different sports activities. The dataset is particularly challenging since each video is on average more than 3 minutes long and contains approximately 15 activity instances with 71% background [42]. While the training videos are all trimmed, the validation set and test set contain 200 and 213 temporally untrimmed videos respectively, with temporal annotations for 20 different activity classes. Following the standard practice on this dataset for this task, we perform

training on the validation set and report results on the test set.

MEXaction2 [106] dataset, on the other hand, consists of approximately 77 hours of untrimmed videos extracted from the archives of the Institut National de l'Audiovisuel (INA), plus some trimmed video clips from YouTube and UCF101 [107]. There are only two activity categories: "HorseRiding" and "BullChargeCape".

3.4.2 Data Preparation

Following SSAD [53], we perform activity detection on clips of length T by densely sampling frames from each untrimmed video. During training, neighboring clips are generated with a 75% overlap to handle activity instances located near the boundary and also to increase the amount of training data. During inference, the overlap is limited to 50% for faster processing. We set T=512 as almost 99% of the activity instances in both THUMOS'14 and MEXaction2 datasets have lengths smaller than 512 frames.

3.4.3 Implementation Details

We initialize RGB and FLOW branches of the 3D CNN feature extractor with the pre-trained weights of D3D [9] and S-3DG [8], respectively. Optical flow is computed using TV-L1 algorithm [108]. We set $K = 5$ with scale ratios $\{0.5, 0.75, 1.0, 1.5, 2.0\}$, *SumPooling* window-size $J = 2$, *Dropout* ratio = 0.2 (for *MFB* and *MFB_new*), α_1 and α_2 to 0.1, β and γ to 10, and tIoU threshold for *NMS* to 0.2. These parameter choices are all based on a small

sub-set of the training data earmarked for hyper-parameter tuning.

3.4.4 Evaluation Metrics

Apart from activity detection in the TAD setting, we also evaluate our approach on another relevant task called Temporal Activity Proposal (TAP) that mainly focuses on the quality of the predicted activity proposals regardless of the class predictions. The performance metrics used for the TAD and TAP tasks are defined below.

Mean Average Precision: For the TAD task, we use the standard performance measure called *mean Average Precision (mAP)*. *mAP* is computed as the mean of the *Average Precision (AP)* values over the different categories present in the dataset. On the other hand, *AP* for a specific category is defined as the weighted mean of the precision values computed at each possible confidence threshold varied from one to zero, with the increase in recall from the previous threshold used as the weight.

$$mAP = \frac{1}{C}\sum_{i=1}^{C} AP_i \ , \tag{3.14}$$

$$AP = \sum_{k=1}^{N-1} [recall@k - recall@(k-1)] \times \max(precision@k, precision@(k-1)) \ , \tag{3.15}$$

Here, C is the total number of categories in the dataset, and N is the number of possible confidence thresholds varied from 1 to 0. In practice, a fixed number of points are evaluated in the range $[0, 1]$ for the value of N.

On the other hand, *precision* and *recall*, , are defined as follows:

$$precision = \frac{\#TP}{\#TP + \#FP} \; ; \qquad recall = \frac{\#TP}{\#TP + \#FN} \; , \tag{3.16}$$

Here, *#TP, #FP, #FN* denote *true positive*, *false positive*, and *false negative* counts, respectively.

Finally, a prediction is considered to be *true positive*, if the temporal IoU of the predicted activity segment with the matching ground-truth is above a threshold, otherwise *false negative*. Any prediction without any matching ground-truth is considered *false negative*.

***Average Recall vs. Average Number of Proposals*:** As for the TAP task, we use the standard metric called *Average Recall vs. Average Number of Proposals (AR-AN)*, which denotes the average recall value (AR) at varying average number of proposals (AN) generated per video. Average recall (AR), on the other hand, is defined as the average of the *recall* computed by varying the $tIoU$ threshold from 0.5 to 1.0 with a step size of 0.05.

How to Fuse	*Where* to Fuse		
	Mid-level		*Late*
	$\mathcal{F}^{'}$	$\mathcal{F}^{MS}$	
Averaging			46.42
Sum	47.08	48.48	
Max	46.09	47.53	
Convolution	46.86	47.45	
MLB	46.65	47.77	
MFB	38.29	49.85	
MFB_new	37.47	**50.88**	

Table 3.1: mAP(%) comparisons among the different *mid-level fusion* methods, along with the *late fusion* approach, when the feature extractor is kept fixed. Results are reported on the THUMOS'14 dataset.

3.5 Results

3.5.1 Evaluation on *How* and *Where* to Fuse

In order to evaluate the different *mid-level fusion* strategies as discussed in Sec. 3.3.2 and 3.3.3, models are trained on the THUMOS'14 dataset by extracting two-stream features ($\mathcal{F}^{'}_{rgb}, \mathcal{F}^{'}_{flow}$), while keeping the feature extractor 3D CNN fixed, followed by fusion of the two streams using the network architectures shown in Fig. 3.3. Table 3.1 shows the results of the different fusion methods. As evident from the table, the *mid-level fusion* strategies perform better than *late fusion*. This can be explained by the fact that *mid-level fusion* utilizes the correlation between features from the different modalities, thus making them more discriminative in the common feature space than their individual feature space [109].

Furthermore, the proposed fusion methods outperform the conventional methods (e.g., *sum, max, convolution*), with the fusion at $\mathcal{F}^{MS}$ level (refer to Sec. 3.3.3.2) producing better results than fusion at the $\mathcal{F}^{'}$ level (refer to Sec. 3.3.3.1).

Notably, the performance gain is more strongly pronounced for *MFB* and *MFB_new* with the latter outperforming the other methods, thereby attesting to its superior representation capacity. Subsequent discussions regarding *MFB* and *MFB_new* models refer to fusion at the $\mathcal{F}^{MS}$ level, unless explicitly mentioned otherwise.

3.5.2 Evaluation on End-to-End Feature Learning

To demonstrate the effect of end-to-end feature learning, we consider three different training configurations based on THUMOS'14 as follows:

1. *Fixed*, which does not train the feature extractor 3D CNN. This configuration is the same used for the experiments described in Sec. 3.5.1.

2. *5c*, which trains only the last convolutional layer of the feature extractor 3D CNN (i.e., *'Mixed_5c'* layer); and

3. *5b_5c*, which trains the last two convolutional layers of the feature extractor 3D CNN (i.e.; *'Mixed_5b' and 'Mixed_5c'* layers).

Table 3.2(a) shows the results. The top part of the table shows the performance of the baseline model using *late fusion*, as well as the models using *MFB* and *MFB_new* fusion

Model	*Fixed*	*5c*	*5b_5c*
Baseline	46.42	47.48	48.97
MFB	49.85	51.39	52.60
MFB_new	**50.88**	**52.08**	**53.95**
RGB	-	-	47.84
FLOW	-	-	48.56

(a)

Model	
Decouple-SSAD [55]	44.20
Decouple-SSAD+*MFB*	47.25
Decouple-SSAD+***MFB_new***	**48.27**

(b)

Table 3.2: (a) *mAP(%)* comparison for end-to-end training. Results for the fused models appear on the top part of the table, while that for the single-stream models are reported on the bottom part. Column *Fixed* copies the results for the corresponding methods from Tab. 3.1 with *late fusion* used as the Baseline. (b) *mAP(%)* comparison of Decouple-SSAD [55] and its variants based on *MFB* and *MFB_new*.

under the different training configurations. As obvious from the table, end-to-end feature learning improves performance of activity detection by a noticeable margin for all 3 models, with *MFB_new* gaining 1.20% and 3.07% absolute *mAP* improvement for the *5c* and *5b_5c* configurations over the *Fixed* training configuration, respectively. This validates our design choice for end-to-end learning, which the existing state-of-the-art single-stage approaches (e.g., [53–55]) could not offer. It is noteworthy to mention that we did not notice any further performance improvement by training more layers of the 3D CNN which could be attributed to the fact that the amount of training data for THUMOS'14 is not sufficient to train all layers of a 3D CNN as was also reported in [40]. All results discussed in the following sections are based on the models trained using *5b_5c* configuration, unless explicitly mentioned otherwise.

3.5.3 Ablation Study

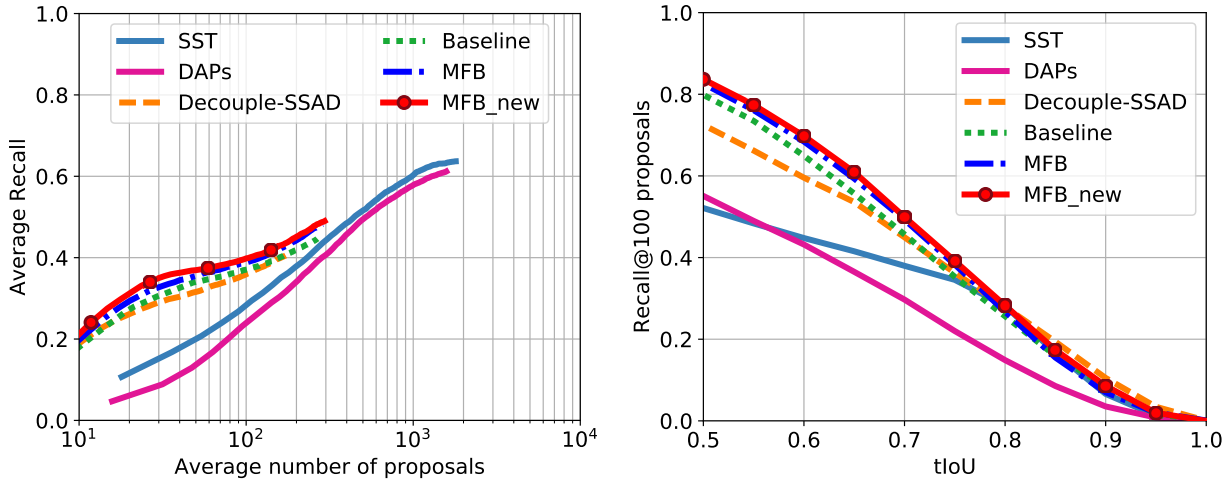
To validate our design choice for using both RGB and FLOW streams, we train models on THUMOS'14 based on the individual streams and compare the results with the fused models. As shown at the bottom part of Table 3.2(a), the fused models outperform both of the single-stream models, thus validating our design choice. These results are in agreement with the observations reported in the video activity recognition literature (e.g., [3, 4, 6, 102]).

We, further, investigate the efficacy of the proposed fusion strategies across other methods for TAD. To this end, we choose Decouple-SSAD [55], the existing state-of-the-art method on THUMOS'14, which is also based on a multi-scale feature hierarchy. Using the author-provided code[1], we train Decouple-SSAD by replacing its *late fusion* strategy with *mid-level fusion* based on *MFB* and *MFB_new*. Table 3.2(b) compares the performance of the original Decouple-SSAD with its variants. As demonstrated in the table, the proposed fusion strategies boost the performance of the native Decoupe-SSAD method. Notably, *MFB_new* outperforms *MFB* even when used with other TAD methods, thus confirming its versatility and generalizability across other methods.

3.5.4 State-of-the-Art Comparison – Temporal Activity Proposal

Before evaluating our approach on the TAD task, we first assess the boundary quality of the activity segments as predicted by our proposed methods regardless of their class predictions.

[1] `https://github.com/HYPJUDY/Decouple-SSAD`



(a) **(b)**

Figure 3.4: (a) AR-AN curves on THUMOS'14. (b) Recall@100 vs. *tIoU* curves on THUMOS'14.

A successful approach should be able to retrieve high-quality activity segments having larger *tIoU* overlaps with the ground-truth segments and generate high recall. More importantly, a successful approach should generate fewer activity segments (a.k.a proposals) to reduce the computational cost of the succeeding steps. We compare our method with two state-of-the-art methods, namely DAPs [44] and SST [110] that are specifically designed for activity proposal generation, along with the current state-of-the-art TAD method Decouple-SSAD [55] on the THUMOS'14 benchmark.

As Figure 3.4(a) depicts, at low average number of proposals (i.e., AN values), our proposed models (Baseline, *MFB*, and *MFB_new*) outperform the other methods, with *MFB_new* delivering the best performance. This clearly suggests that our top predictions

are much more likely to contain activity segments than the other methods. However, just like in the case of Decouple-SSAD [55], the average recall (AR) of our proposed models reaches a saturation point more rapidly when compared to the proposal generation methods. This is primarily because the proposal generation methods produce a significantly higher number of proposals in contrast to the activity detection methods, which subsequently leads to an increase in the computational cost of the subsequent steps.

To further zoom into the boundary quality of the predicted activity segments, we plot AR values for the top 100 predictions against higher $tIoU$ thresholds as shown in Fig. 3.4(b). *MFB_new* outperforms all other methods for most of the $tIoU$ thresholds. Overall, our proposed models consistently outperform others in predicting high-quality activity segments as demonstrated across various metrics. Additionally, the proposed methods generate a minimal number of activity proposals.

3.5.5 State-of-the-Art Comparison – Temporal Activity Detection

Table 3.3(a) shows the performance of our proposed methods in comparison with the state-of-the-art methods on the TAD task at various $tIoU$ thresholds ranging from 0.1 to 0.7 on the THUMOS'14 benchmark. As the table reveals, our proposed models (Baseline, *MFB*, and *MFB_new*) consistently and significantly outperform the other methods across a range of $tIoU$ thresholds. Of particular interest is $tIoU$=0.5, where *MFB_new* sets new state-of-the-art results outperforming the existing state-of-the-art method

Stage	Method	*mAP* @*tIoU* (%)						
		0.1	0.2	0.3	0.4	0.5	0.6	0.7
Two-Stage	SCNN [33]	47.7	43.5	36.3	28.7	19.0	10.3	5.3
	DAPs [44]	-	-	-	-	13.9	-	-
	SST [110]	-	-	41.2	31.5	20.0	10.9	4.7
	TCN [36]	-	-	-	33.3	25.6	15.9	9.0
	R-C3D [39]	54.5	51.5	44.8	35.6	28.9	19.1	9.3
	SSN [111]	66.0	59.4	51.9	41.0	29.8	19.6	10.7
	CBR [37]	60.1	56.7	50.1	41.3	31.0	19.1	9.9
	BSN [112]	-	-	53.5	45.0	36.9	28.4	20.0
	TAL-net [42]	59.8	57.1	53.2	48.5	42.8	33.8	**20.8**
Single-Stage	SSAD [53]	50.1	47.8	43.0	35.0	24.6	15.4	7.7
	SS-TAD [49]	-	-	45.7	-	29.2	-	9.6
	G-TAN [54]	69.1	63.7	57.8	47.2	38.8	-	-
	DSSAD [55]	-	-	60.2	54.1	44.2	32.3	19.1
	Baseline	69.5	68.4	66.1	61.1	49.0	32.9	16.7
	MFB	70.7	69.6	67.1	61.9	52.6	35.5	18.0
	MFB_new	**73.0**	**71.9**	**69.2**	**65.0**	**53.9**	**38.1**	19.8

(a)

SCNN [33]	SSAD [53]	***MFB_new***
7.4	11.0	**16.4**

(b)

Table 3.3: Comparison of our proposed method with the state-of-the-art TAD methods. (a) *mAP(%)* comparison at various *tIoU* thresholds on THUMOS'14. (b) *mAP(%)* comparisons at *tIoU*=0.5 on MEXaction2.

Decouple-SSAD [55] by an absolute **9.7% *mAP* (53.9% vs. 44.2%)**. Class-wise AP comparisons between *MFB_new* and Decouple-SSAD [55] are shown in Table 3.4. Our proposed approach outperforms Decouple-SSAD [55] in **16** out of **20** activity categories.

Method	Baseball Pitch	Basketball Dunk	Billiards	Clean and Jerk	Cliff Diving	Cricket Bowling	Cricket Shot	Diving	Frisbee Catch	Golf Swing	Hammer Throw	High Jump	Javelin Throw	Long Jump	Pole Vault	Shotput	Soccer Penalty	Tennis Swing	Throw Discus	Volleyball Spiking
[55]	33.2	28.3	7.6	**48.1**	56.9	14.0	5.5	58.8	11.9	42.1	**75.2**	76.6	**83.8**	**94.6**	84.0	40.8	15.1	9.3	71.4	16.8
MFB_new	**43.0**	**54.6**	**8.0**	44.6	**75.9**	**43.7**	**28.7**	**76.7**	**34.9**	**57.8**	73.0	**79.9**	76.1	82.5	**84.6**	**41.6**	**32.9**	**24.4**	**80.5**	**35.5**

Table 3.4: Class-wise *AP(%)* comparison between Decouple-SSAD [55] and *MFB_new* at *tIoU*=0.5 on THUMOS'14. *MFB_new* outperforms Decouple-SSAD in 16 out of 20 categories.

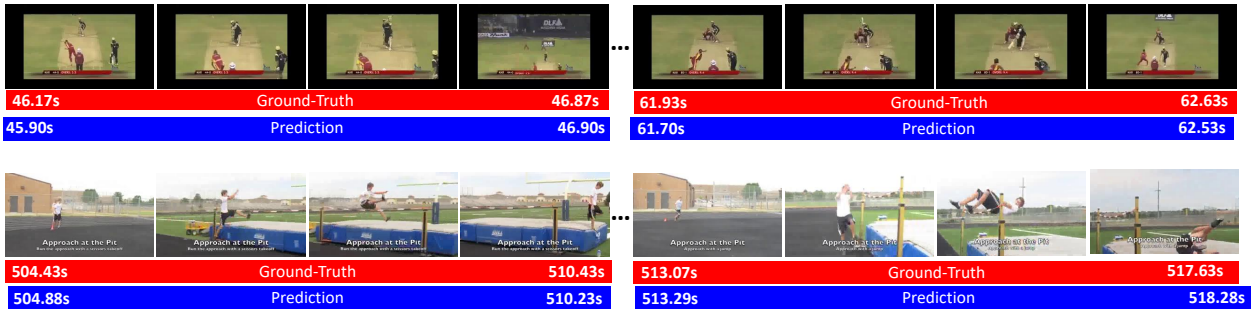


Figure 3.5: Visualization of the top predicted activity instances on two test videos from THUMOS'14. For each video, the first row shows some representative frames from two consecutive activity segments, while the second and third row represent the ground-truth (in red) and the predictions (in blue) of our proposed model *MFB_new*, respectively.

3.5.6 Results on MEXaction2

Table 3.3(b) reports results on the MEXaction2 dataset. We compare *MFB_new* with SCNN [33] and SSAD [53], as these are the state-of-the-art methods on this dataset. We outperform both methods by significant margin pushing the current state-of-the-art from **11%** ***mAP*** to **16.4%** ***mAP***.

3.5.7 Qualitative Results

Figure 3.5 shows some qualitative results of our proposed approach on THUMOS'14. For each example, predictions are visualized for two consecutive ground-truth activity segments from a test video. As can be seen, our proposed approach can accurately localize and classify the activity segments, while being able to handle moderate variations in activity duration (e.g., short and long activity instances).

3.6 Summary

In this chapter, we have presented an end-to-end deep learning framework for temporal activity detection in untrimmed videos. We have also demonstrated effective ways to fuse multi-stream feature representations of a video and proposed a novel *mid-level fusion* strategy for combining appearance and motion information of a video in order to enhance the performance of activity detection in the TAD setting. The proposed method is empirically evaluated on the highly challenging THUMOS'14 and MEXaction2 benchmarks, and outperforms the current state-of-the-art methods on both benchmarks.

Chapter 4

From Temporal to Spatio–Temporal Activity Detection

4.1 Introduction

In this chapter, we discuss how activity detection from the temporal domain (i.e., TAD setting) can be extended to the spatio-temporal domain (i.e., STAD setting). To this end, we propose an efficient approach to jointly optimize spatial and temporal localization of activities using a single-stage deep learning framework in a holistic end-to-end manner.

Spatio-temporal activity detection requires classifying as well as localizing human activities both in space and time in temporally untrimmed videos. While activity detection in the TAD setting is able to address the ***'what'*** and ***'when'*** aspects of the problem as

discussed in the Introduction (Chap. 1) and the previous chapter, it fails to answer ***'where'*** in the video frames the activities are happening. However, there are applications (e.g., video surveillance, advanced video search) that require not only the temporal boundaries of the activities but also the spatial extents of the actors within the individual video frames to fully comprehend the scene dynamics. This has essentially resulted in the need for activity detection in the STAD setting. This chapter focuses on tackling the STAD problem, which poses an even greater challenge compared to the TAD problem, primarily due to the vast search space that needs to be explored in both spatial and temporal dimensions.

As discussed in Chapter 2, the typical recipe of STAD is mainly based on the object detection and linking pipeline, where actors are detected on individual video frames which are then linked via complex heuristics-based methods (e.g., dynamic programming [67–72], or temporal sliding window [73]). However, frame-based methods fail to fully capture the temporal structure of the activities, and as such, struggle to disambiguate activities that require the temporal contexts to comprehend (e.g., *sitting down* vs. *standing up*) [75]. Therefore, recent methods [74, 75] adopt a multi-frame approach that takes a short sequence of frames as input and performs localization of activities over short tubelets. However, due to the lack of direct temporal regression, these methods still need to rely on complex optimization for stitching the tubelets which may not result in optimal outputs [29]. Not to mention, the short temporal footprint of these methods presents an

obstacle to achieving satisfactory performance on complex and longer activities.

Furthermore, the existing SOTA multi-frame methods [78,79,113] do not incorporate joint optimization for both spatial and temporal activity localization, instead relying on separate, disconnected pipelines. This disjoint approach incurs heightened computational demands likely due to redundant processing, increases the likelihood of sub-optimal outcomes, and most importantly, it impedes the ability to train these methods in a holistic, end-to-end fashion [114]. There has been limited research in this direction, and the few proposed methods, such as STAR [29], fall short of performance.

The goal of this chapter is to address the aforementioned gaps in activity detection in the STAD setting and propose an efficient solution to jointly optimize both spatial and temporal localization of activities simultaneously. We are particularly inspired by the impressive success of the SOTA methods for TAD that employ direct temporal regression (e.g., [42, 49, 53–55, 115]), hence propose to incorporate the same for activity detection in the STAD setting. To this end, we build upon the TAD framework proposed in Chapter 3, and extend it to incorporate a spatial localization branch. With two parallel branches, the spatial localization branch performs actor detection and actor tube building, while the temporal localization branch pinpoints the start and end times as well as classifies the activities taking the actor tubes as input.

Contributions: The main contributions in this chapter are summarized as follows:

1. We propose a novel end-to-end trainable deep learning framework to jointly optimize

spatial and temporal localization of activities.

2. We directly regress on the temporal bounds of the activities and integrate it into the training process to increase temporal localization accuracy. This is in contrast to the existing expensive and complicated temporal solutions that fall short of temporal localization accuracy [116].

3. We introduce spatio-temporal Non-maximum Suppression (NMS), a variant of the NMS technique, to improve the performance of the STAD task.

4. We evaluate the effectiveness of our proposed approach on the challenging UCF101-24 benchmark for STAD, and set new state-of-the-art results.

Related Publications: The methods and results presented in this chapter have been accepted for publication in the following conference.

- **M. A. Rahman** and R. Laganière, "Spatio-temporal activity detection via joint optimization of spatial and temporal localization," accepted in *IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW), USA*, 2024.

4.2 Problem Statement

The problem definition of STAD is formulated as follows. Given a temporally untrimmed long video sequence $\mathcal{I} = \{i_1, i_2, \ldots, i_T\}$ containing T frames, the goal of the STAD task

is to output a set $\Psi = \{\psi_1, \psi_2, \ldots, \psi_N\}$ consisting of N predictions of spatio-temporal activity segments such that each predicted activity segment $\psi_n = (\phi^n_{start}, \phi^n_{end}, c^n, p^n, R^n = \{r^n_1 \ldots r^n_M\})$ has associated with it, the activity (start, end) times $(\phi^n_{start}, \phi^n_{end})$, the predicted activity category $c^n \in \{1, \ldots, C\}$, the prediction confidence score p^n, as well as a set $R^n = \{r^n_1 \ldots r^n_M\}$ containing M predicted bounding boxes tightly encapsulating the actor on each of the M frames belonging to the video segment between ϕ^n_{start} and ϕ^n_{end}. Here, C denotes the total number of activity categories.

4.3 Proposed Approach

The proposed STAD framework is illustrated in Fig. 4.1. Building upon the TAD framework proposed in Chapter 3, the STAD framework comprises five main modules – (i) base network, (ii) spatial localization branch, (iii) actor tube building, (iv) temporal localization branch, and (v) spatio-temporal NMS.

4.3.1 Base Network

We use the same two-stream 3D CNN, called S-3DG [8], as the base network for feature extraction in the proposed STAD framework, just as used in the TAD framework. As a quick recap, the RGB stream of S-3DG is fed with a video segment $\mathcal{I}_{\text{rgb}} \in \mathbb{R}^{T \times H \times W \times 3}$ consisting of T number of RGB frames, each with height, width and channel dimension of H, W, and 3 respectively. The FLOW stream, on the other hand, takes the optical FLOW

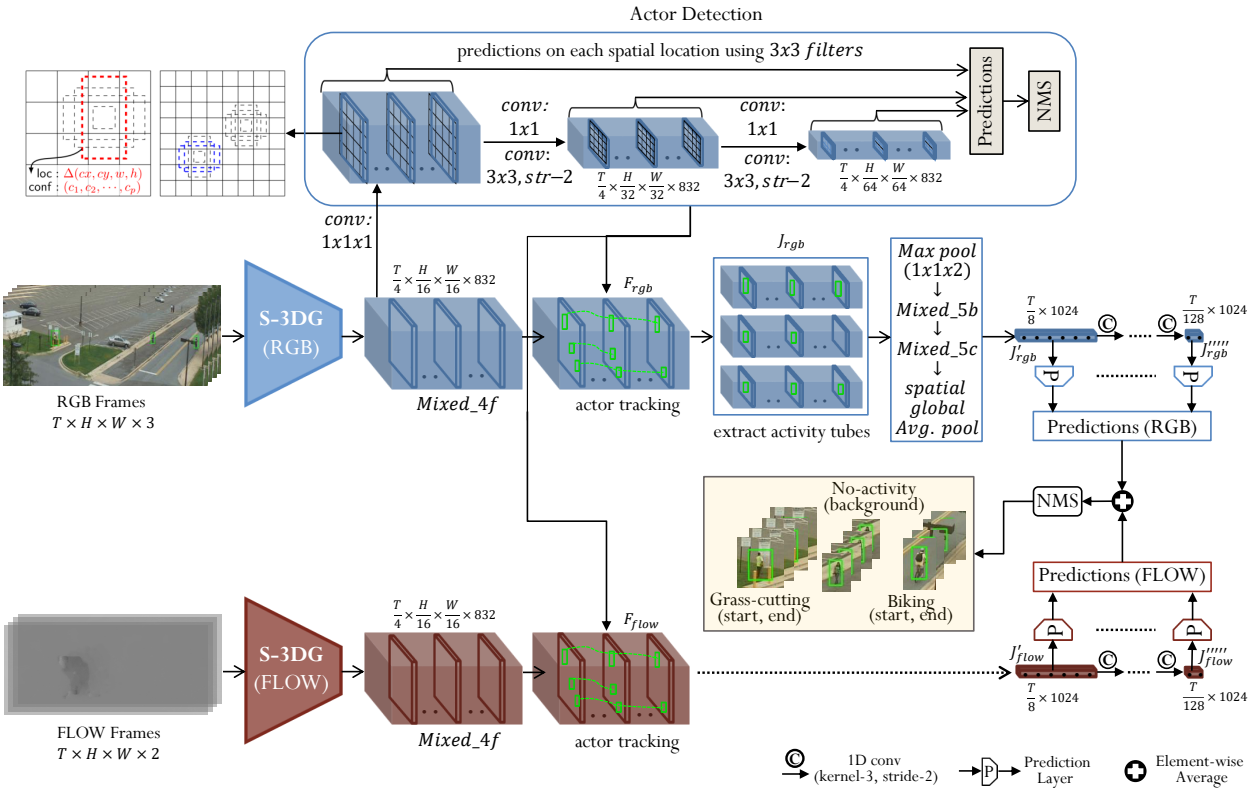


Figure 4.1: Architecture of the proposed end-to-end STAD framework. Building upon the TAD framework depicted in Fig. 3.1, we add a spatial localization branch on top of the *Mixed_4f* block of the RGB stream, employing a multi-scale feature hierarchy on the temporal slices of *Mixed_4f* to perform actor detection, followed by tracking of the actors. The actor tracks are then projected back to the *Mixed_4f* blocks of both RGB and FLOW streams to extract actor tube features, which are eventually fed as input to the multi-scale temporal localization branch for direct temporal regression of activities. The entire network is trained end-to-end with joint optimization of the spatial and temporal localization branches.

frames $\mathcal{I}_{\text{flow}} \in \mathbb{R}^{T\times H\times W\times 2}$ corresponding to the RGB frames as input, thus having the same temporal and spatial dimensions as the RGB input, but with channel dimension 2 (i.e., FLOW in X and Y directions). The basic idea behind this two-stream architecture is to

capture the appearance information of the video through the RGB stream, while harvesting the motion information through the FLOW stream. We extract rich two-stream spatio-temporal feature representations ($\mathcal{F}_{\text{rgb}}$, $\mathcal{F}_{\text{flow}}$) $\in \mathbb{R}^{\frac{T}{4}\times\frac{H}{16}\times\frac{W}{16}\times 832}$ of the input video from the *Mixed_4f* block of S-3DG. We then exploit ($\mathcal{F}_{\text{rgb}}$, $\mathcal{F}_{\text{flow}}$) as base feature maps that are shared among the spatial and temporal localization modules, thus allowing these feature maps to be learned end-to-end with respect to the overall objective of the STAD task. The base network being fully-convolutional, the length of the video sequence T can be arbitrarily long, consequently constrained only by the amount of physical memory.

4.3.2 Spatial Localization Branch

The spatial localization branch is responsible for detecting the actors (i.e., persons) in the video frames. To this end, we feed the base RGB feature map $\mathcal{F}_{\text{rgb}}$ as input to this branch, after having it passed through a 1×1×1 convolution layer as shown in Fig. 4.1. We then employ a multi-scale spatial feature hierarchy on top of the temporal slices of $\mathcal{F}_{\text{rgb}}$ with a view to detecting actors (i.e., persons) at varying scales. To this end, we fuse the temporal dimension of F_{rgb} with the batch dimension before repeatedly applying 1×1 convolutions (with stride 1), followed by 3×3 convolutions (with stride 2) to generate a multi-scale feature hierarchy. Such a multi-scale feature hierarchy helps achieve superior generalization ability over different scales of persons as demonstrated in our earlier work [117]. This is particularly crucial in the context of realistic videos that tend to exhibit wide variability in person scales.

Leveraging the multi-scale feature hierarchy, we adopt an anchor-based prediction architecture, as commonly used in the single-stage object detection methods (e.g., [18, 51, 52]), to perform predictions of activity class and actor locations. Anchor-based prediction for object detection essentially employs a set of default bounding boxes with varying scales and aspect ratios at each spatial location in a feature map. Each default box has it's own default center, width, and height. Predictions about the activity class confidence scores along with the width and height of the persons with respect to the default boxes' widths and heights are then generated using 3×3 convolutions, as shown on the top-left part in Fig. 4.1. Finally, the predictions are post-processed using NMS. Please refer to the single-stage object detection method called SSD [18] for more details about the anchor-based prediction architecture.

4.3.3 Actor Tube Building

With the actors detected and localized by the spatial localization branch, each detected actor is then tracked using an online tracker called DeepSORT [118]. After that, the actor tracks are projected back to the two-stream base feature maps ($\mathcal{F}_{\text{rgb}}$, $\mathcal{F}_{\text{flow}}$) to extract spatio-temporal features for the tubes corresponding to the actors. To this end, for each actor track, the spatial region corresponding to the actor is cropped out of the temporal slices of ($\mathcal{F}_{\text{rgb}}$, $\mathcal{F}_{\text{flow}}$) in order to create two-stream spatio-temporal tube feature maps ($\mathcal{J}_{\text{rgb}}$, $\mathcal{J}_{\text{flow}}$) having the same temporal and channel dimensions as ($\mathcal{F}_{\text{rgb}}$, $\mathcal{F}_{\text{flow}}$). The spatial region is

cropped based on the minimal square box that encapsulates all bounding boxes belonging to the actor track, after having the box slightly expanded to capture sufficient contexts around the actor.

4.3.4 Temporal Localization Branch

The purpose of the temporal localization branch is to locate the activities in time and classify them. To this end, the tube feature maps ($\mathcal{J}_{\text{rgb}}$, $\mathcal{J}_{\text{flow}}$) corresponding to the detected actors are arranged into a batch after having them resized spatially to a common dimension $D \times D$. These resized feature maps are then *max-pooled* using $1 \times 1 \times 2$ filter to reduce the temporal dimension by half. They are subsequently passed through the *Mixed_5b* and *Mixed_5c* blocks of S-3DG before their spatial dimension is completely collapsed using *global spatial average pooling* to produce two-stream temporal-only feature maps ($\mathcal{J}'_{\text{rgb}}$, $\mathcal{J}'_{\text{flow}}$) $\in \mathbb{R}^{\frac{T}{8} \times 1024}$ as shown in the middle-right of Fig. 4.1. We then feed ($\mathcal{J}'_{\text{rgb}}$, $\mathcal{J}'_{\text{flow}}$) to our TAD framework to realize temporal localization and classification of the activities happening in each actor tube. Since the TAD framework itself employs multi-scale temporal feature hierarchy, it can handle wide variations in activity lengths, a common phenomenon in the case of realistic videos. Please refer to Sec. 3.3.1 for details about the TAD framework.

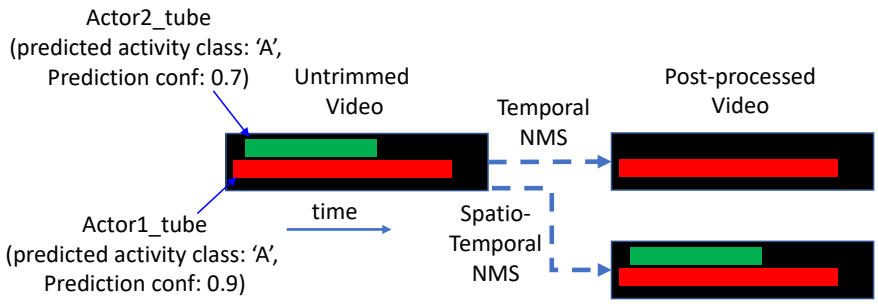


Figure 4.2: Illustration of the spatio-temporal NMS technique. Temporal NMS would remove *'Actor2_tube'* as it overlaps in time with the higher confidence and same class prediction *'Actor1_tube'*. The proposed spatio-temporal NMS would retain both tubes as they do not overlap both in time and space.

4.3.5 Spatio-Temporal NMS

We introduce spatio-temporal NMS in this work to post-process the activity predictions generated by the temporal localization branch. Unlike the temporal NMS technique commonly used in the literature (e.g., [42, 49, 53–55, 115]), which cancels out duplicate temporal segments solely based on the temporal overlaps among the candidate predictions, the proposed spatio-temporal NMS technique eliminates duplicates by considering both spatial and temporal overlaps among the predicted spatio-temporal segments. This is motivated by the observation that in an untrimmed and unconstrained video, activities of the same type may occur concurrently, with the actors located spatially close to each other (e.g., two persons cycling side-by-side). Under this scenario, the temporal NMS technique would remove all but the temporal segment with the highest confidence score, as they

overlap in time and have the same class predictions. On the other hand, the proposed spatio-temporal NMS technique would be able to retain all spatio-temporal segments as long as they do not overlap both in space and time. This is illustrated in Fig. 4.2.

4.4 Training and Inference

4.4.1 Loss Function

The proposed STAD framework was trained using a multi-task loss $\mathcal{L}$ having two components – object detection loss $\mathcal{L}^{\text{obj}}$, and temporal detection loss $\mathcal{L}^{\text{tem}}$. $\mathcal{L}^{\text{obj}}$, in turn, comprises object classification loss $\mathcal{L}^{\text{obj}}_{\text{cls}}$, and object localization loss $\mathcal{L}^{\text{obj}}_{\text{loc}}$ (for actor width, height, center-X, and center-Y). On the other hand, $\mathcal{L}^{\text{temp}}$ consists in temporal classification loss $\mathcal{L}^{\text{tem}}_{\text{cls}}$, temporal localization loss $\mathcal{L}^{\text{tem}}_{\text{loc}}$ (for activity start and end times), and temporal overlap loss $\mathcal{L}^{\text{tem}}_{\text{ov}}$. Equation 4.1 shows the formulation of the loss function.

$$\begin{aligned} \mathcal{L} &= \mathcal{L}^{\text{obj}} + \lambda \mathcal{L}^{\text{tem}} \\ &= \mathcal{L}^{\text{obj}}_{\text{cls}} + \mathcal{L}^{\text{obj}}_{\text{loc}} + \lambda(\mathcal{L}^{\text{tem}}_{\text{cls}} + \mathcal{L}^{\text{tem}}_{\text{loc}} + \mathcal{L}^{\text{tem}}_{\text{ov}}) \end{aligned} \tag{4.1}$$

Here, λ is a hyper-parameter used to trade-off between the two loss components. We used *multi-class cross-entropy loss* for $\mathcal{L}^{\text{obj}}_{\text{cls}}$ and $\mathcal{L}^{\text{tem}}_{\text{cls}}$, while *Smooth-L1 loss* was used for $\mathcal{L}^{\text{obj}}_{\text{loc}}$, $\mathcal{L}^{\text{tem}}_{\text{loc}}$, and $\mathcal{L}^{\text{tem}}_{\text{ov}}$. Please refer to Sec. 3.3.5 for details about $\mathcal{L}^{\text{tem}}_{\text{cls}}$, $\mathcal{L}^{\text{tem}}_{\text{loc}}$, and $\mathcal{L}^{\text{tem}}_{\text{ov}}$.

4.4.2 Final Predictions

The actor detections from the spatial localization branch are available at every l^{th} frame, which is dictated by the temporal stride of the input feature map F_{rgb} and the frame sampling rate. For example, since F_{rgb} has a temporal stride of 4, l will be 4/8/16 for a frame sampling rate of 1/2/4. Therefore, in order to obtain the detections on the intermediate frames, we use linear interpolations. Finally, an spatio-temporal activity prediction instance ψ_n, as defined in Sec. 4.2, is made up by combining the actor bounding boxes $R^n = \{r_1^n \dots r_M^n\}$ with the activity predictions $(\phi_{start}^n, \phi_{end}^n, c^n, p^n)$ as output from the spatio-temporal NMS module, such that $\psi_n = (\phi_{start}^n, \phi_{end}^n, c^n, p^n, R^n = \{r_1^n \dots r_M^n\})$.

4.5 Experimental Setup

4.5.1 Dataset

To evaluate the proposed STAD framework, we performed experiments on the challenging UCF101-24 [119] benchmark, which is one of the largest and most diversified and challenging spatio-temporal action detection datasets containing temporally untrimmed videos. It is derived from the UCF101 [107] dataset by providing spatio-temporal annotations for 24 classes in the form of bounding box and tube annotations for the actors. The dataset is challenging as the average number of action instances in a video is 1.5, with each action

instance spanning 70% of the video duration on average. However, certain classes may have instances with an average duration of as low as 30% of the video length. Keeping aligned with the standard practice on this dataset, and to be able to compare results with the SOTA methods, we used the revised annotations provided by Singh *et al.* [68] that includes 3,207 videos, with 2,293 for training and 914 for testing.

4.5.2 Evaluation Metrics

The proposed approach is evaluated using the established performance metric for the STAD task, which is the *mean Average Precision (mAP)* metric, both at the video-level and frame-level, as documented in existing literature [68, 73, 75, 120].

Video-mAP is used to evaluate the performance of the STAD task at the video-level detections, and involves regressing a series of temporally linked bounding boxes, also known as "activity tubes", along with the relevant class label. It is defined as the mean of the average precision (AP) over all classes. The mathematical definition of mAP and AP appear in Sec. 3.4.4.

For Video-mAP, the IoU is computed as the product of the temporal IoU between ground-truth and predicted activity tubes and the average of the spatial IoUs between the ground-truth and predicted bounding boxes.

Frame-mAP is used to evaluate the detection performance at the frame-level and involves detecting the instance bounding boxes in each video frame along with the

associated class label. It is defined analogously as Video-mAP except that the IoU is computed spatially.

4.5.3 Implementation Details

We apply data augmentation during training. Input images first undergo center-cropping to the desired resolution (i.e., 224x224 for the best model configuration), followed by various data augmentations including random consistent left-right flipping, random cropping with aspect-ratio resizing. In random consistent left-right flipping, either all or none of the frames belonging to the input sequence are flipped horizontally. However, no data augmentation is applied during test. The input sequence length T is set to 352. The base network S-3DG is initialized with pre-trained weights from the Kinetics-600 [121] dataset, while the spatial and temporal localization branches are initialized randomly. The batch size for the temporal branch is set to 4, which requires to set the batch size for the spatial branch to $4\times T$. The spatial branch employs anchors with 6 different scales linearly ranging from $0.2 - 0.95$ and 3 different aspect ratios $\{0.5, 1, 2\}$. On the other hand, the number of default temporal segments K for the temporal branch is set to 5 with scale ratios $\{0.5, 0.75, 1.0, 1.5, 2.0\}$. The whole framework is trained using Adam optimizer with a fixed learning rate of 0.0005. D for feature cropping in the temporal localization branch is set to 10. To generate consistent actor tubes, DeepSORT [118] is applied with a confidence threshold of above 0.3. The whole framework was implemented based on the TensorFlow Object Detection API [122]

Table 4.1: Exploration study on model configuration w.r.t. spatial and temporal resolution as well as input modalities. Both Frame-mAP and Video-mAP are reported at IoU=0.5. Results are reported on the UCF101-24 dataset.

Input	Resolution	Sampling Rate	Frame-mAP (%)	Video-mAP (%)
RGB	160x160	every frame	65.7	54.8
	192x192		71.5	58.2
	224x224		74.9	60.1
RGB	224x224	every 2^{nd} frame	69.4	57.5
		every 4^{th} frame	64.2	53.1
RGB+FLOW	**224x224**	**every frame**	**74.9**	**61.3**

and trained on 2× NVIDIA RTX 3090 GPUs.

4.6 Results

In this section, we present the results of the proposed approach for the STAD task based on the UCF101-24 dataset. In order to validate certain design choices, while also allowing us to determine the optimal model configurations, we first study different model configurations and perform ablation studies. We then compare the results with the SOTA methods based on the optimal model configurations.

4.6.1 Model Configurations

Different input configurations are explored with a view to selecting the optimal model configuration. To be specific, the effect of spatial resolution and temporal sampling rate, as well as the impact of FLOW inputs are studied. Table 4.1 shows the results of the

proposed STAD framework based on the different model configurations. The top and middle part of the table show results as the spatial resolution and temporal sampling rate of the input frame sequence are varied, respectively. Fixing upon the best configurations (i.e., frame resolution of 224×224, while sampling every frame), the bottom part of the table shows the impact of optical FLOW on the STAD task. As revealed from the table, higher spatial resolution and increased frame rate both contribute to superior spatio-temporal detection of the activities, with the addition of optical FLOW input further boosting the performance. The optimal model configuration with RGB and FLOW inputs, as shown in the bottom part of the table, achieves a Frame-mAP of **74.9%**, while reaching a Video-mAP of **61.3%**.

4.6.2 Ablation Study

4.6.2.1 Effect of Temporal Localization

In order to validate our design choice of including end-to-end temporal localization, we conducted experiments with and without temporal localization of the discovered actor tubes. To this end, the best model configuration with RGB input (refer to the top part of Tab. 4.1) is used to train two separate models – one employing temporal localization and classification of the activities in the discovered actor tubes, and the other employing only classification of the actor tubes without performing any temporal localization. The top part of Tab. 4.2 shows the results of this exercise which reveals that temporal localization along with

Table 4.2: Ablation study on the effect of temporal localization and spatio-temporal NMS for the STAD task. Results are reported based on RGB input at IoU=0.5.

Ablation Experiment	Video-mAP (%)
Temporal Classification	56.4
Temporal Localization	**60.1**
Temporal NMS	59.7
Spatio-Temporal NMS	**60.1**

classification of the the actor tubes improves the performance of the STAD task by an absolute 3.7% Video-mAP over only classification of the tubes as adopted by some earlier methods (e.g., [113]).

The impact of temporal localization on the STAD task would be more pronounced in situations where an actor performs consecutive sequential activities. For example, in an autonomous driving scenario, fully comprehending a pedestrian's intentions at, through, and around a road intersection will involve localizing the pedestrian in space and time and recognizing the subsequent actions of the pedestrian as it waits to cross the intersection, then the actual crossing activity itself, and ending in waiting on, or exiting through the other side of the crossing, all of which are to be performed by the same actor consecutively. As a result, the inclusion of direct temporal regression in the STAD task to precisely localize each subsequent activity is crucial for real-world situations.

4.6.3 Effect of Spatio-Temporal NMS

We also perform experiments to tease out the impact of using spatio-temporal NMS as opposed to temporal NMS for the STAD task. In pursuit of this goal, akin to the ablation study for temporal localization, we evaluate the optimal model configuration with RGB input in two distinct scenarios: one utilizing temporal NMS and the other employing spatio-temporal NMS. The bottom part of Tab. 4.2 shows the results indicating that use of spatio-temporal NMS improves the Video-mAP of the proposed STAD framework by 0.4% over using temporal NMS.

4.6.4 State-of-the-Art Comparison

With the optimal model configurations selected from Tab. 4.1, we perform comparisons of our proposed STAD framework with the SOTA methods based on the UCF101-24 dataset. Table 4.3 shows the comparisons in terms of Frame-mAP at the standard IoU threshold of 0.5, and Video-mAP at IoU thresholds of 0.2 and 0.5, keeping it aligned with the standard practice on this dataset. As evident from the table, the multi-frame methods generally perform better than the single-frame methods when it comes to Video-mAP, primarily because the former methods demonstrate better temporal localization ability as they enjoy larger temporal footprints by leveraging multiple frames as input.

Our proposed method, which employs end-to-end temporal localization of activities as

Table 4.3: Comparison of our proposed method with the state-of-the-art STAD methods on the UCF101-24 benchmark based on Frame-mAP (IoU=0.5) and Video-mAP (IoU=0.2 and 0.5). 'R' and 'F' denote RGB and FLOW, respectively.

Temporal Localization	Method	Input	Frame-mAP (%)	Video-mAP (%)	
			0.5	0.2	0.5
Frame-based	Peng *et al.* [123]	R+F	39.6	42.3	-
	Saha *et al.* [120]	R+F	-	66.8	35.9
	Weinzaepfel *et al.* [73]	R+F	-	58.9	-
	AMTnet [76]	R+F	-	78.5	49.7
	Gurkirt *et al.* [68]	R+F	-	73.5	46.3
	Pramono *et. al* [124]	R+F	73.7	80.4	49.5
	Zhao *et al.* [125]	R+F	-	78.5	50.3
Tubelet linking	Chéron *et al.* [126]	R+F	-	76.0	50.1
	Gu *et al.* [113]	R+F	76.3		59.9
	T-CNN [74]	R	41.4	-	47.1
	ACT [75]	R+F	67.1	77.2	51.4
	TACNet [78]	R+F	72.1	77.5	52.9
	STEP [79]	R+F	75.0	76.6	-
	MOC [77]	R+F	**78.0**	82.8	53.8
	3D-RetinaNet [127]	R	75.2	82.4	58.2
End-to-End	STAR [29]	R	63.0	77.9	53.0
	Ours	R	74.9	82.5	**60.1**
	Ours	R+F	74.9	**83.4**	**61.3**

opposed to localization and linking of tubelets, outperforms the SOTA methods in Video-mAP while producing competitive results in Frame-mAP. We achieve a Video-mAP of **60.1%** at the standard IoU threshold of 0.5 using only RGB input; with the addition of FLOW input further pushing the Video-mAP to **61.3%**. On the other hand, the proposed method achieves a Frame-mAP of **74.9%**.

Table 4.4: Class-wise Video-mAP (%) at IoU=0.5 on the UCF101-24 dataset.

Basketball	Basketball Dunk	Biking	Cliff Diving	Cricket Bowling	Diving	Fencing	Floor Gymnastics	Golf Swing	Horse Riding	Ice Dancing	Long Jump	PoleVault	Rope Climbing	Salsa Spin	Skate Boarding	Skiing	Skijet	Soccer Juggling	Surfing	Tennis Swing	Trampoline Jumping	Volleyball Spiking	Walking With Dog
71.1	76.8	56.9	75.2	71.6	85.7	46.3	84.6	76.4	81.1	46.8	78.7	73.5	74.9	35.4	74.5	29.5	30.1	61.3	29.4	56.5	22.1	59.9	77.8

It is noteworthy to mention that methods achieving a higher Frame-mAP than ours typically utilize pre-trained external object detectors (e.g., MOC [77], Gu *et al.* [113]), or depend on significantly higher input resolutions (e.g., STEP [79], 3D-RetinaNet [127]). Additionally, some of these methods capitalize on both RGB and FLOW streams to perform actor detection (e.g., MOC [77]), thereby incurring increased computational demands.

Table 4.4 shows the class-wise Video-mAP for the optimal model configuration. Noteworthy among the top-performing classes are 'Diving,' 'Floor Gymnastics', and 'Horse Riding.' In contrast, the model demonstrates lower performance on classes like 'Trampoline Jumping', 'Skiing,' and 'Skijet.' This could be attributed to challenges these classes pose, such as fast-moving actors and the presence of multiple smaller actors.

4.6.5 Qualitative Results

Figure 4.3 shows some sample qualitative results of the proposed STAD framework on two test videos from the UCF101-24 dataset. Top row shows predictions on the 'Diving' class with a length of 9.3s, while the bottom row shows predictions for the 'Trampoline Jumping'

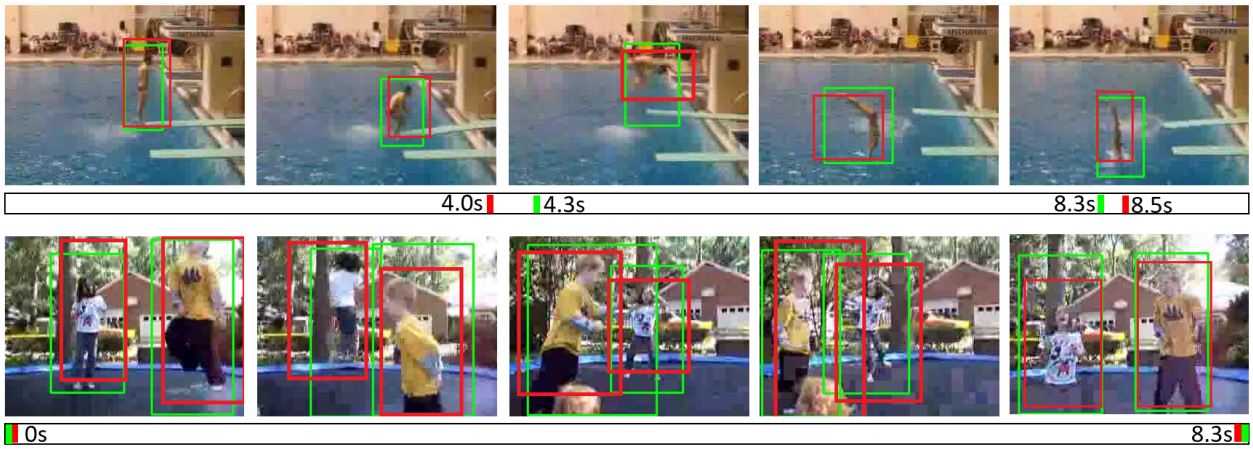


Figure 4.3: Visualizations of spatio-temporal detection of activities generated by the proposed STAD framework on sample frames from two test videos from the UCF101-24 dataset. Green and red denote ground-truth and predictions, respectively.

class over a 8.3s long video. As can be seen, our proposed method is able to detect the actor tubes while localizing the temporal bounds of the activities with higher precision. It is also obvious from these visualizations that the proposed method is capable of adapting to varying scales of the actors as well as the lengths of the activities, thanks to the multi-scale architecture incorporated in the spatial and temporal localization modules of the framework.

4.6.6 Training and Inference Time

The proposed STAD framework tries to optimize a multi-objective loss function that has several components as detailed in Sec. 4.4.1. As a result, the network takes approximately 5 days to converge while trained on 2× NVIDIA RTX 3090 GPUs. Since it is devoid of any external object detector, combined with the fact that the spatio-temporal feature maps are

shared among the spatial and temporal localization modules, our proposed method runs at a moderately higher speed, achieving a frame rate of 250 frames per second with an input size of 224×224.

4.7 Summary

In this chapter, we have presented a novel end-to-end STAD framework that is capable of performing joint optimization of spatial and temporal localization of activities from temporally untrimmed videos. Leveraging shared feature maps, and multi-scale spatial and temporal feature hierarchy, the proposed framework achieves new SOTA results on the highly challenging UCF101-24 benchmark.

Chapter 5

Human Pose for Activity Recognition

5.1 Introduction

In this chapter, our focus is on exploring how human pose information can be leveraged to better grasp and understand motion dynamics in videos. To achieve this goal, we introduce several novel representations of actor motion based on human pose information that prove to be effective for video activity recognition, all while maintaining computational efficiency. Additionally, we demonstrate the effectiveness of end-to-end learning of human pose and activity recognition altogether.

Since human activity involves the movement of whole body and/or body parts over time, accurate modeling of the motion information is at the heart of activity recognition from videos. To capture the motion information, the state-of-the-art activity recognition methods

Inference Steps	TAL-net [42]	Decouple-SSAD [55]
Optical flow extraction	204.65 ms/frame	204.65 ms/frame
RGB and Flow feature extraction	15.64 ms/frame	9.36 ms/frame
Classification and localization	0.003 ms/frame	0.002 ms/frame

Table 5.1: Breakdown of the running time of two state-of-the-art temporal activity detection methods. All times are reported on GPU.

(e.g., [3, 6, 8, 9, 42, 55, 115, 128, 129]) have primarily relied on dense optical flow due to its outstanding performance. However, the astronomical cost of computing dense optical flow makes these methods practically infeasible for real-time applications. As a reference, Table 5.1 shows a breakdown of the running time of two state-of-the-art methods for activity detection in untrimmed videos. As evident from the table, the computational bottleneck is on the optical flow extraction. This means that any method relying on dense optical flow would not be able to support real-time solutions.

We therefore explore alternative ways to model actor motion that are computationally efficient, while capable of maintaining comparable accuracy to using optical flow for video activity recognition. To this end, we tap into human pose information as a foundation for modeling actor motion. Human pose refers to the specific configuration or arrangement of the human body joints (also known as keypoints) at a given moment in time as illustrated in Fig. 5.1. We specifically leverage human 2D pose that captures the 2D spatial location of the body joints and the joint associations in an input image. This choice is particularly inspired by several key aspects of 2D pose as outlined below.

Figure 5.1: Valid configuration of human body joints/keypoints based on the COCO keypoints dataset [130]. Figure source [131].

- 2D pose can be estimated at a relatively lower computational overhead, particularly thanks to the recent advancements in real-time and highly accurate deep learning based human pose estimation methods, such as OpenPose [132], PifPaf [133] and OpenPifPaf [134].

- 2D pose is invariant to contextual nuisances, such as background variation, lighting changes, and occlusion. This property allows us to build context-agnostic solutions for activity recognition, a feat that appearance-based methods struggle to achieve.

- 2D pose focuses on body joint positions which are relatively consistent across persons. Consequently, motion representations based on such information has the

obvious advantage of being generalizable across different persons, a highly desirable attribute for any type of feature representation to be used in a learning task. This is particularly important for scenarios involving multiple users or when training data is limited.

While there has been a considerable body of work proposed in the literature focusing on utilizing human 2D pose for video activity recognition (e.g., [83–88, 93–95]), most of these methods directly feed raw 2D pose data into a subsequent processing pipeline. However, the idea of modulating raw pose data with a view to enhancing their representational capacity for modeling motion information has largely been overlooked. Our primary focus in this work is to address this gap.

The main contributions of this chapter are summarized as follows.

- We propose three novel techniques to model actor motion based on raw 2D pose data, namely, Bone Flow Map, Bone Orientation Map, and Joint Color Map.
- We investigate if, and to what extent, pose-based motion representation of actors can facilitate video activity recognition.
- We evaluate the effectiveness of the proposed methods and demonstrate their efficacy for human activity recognition, both in accuracy and computational efficiency.

5.2 Methodology

The main objective of this chapter is to propose novel representations of actor motion based on human 2D pose information and demonstrate how these motion representations can be leveraged for human activity recognition in videos, serving as robust alternatives to optical flow. To achieve this, we rely on human body joint locations and joint associations as output by the 2D pose estimation methods. To this end, we pursue the processing pipeline as outlined below.

1. Extract human 2D pose to obtain joint location confidence maps and joint association maps.

2. Develop novel motion representation of actors based on the joint confidence maps and the joint association maps.

3. Leverage the proposed motion representations to perform activity recognition from videos.

In the following subsections, we elaborate on each stage of this pipeline by discussing its components, algorithms, and implementation details.

5.2.1 Extract Human 2D Pose

Human 2D pose extraction involves estimating the spatial locations of the body joints from the input image and associating the joints according to a valid skeleton schema, such as the one shown in Fig. 5.1. The remarkable success of the CNN-based 2D pose estimation methods (e.g., OpenPose [132], PifPaf [133], OpenPifPaf [134]), along with advancements in large and diverse 2D pose estimation datasets (e.g., COCO [130], MPII [135]), coupled with the impressive strides in GPU hardware, have enabled the extraction of rich and accurate 2D pose information in real-time utilizing readily available pre-trained pose estimation tools. In this work, we therefore, take advantage of the state-of-the-art 2D pose estimator called OpenPose [132] to extract 2D pose information from the input video frames. OpenPose is selected due it's real-time speed and superior performance in multi-person pose estimation as reported in the literature.

As the motion representations proposed in this work build upon intermediate outputs generated by the pose estimator, we will provide a brief overview of the pose estimation technique employed by OpenPose in the upcoming sections. For additional details about OpenPose, please refer to the original work proposed in [132].

5.2.1.1 OpenPose Working Pipeline

The overall pose estimation pipeline of OpenPose is depicted in Fig. 5.2. OpenPose employs a deep CNN to process an input image with height H and width W in order to jointly predict

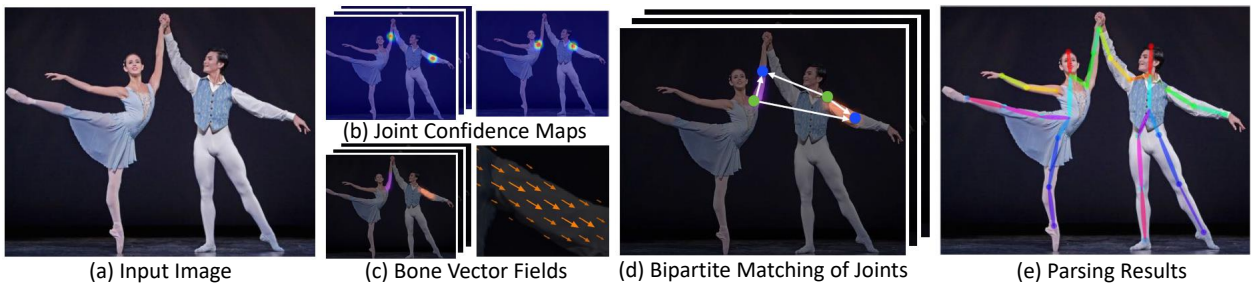


Figure 5.2: OpenPose working pipeline. (a) The input image is processed by a CNN that jointly predicts – (b) body joint confidence maps, and (c) bone vector fields for joint association. (d) Parsing of multi-person full-body pose is achieved via a set of bipartite matchings to associate the candidate body joints. (e) Parsing results. Figure source [132].

two different outputs as mentioned below.

- A set $\mathbf{S} = \{\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_J\}$ consisting of J number of confidence maps, one for each body joint type $j \in \{1...J\}$, where each confidence map $\mathbf{S}_j \in \mathbb{R}^{W \times H}$ stores the probability of each spatial location of the input image to be on the particular joint j (Fig. 5.2b).

- A set $\mathbf{L} = \{\mathbf{L}_1, \mathbf{L}_2, ..., \mathbf{L}_B\}$ consisting of B number of vector fields one for each bone type $b \in \{1...B\}$, where each spatial location in a vector field $\mathbf{L}_b \in \mathbb{R}^{W \times H \times 2}$ that is in the area belonging to the particular bone b stores a 2D vector encoding the direction pointing from the root joint to the end joint of b (Fig. 5.2c). These vector fields are later used for association of the different joints, and is referred to as part affinity fields. The connectivity and the root and end joints for each bone is defined according to a particular skeleton schema (e.g., Fig 5.1).

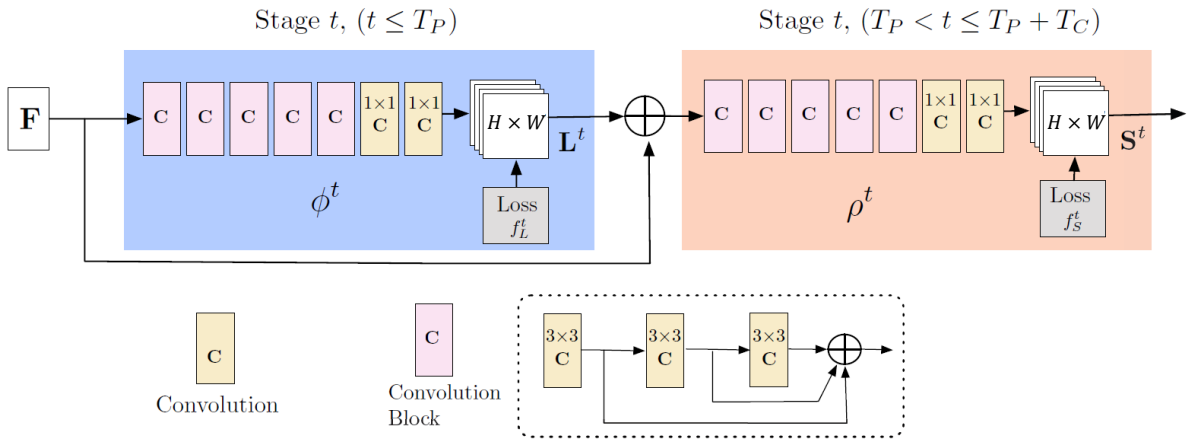


Figure 5.3: Network Architecture of the multi-stage CNN employed by OpenPose. The block marked in blue shows one of the T_p number of stages used to predict the vector fields for joint association. The block marked in beige, which employs T_c number of stages, consumes the output of the last stage from the other block in order to predict the joint confidence maps. Figure source [132].

5.2.1.2 OpenPose Network Architecture

The architecture of the multi-stage CNN employed by OpenPose is depicted in Fig. 5.3. The block marked in blue is one among the T_p number of stages that predict the vector fields for joint association in an iterative fashion. The first stage receives a set of feature maps $\mathbf{F}$ produced from the input image by a CNN (initialized by the first 10 layers of VGG-19 [50] with fine-tuning) and generates a set of vector fields $\mathbf{L}^1$. Each subsequent stage concatenates $\mathbf{F}$ with the predictions from the previous stage and uses this combined features to fine-tune the predictions,

$$\mathbf{L}^t = \phi^t(\mathbf{F}, \mathbf{L}^{t-1}), \qquad \forall 2 \leq t \leq T_p, \tag{5.1}$$

where ϕ^t refers to the CNNs for inference at stage t.

In a similar vein, the block marked in beige employs T_c number of stages to predict and fine-tune the joint confidence maps, only after the first T_p iterations for predicting the vector fields are finished.

$$\mathbf{S}^{T_p} = \rho^t(\mathbf{F}, \mathbf{L}^{T_p}), \qquad \forall t = T_p, \tag{5.2}$$

$$\mathbf{S}^t = \rho^t(\mathbf{F}, \mathbf{L}^{T_p}, \mathbf{S}^{t-1}), \qquad \forall T_p < t \leq T_p + T_c, \tag{5.3}$$

5.2.1.3 OpenPose Multi-person Pose Parsing

In order to parse the full-body pose of all persons present in the input image, the candidate body joints need to be associated with each other following a valid skeleton schema (e.g., Fig. 5.1). However, for each joint type, there may be several candidates, due to multiple persons present in the input image and/or false positives. To obtain the optimal association, OpenPose first performs non-maximum suppression on the set $\mathbf{S}$ of joint confidence maps. This results in a discrete set $\mathcal{D}_\mathcal{J}$ of joint detection candidates,

$$\mathcal{D}_\mathcal{J} = \{\mathbf{d}_j^m \ : \ \text{for} \ \ j \in \{1...J\}, \ \ m \in \{1...N_J\}\}, \tag{5.4}$$

where $\mathbf{d}_j^m \in \mathbb{R}^2$ is the location of the m-th detection candidate for joint type j, and N_J is the total number of detection candidates for joint type j.

Figure 5.4: OpenPose joint association strategy between a pair of candidate joints (red and blue dots). (a) All possible associations (grey lines). (b) Two sets of viable associations – valid associations (black lines) and invalid associations (green lines). (c) Final association results achieved using maximum weight bipartite graph matching [136] with the corresponding vector field (yellow arrows) used to generate the association weights. The vector fields help eliminate invalid associations by encoding both position and orientation information over the support of the bone.

However, the problem of finding the optimal valid association between the candidate body joints in the set $\mathcal{D}_{\mathcal{J}}$ corresponds to a K-dimensional matching problem and is known to be NP-hard [136]. Therefore, OpenPose applies some greedy relaxations to decompose the original K-partite graph matching problem into a set of bipartite graph matching subproblems. With this new problem formulation, OpenPose leverages maximum weight bipartite graph matching [136] to find the optimal association between a *single* pair of candidate body joints j_1 and j_2 connecting the b-th bone according to the valid skeleton schema, with the corresponding vector field $\mathbf{L}_b$ used to generate the association weights.

This process is illustrated in Fig 5.4.

Under this problem setting, the nodes in the bipartite graph are the body joint candidates in the two sets $\mathcal{D}_{j_1} \in \mathcal{D}_{\mathcal{J}}$ and $\mathcal{D}_{j_2} \in \mathcal{D}_{\mathcal{J}}$, while the edges are all possible connections between the nodes from the two sets (Fig. 5.4a). Moreover, the weight of an edge is defined by the integral over the vector field $\mathbf{L}_b$ along the line segment connecting the joints j_1 and j_2 as below.

$$E = \int_{u=0}^{u=1} \mathbf{L}_b(p(u)) \frac{\mathbf{d}_{j_2} - \mathbf{d}_{j_1}}{||\mathbf{d}_{j_2} - \mathbf{d}_{j_1}||^2} du, \tag{5.5}$$

where $p(u)$ interpolates the position $\mathbf{d}_{j_1}$ and $\mathbf{d}_{j_2}$ of the two body joints. With this, the optimal association is obtained by finding a matching with maximum weight for the chosen edges as follows.

$$E_b = \max_{\mathcal{Z}_b} \sum_{m \in \mathcal{D}_{j_1}} \sum_{n \in \mathcal{D}_{j_2}} E_{mn} \, z_{j_1 j_2}^{mn}, \tag{5.6}$$

$$\text{s.t.} \qquad \forall m \in \mathcal{D}_{j_1}, \sum_{n \in \mathcal{D}_{j_2}} z_{j_1 j_2}^{mn} \leq 1, \tag{5.7}$$

$$\forall n \in \mathcal{D}_{j_2}, \sum_{m \in \mathcal{D}_{j_2}} z_{j_1 j_2}^{mn} \leq 1, \tag{5.8}$$

where, E_b is the overall weight of the matching for bone type b, $\mathcal{Z}_b$ is the subset of $\mathcal{Z}$ for bone type b, and E_{mn} is the association weight (i.e., edge weight) for connecting the m-th detection candidate joint of type j_1 with the n-th detection candidate joint of type j_2 as defined in Eq. 5.5. The set $\mathcal{Z}$, on the other hand, is a collection of binary indicator variables

$\mathcal{Z} = \{z_{j_1 j_2}^{mn} : \text{ for } \ j_1, j_2 \in \{1...J\}, m \in \{1...N_{j_1}\}, n \in \{1...N_{j_2}\}$, where each indicator variable $z_{j_1 j_2}^{mn} \in \{0, 1\}$ indicates if the m-th detection candidate joint of type j_1 is connected to the n-th detection candidate joint of type j_2 or not.

5.2.2 Motion Representations using 2D Pose

Leveraging the pose information available from the off-the-shelf 2D pose estimator, such as OpenPose, we propose three different representations of actor motion in this work that are amenable to action recognition from videos.

1) Motion representation based on bone movement

2) Motion representation based on bone orientation

3) Motion representation based on joint confidence maps

5.2.2.1 Motion Representation based on Bone Movement

Inspired by optical flow, in this work we propose a motion representation that is based on the movement of the individual bones belonging to the person skeleton available from the full-body pose estimation results. We refer to this representation as *Bone Flow*. The primary motivations behind motion representation using *Bone Flow* as opposed to optical flow, in the context of activity recognition from videos, can be summarized as follows.

1. *Bone Flow* is actor-centric, thereby allowing it to effectively suppress background

movement and significantly reduce video background noise. In contrast, dense optical flow encodes the movement of all objects in the scene.

2. *Bone Flow* has the potential for higher computational efficiency when compared to dense optical flow. This efficiency arises because *Bone Flow* can be applied selectively to pixels in the vicinity of the detected persons in the frame, without requiring any processing for frames lacking persons. In contrast, as mentioned in the Introduction, accurate and dense optical flow is computationally intensive, partly because it processes every pixels.

For the purpose of easy explanation, we first describe the *Bone Flow* algorithm with reference to a single bone in Sec. 5.2.2.1.1, while Sec. 5.2.2.1.2 elaborates how the algorithm can be scaled to all bones in a person skeleton across multiple persons throughout a whole video sequence.

5.2.2.1.1 *Bone Flow* Algorithm The *Bone Flow* algorithm is illustrated in Fig. 5.5 with respect to an arbitrary bone and arbitrary pixel, while the operation of the algorithm appears in Algorithm 1. The core idea behind the algorithm is to compute the bone displacement vectors (denoted by 'delta_vec' in Fig. 5.5) for each bone as it moves over time. This is achieved by finding a mapping of pixels surrounding the bone between consecutive frames. The input to the algorithm is the location of the root and end joints of the bone over two time steps (i.e., over a two-frame sequence), the dimensions of the image,

a threshold distance D_{th} such that pixels within this distance from the bone are considered, as well as a global scaling parameter *'scale'* to scale the length of the bone. The algorithm outputs a vector field $V \in \mathbb{R}^{H \times W \times 2}$ storing the bone displacement vectors at each spatial location for the particular bone.

The steps of the algorithm are outlined below.

i) First, the distance from all pixels belonging to the frame at the current time step $t = 2$ to the nearest point on the bone (i.e., the bone line or the end points of the bone) is computed (line number 12 and elaborated in line numbers $20 - 32$).

ii) For each pixel *(i, j)* within a distance D_{th} from the bone at time step $t = 2$, a change of basis is performed to find the coordinates of the pixel with respect to the unit bone vector and the vector perpendicular to the unit bone vector (line numbers $14 - 15$).

iii) Then, these new coordinates (i.e., *(u, v)*) are used to find a corresponding pixel with respect to the same bone on the frame at a previous time step $t = 1$ (line number 16).

iv) Finally, the displacement vector between the corresponding pixels on the two frames are computed (line number 17).

v) After computing the bone displacement vector at pixel location *(i, j)*, it is gated through the bone confidence at that location. To this end, each component of the vector is multiplied by the confidence of the bone at that pixel, where the confidence is computed by passing the distance from that pixel location to the bone (step ii above) through a

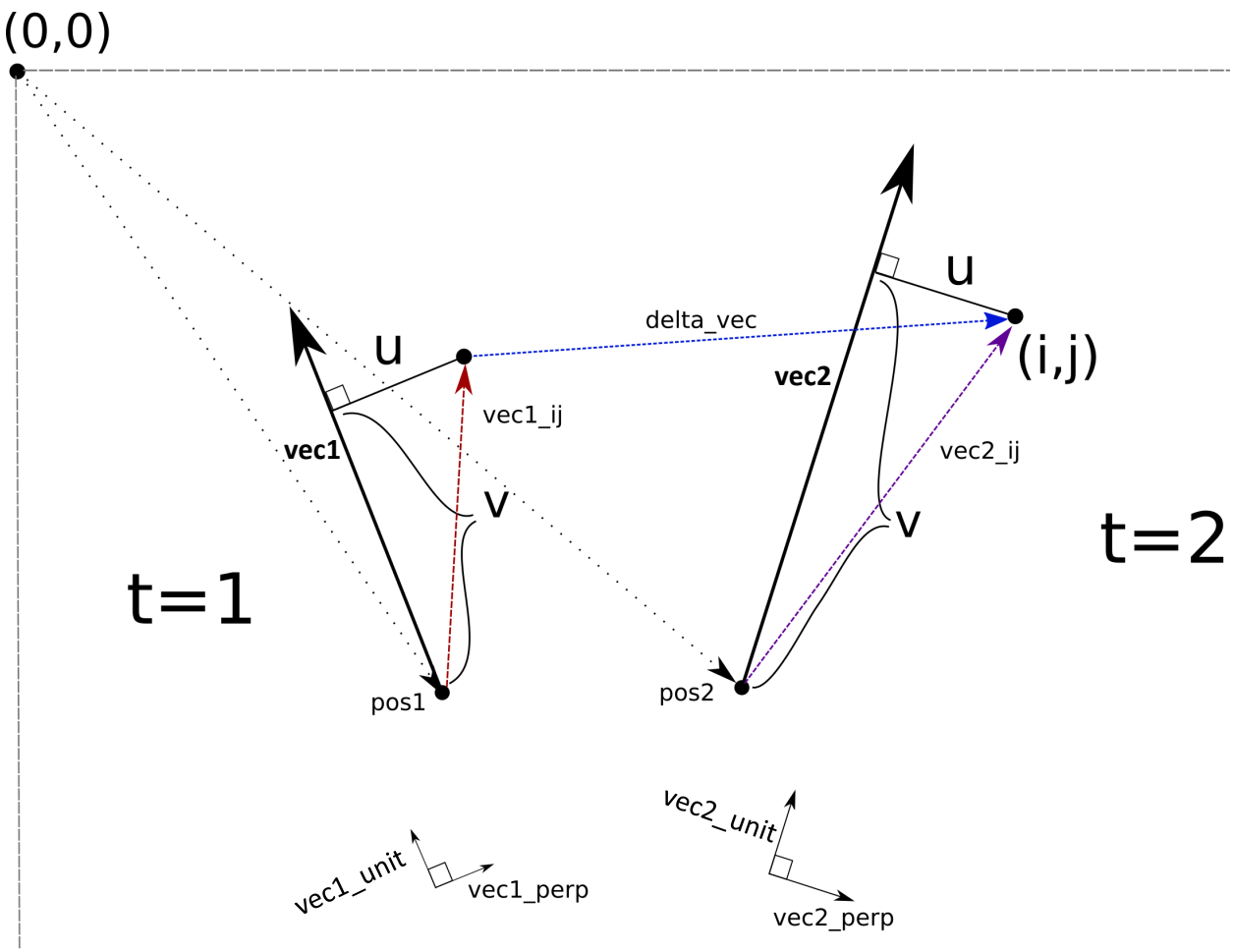


Figure 5.5: Illustration of the proposed *Bone Flow* algorithm with respect to an arbitrary bone and arbitrary pixel. Top-left corner *(0, 0)* is the origin of the image coordinate. The position of the bone root joint, bone vector, unit bone vector and the vector perpendicular to the unit bone vector in time step $t = 1$ ($t = 2$) are referred to by *pos1* (*pos2*), *vec1* (*vec2*), *vec1_unit* (*vec2_unit*), and *vec1_perp* (*vec2_perp*), respectively. In the bone vector space, *(u, v)* represents the coordinates of the pixel *(i, j)* at $t = 2$, and is used to find the coordinates of the corresponding pixel at $t = 1$. *delta_vec* represents the bone displacement vector between the two time steps.

Algorithm 1: *Bone Flow* Algorithm

Input : *root1, root2, end1, end2* : location of the bone root and end joints in image coordinates at time step $t = 1$, and $t = 2$, respectively; H, W : height, width of the image; D_{th} : distance threshold, pixels within which from the bone is to be considered for processing; *scale* : a global parameter to scale the length of the bone

Output: $V \in \mathbb{R}^{H \times W \times 2}$: vector field storing a displacement vector at each spatial location

vec1 $\leftarrow$ *end1* $-$ *root1*
vec2 $\leftarrow$ *end2* $-$ *root2*
vec1_unit $\leftarrow$ **vec1**$/\|$**vec2**$\|_2$
vec2_unit $\leftarrow$ **vec2**$/\|$**vec2**$\|_2$
vec1_perp $\leftarrow$ Compute vector perpendicular to **vec1_unit**
vec2_perp $\leftarrow$ Compute vector perpendicular to **vec2_unit**
A1 $\leftarrow \begin{bmatrix} \textbf{vec1\_unit} \\ \textbf{vec1\_perp} \end{bmatrix}$ `// change of basis matrix for` $t = 1$
A2 $\leftarrow \begin{bmatrix} \textbf{vec2\_unit} \\ \textbf{vec2\_perp} \end{bmatrix}$ `// change of basis matrix for` $t = 2$
V $\leftarrow 0$ `// initialize vector field`
for $i \leftarrow 1$ **to** W **do**
  for $j \leftarrow 1$ **to** H **do**
    dist $\leftarrow$ `ComputeDistance(`*(i,j), root2,* ***vec2, vec2_unit***`)`
    if *dist* $\leq D_{th}$ **then**
      vec2_ij $\leftarrow$ (i, j) - *root2*
      vec_uv $\leftarrow$ A2$^{-1}\times$ **vec2_ij**
      vec1_ij $\leftarrow$ A1 $\times$ **vec_uv**
      delta_vec $\leftarrow$ (*root2* + **vec2_ij**) - (*root1* + **vec1_ij**)
      $\sigma \leftarrow$ *scale* $*$ $\|$**vec2**$\|_2$
      V$[i, j] \leftarrow$ **delta_vec** $* \exp(-\frac{dist^2}{2\sigma^2})$ `// gating delta vector by bone confidence`

Function `ComputeDistance(`***point****, root,* ***bone****,* ***bone_unit***`):`
  bone_point_dist $\leftarrow 0$
  root_point_vec $\leftarrow$ **point** - *root*
  end_point_vec $\leftarrow$ **root_point_vec** - **bone**
  projection $\leftarrow$ **bone_unit** $\cdot$ **root_point_vec** `// dot product`
  bone_point_vec $\leftarrow$ **root_point_vec** - **bone_unit** $*$ *projection*
  if *projection* < 0 **then**
    bone_point_dist $\leftarrow \|$**root_point_vec**$\|_2$
  else if *projection* $> \|$***bone***$\|_2$ **then**
    bone_point_dist $\leftarrow \|$**end_point_vec**$\|_2$
  else
    bone_point_dist $\leftarrow \|$**bon_point_vec**$\|_2$
  return *bone_point_dist*

Gaussian function, with the length of the bone (scaled by a factor called *scale*) used as the width of the Gaussian (line number $18 - 19$).

Bone Flow Map: The bone displacement vector field $V \in \mathbb{R}^{H \times W \times 2}$ as computed above is eventually encoded into a 3-channel RGB image $M \in \mathbb{R}^{H \times W \times 3}$ which we refer to as Bone Flow Map. To this end, we first rescale the absolute values of the vector field into the range $[0 - 255]$, followed by assigning the X and Y components to R and G channels respectively, while setting the B channel to 0. Equation 5.9 shows the encoding process, while Figure 5.6 shows the results of the encoding for an arbitrary bone displacement vector field.

$$M_{i,j,0:2} = 255 * (max(V) - |V_{i,j,:}|)/(max(V) - min(V)), \quad \forall i \in \{1...H\}, j \in \{1...W\}, \quad (5.9)$$

5.2.2.1.2 Computing Bone Flow Maps for Multiple Persons The *Bone Flow* algorithm can be further leveraged to compute the Bone Flow Maps for multiple persons in a video sequence. The basic idea is to compute the bone displacement vector fields corresponding to each person by averaging over the vector fields for all bones visible for that particular person, followed by computing the max-normed vector at each spatial location across the vector fields for all persons.

The operations involved in the process is presented in Algorithm 2. The input to the algorithm is a tensor storing the body joint locations of all persons visible in each frame in a video sequence, which can be made available by an off-the-shelf pose estimator (e.g.,

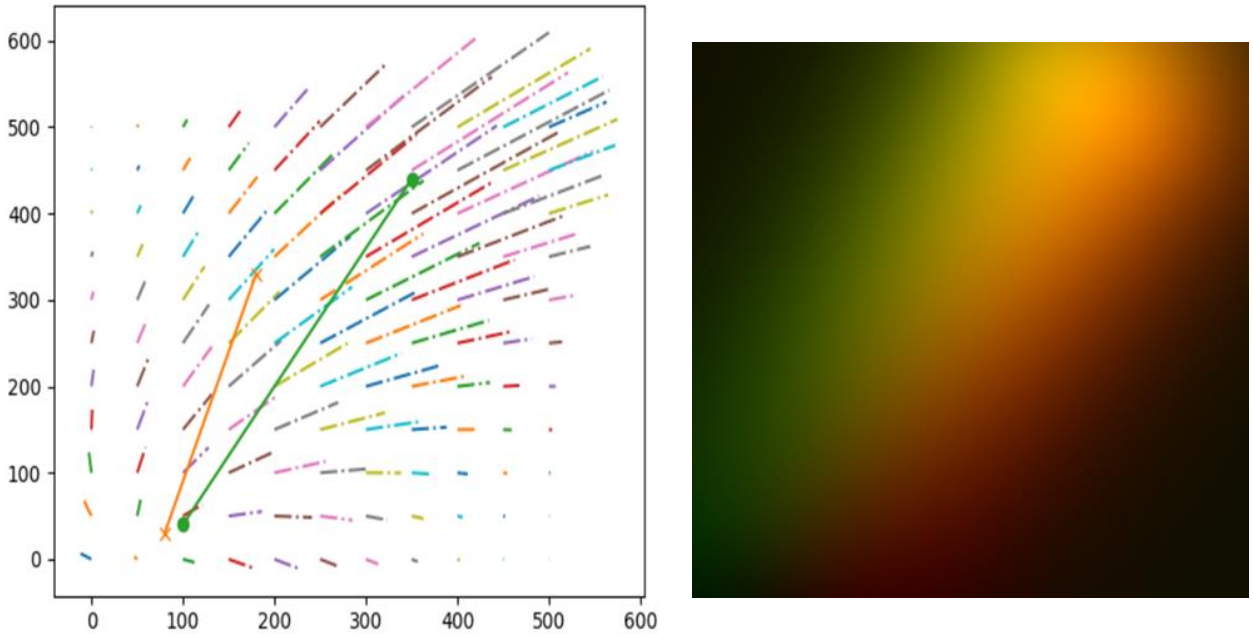


Figure 5.6: (left) Vector field showing the bone displacement vectors for an arbitrary bone, with the green and orange lines representing the bone vector at time step $t = 2$ and $t = 1$, respectively. (right) 3-channel RGB image encoding of the vector field, with R and G channels corresponding to X and Y components of the vector field respectively, and B set to 0. The horizontal and vertical movements of the bone can be inferred by the red and green colors in the image.

OpenPose) as described in Sec. 5.2.1. Besides, the height and width of the video frames, and a threshold time gap $t_{\max}$ from the current time step within which to consider generating Bone Flow Maps are also input to the algorithm. The output of the algorithm are the Bone Flow Maps corresponding to each video frame. The steps of the algorithm are outlined below.

i) For each time step t, and for each person visible on t, the bone displacement vector

Algorithm 2: Computing Bone Flow Maps for Multiple Persons in a Video

Input : $K \in \mathbb{Z}^{T\times P\times J\times 2}$: a tensor storing the joint locations of J body joints for a maximum of P persons present in each of the T frames in an input video; S : a person skeleton schema consisting of B bones, where each bone is defined by the connectivity between the root and end joints of the bone; H, W : height, width of the video frames; $t_{\max}$: threshold time gap

Output: $M \in \mathbb{R}^{T\times H\times W\times 3}$: Bone Flow Maps for the T frames in the video

1 $\mathbf{M} \in \mathbb{R}^{T\times H\times W\times 3} \leftarrow 0$ `// initialize Bone Flow Maps for all frames`
2 $\mathbf{V} \in \mathbb{R}^{T\times H\times W\times 2} \leftarrow 0$ `// initialize vector field for all frames`
3 **for** $t \leftarrow 1$ **to** T **do**
4 $N \in \mathbb{R}^{P\times H\times W\times 2} \leftarrow 0$ `// initialize vector field for all persons`
5 **for** $p \leftarrow 1$ **to** P **do**
6 $O \in \mathbb{R}^{B\times H\times W\times 2} \leftarrow 0$ `// initialize vector field for all bones`
7 **for** $b \leftarrow 1$ **to** B **do**
8 **if** *b is visible on current time step t* **then**
9 $r, e \leftarrow$ *get root and end joint ids of b using schema S*
10 $root2 \leftarrow K_{t,p,r}$
11 $end2 \leftarrow K_{t,p,e}$
12 **if** *b was visible on a prior time step q; s.t.* $(t - t_{max}) \leq q < t$ **then**
13 $root1 \leftarrow K_{q,p,r}$
14 $end1 \leftarrow K_{q,p,e}$
15 $O[b] \leftarrow$ *Compute vector filed for b using* ***Algorithm 1***
16 *root1, end1* $\leftarrow$ *root2, end2* `// update bone status`
17 $N[p] \leftarrow \underset{b\in\{1\ldots B\}}{\text{avg}}(O[b])$ `// average across all bones`
18 $V[t] \leftarrow \underset{p\in\{1\ldots P\}}{\text{max-normed-vector}}(N[p])$ `// max-normed vector across all persons`
19 **for** $t \leftarrow 1$ **to** T **do**
20 $M[t, :, :, 0:2] \leftarrow 255 * (\max(|V|) - |V[t]|)/(\max(|V|) - \min(|V|))$

field corresponding to each bone belonging to a particular person is computed using Algorithm 1, provided that the bone is visible on t as well as on any prior time step within $t_{\max}$ distance from t (line numbers $8 - 16$).

ii) These vector fields are then averaged to produce a single vector field representing the bone movements of the particular person on time step t (line number 17).

iii) The vector fields for all the persons visible on t, as obtained from step ii), are aggregated by taking the max-normed vector at each spatial location across all such vector fields. The resultant vector field represents the bone movement of all persons present in the frame on time step t (line number 18).

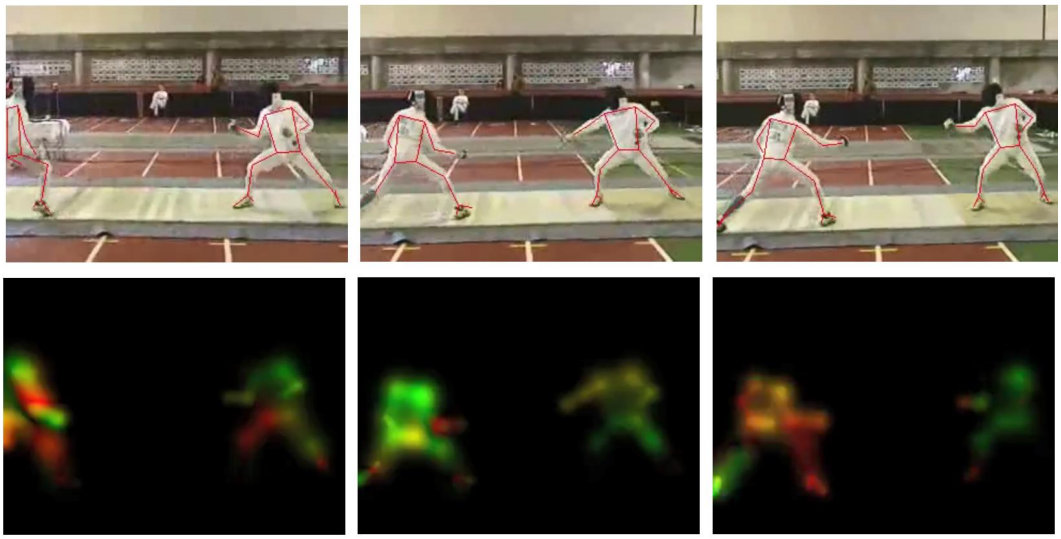
iv) Finally, the Bone Flow Maps for each time step are computed by encoding the corresponding vector fields into a 3-channel RGB image representation using Eq. 5.9 (line numbers $19 - 20$).

Figure 5.7 shows the Bone Flow Maps for some example video frames from the UCF101 dataset [107]. The horizontal and vertical movements of the bones are encoded in the 'R' and 'G' channels of the image respectively, and as such, can be inferred by red and green colors. As evident from the figure, the Bone Flow Map representation can reliably capture actor motion information for both single-person and multi-person activities, while effectively suppressing unnecessary background clutter, as seen in the original RGB frames.

5.2.2.2 Motion Representation based on Bone Orientation

The Bone Flow Map representation, as described in the previous section, relies on the person skeleton information output from a 2D pose estimator and attempts to capture actor motion

(a) Bone Flow Maps for a single-person activity



(b) Bone Flow Maps for a multi-person activity

Figure 5.7: Bone Flow Maps for example frames from the UCF101 dataset showing (a) a single-person activity, and (b) a multi-person activity. For each example, the top row shows the RGB frames with the person skeleton superimposed, while the bottom row shows the corresponding Bone Flow Maps.

by modeling their bone movements over time. However, we can alternatively model actor motion based on bone orientation information available from a 2D pose estimator, such as OpenPose.

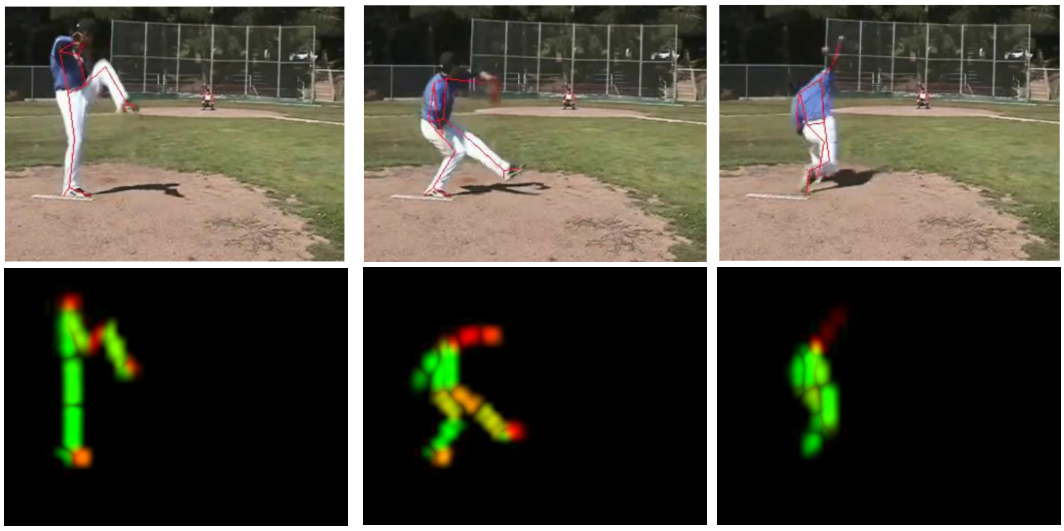
To this end, we directly tap into the joint association vector fields $\mathbf{L} = \{\mathbf{L}_1, \mathbf{L}_2, ..., \mathbf{L}_B\}$ that are available as intermediate output from OpenPose as described in Sec. 5.2.1.1. To quickly recap, given an input image, a joint association vector field $\mathbf{L}_b \in \mathbb{R}^{W \times H \times 2}$ stores a 2D vector at each spatial location encoding the direction pointing from the root joint to the end joint of a particular bone $b \in \{1...B\}$ for all persons visible in the image, where B is the total number of bones (refer to Fig. 5.2c).

We first combine the B vector fields into one by taking the max-normed vector across all bone vectors at each spatial location as follows.

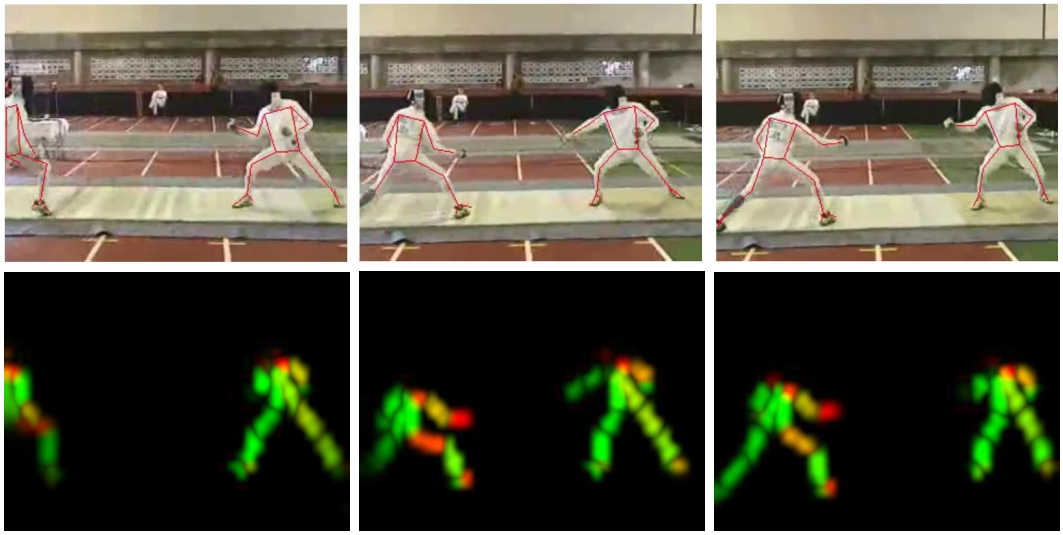
$$\mathbf{L}_{\max} \in \mathbb{R}^{W \times H \times 2} = \underset{b \in \{1...B\}}{\text{max-normed-vector}}\{\mathbf{L}_b\} \tag{5.10}$$

Similar to Bone Flow Map, we then encode $\mathbf{L}_{\max}$ into a 3-channel RGB image using Eq. 5.9, with the X and Y components of the vector field encoded into 'R' and 'G' channels respectively, and the 'B' channel set to 0. We refer to this representation as Bone Orientation Map.

Figure 5.8 shows the Bone Orientation Maps for some example video frames from the UCF101 dataset [107]. This representation, as evident from the figure, mainly encapsulates



(a) Bone Orientation Maps for a single-person activity



(b) Bone Orientation Maps for a multi-person activity

Figure 5.8: Bone Orientation Maps for example frames from the UCF101 dataset showing (a) a single-person activity, and (b) a multi-person activity. For each example, the top row shows the RGB frames with the person skeleton superimposed, while the bottom row shows the corresponding Bone Orientation Maps.

the person skeleton information while highlighting the bone regions more prominently than the Bone Flow Map representation as well as person skeleton.

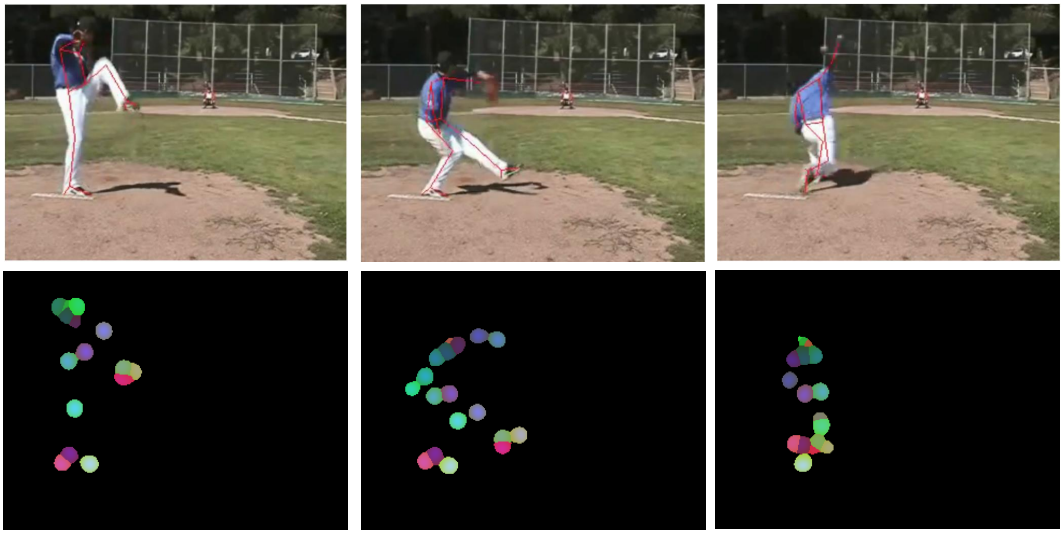
5.2.2.3 Motion Representation based on Joint Confidence Map

We also leverage the joint confidence maps $\mathbf{S} = \{\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_J\}$ that are available as an intermediate output from OpenPose (refer to Sec. 5.2.1.1) to build a motion representation. As a quick recap, given an input image, a joint confidence map $\mathbf{S}_j \in \mathbb{R}^{W \times H}$ stores the probability of each spatial location in the image to be on a particular joint $j \in \{1...J\}$, where J is the total number of joints. The basic idea behind this motion representation is to generate a 3-channel pose embedding that encodes the joint type with 2 channels (e.g., 'R' and 'G'), and the joint location confidence with 1 channel (e.g., 'B'). We refer to this representation as Joint Color Map.

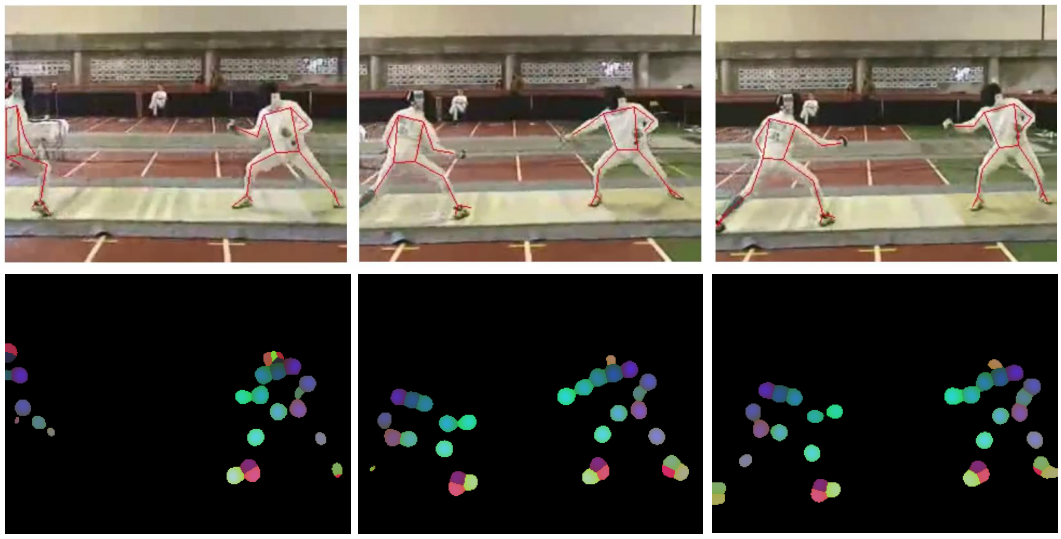
To this end, the joint type with the highest confidence across all joints at each spatial location is computed as follows.

$$\mathbf{S}_{\max} \in \mathbb{R}^{W \times H} = \underset{j \in \{1...J\}}{\operatorname{argmax}} \{\mathbf{S}_j\}, \tag{5.11}$$

Finally, the 3-channel pose embedding is achieved by encoding the joint type at each spatial location of $\mathbf{S}_{\max}$ with 'R' and 'G' channels, and the joint confidence by the 'B' channel. To encode the joint type using two channels, we create a maximally separated grid over 'R' and



(a) Joint Color Maps for a single-person activity



(b) Joint Color Maps for a multi-person activity

Figure 5.9: Joint Color Maps for example frames from the UCF101 dataset showing (a) a single-person activity, and (b) a multi-person activity. For each example, the top row shows the RGB frames with the person skeleton superimposed, while the bottom row shows the corresponding Joint Color Maps. Joint types are represented by the different colors.

'G' dimensions for J number of joints as shown below.

$$\text{RG\_codes} = \{(i * \text{color\_step}, j * \text{color\_step}) \quad : \quad \forall i, j \in \{1...\sqrt{J}\}\}, \tag{5.12}$$

$$\text{color\_step} = 255/(\sqrt{J} + 1), \tag{5.13}$$

Figure 5.9 shows the Joint Color Map representations for some example video frames from the UCF101 dataset [107]. Unlike the two other representations presented above, the Joint Color Map representation leverages the colors to distinguish the different joints.

5.2.2.4 Relative Comparison of the Different Motion Representations

A relative comparison among the different motion representations is presented below based on the following properties.

i) **Ability to capture person motion:** The Bone Flow Map representation directly encodes the body movement of the persons, whereas, the Bone Orientation Map and the Joint Color Map representations are more focused on enhancing person skeleton information by encoding bone orientation and joint location, respectively.

ii) **Ability to capture person structure:** Bone Flow Map captures a more prominent person structure compared to the other two representations. The Bone Orientation Map mainly highlights the region of support over the bones, while the Joint Color Map marks only the joint regions without conveying person structure information. This can

be verified from the respective example representations as depicted in Fig. 5.7, 5.8, and 5.9.

iii) **Intensity of visual signature:** Since the Bone Flow Map representation is centered around bone movement, it may lack a distinct visual signature in the image representation in case there's no or limited person movement between consecutive frames. In contrast, the bone orientation map and Joint Color Map representations are based on the presence of bones and joints, respectively. Thus, regardless of body movement, these representations will consistently exhibit a strong visual signature in the resulting images.

iv) **Sensitivity to occlusion:** Bone Flow Map suffers from self occlusion (e.g., bone-bone overlaps) and person-person occlusion, because the bone flow vectors spread over neighboring regions. This is particularly true in crowded scenes with overlapping persons, leading to a loss of pose information. On the other hand, Bone Orientation Map is less susceptible to occlusion since it is based on vectors covering the region of support of the bones. As expected, the Joint Color Map representation is the least affected by occlusion because joints, especially those of the same type, are much less likely to occlude each other compared to bones.

5.2.3 Activity Recognition using Proposed Motion Representations

In the preceding sub-sections, we have introduced three novel motion representations based on human 2D pose information. Referring back to the processing pipeline outlined in Sec. 5.2, we now discuss how these motion representations can be leveraged to perform activity recognition from videos. To this end, we explore two different approaches, i) feed the static motion representations as input to an activity recognition module, and ii) perform end-to-end learning of pose and activity recognition.

5.2.3.1 Fixed Motion Representation

The various motion representations proposed are all encoded as 3-channel RGB images. This encoding choice enables us to directly feed these motion representations as input to the state-of-the-art video activity recognition models (e.g., [5–8, 137]). Specifically, we use the S-3DG video activity recognition network [8] as employed for feature extraction for the TAD and STAD tasks presented in Chap. 3 and 4, respectively. Since S-3DG employs 3D convolutions to process a video, we feed T number of sequential frames with dimensions $H \times W \times 3$ generated by the different motion representations (i.e., Bone Flow Map/Bone Orientation Map/Joint Color Map) from the video as input to S-3DG. The output of S-3DG is a C-dimensional vector representing the class probabilities of the input over C classes.

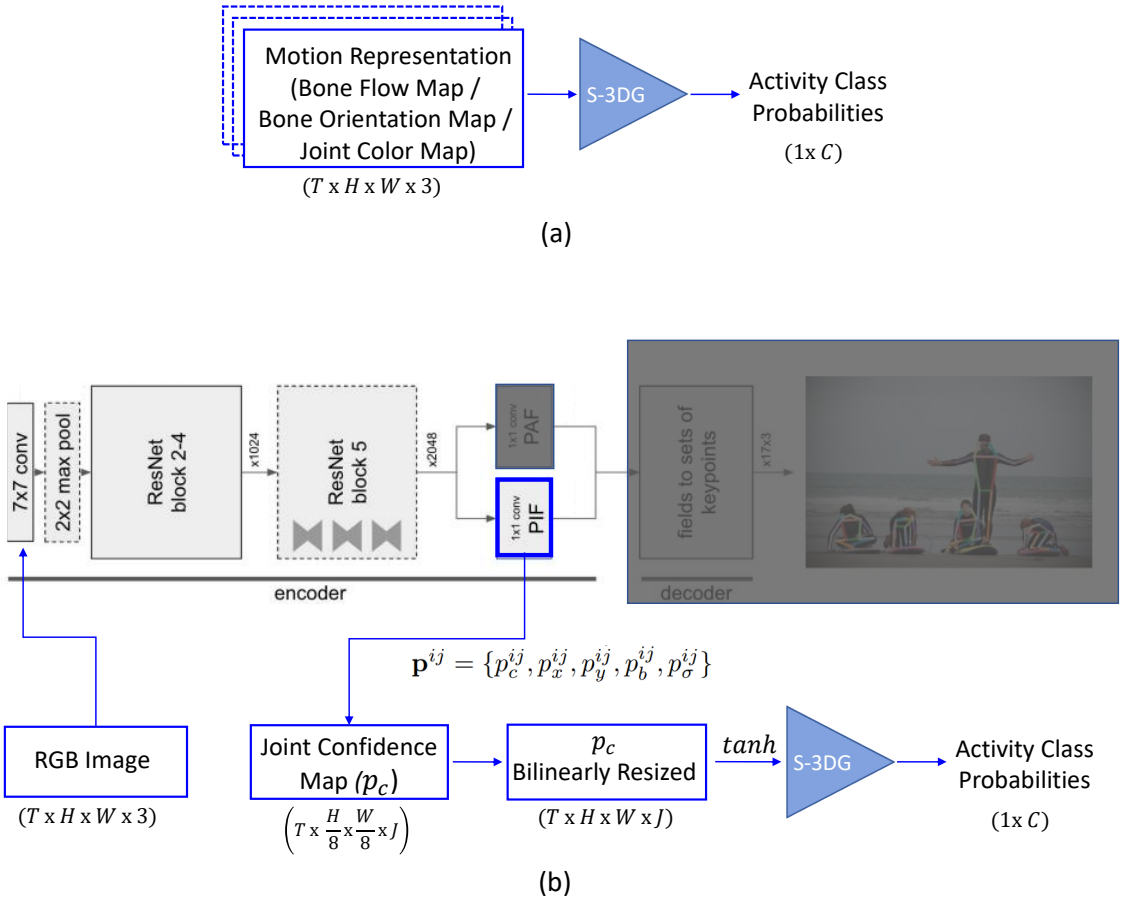


Figure 5.10: (a) Activity recognition using the proposed motion representations. (b) End-to-End learning of pose and activity recognition. The decoder and the PAF modules of the PifPaf network are not used in the end-to-end learning process, hence shaded in dark colors.

Figure 5.10(a) shows this pipeline. For additional details and internal architecture of the S-3DG network, please refer to the original work proposed in [8].

5.2.3.2 End-to-End Learning

We also investigate the concept of end-to-end learning for pose and activity recognition, while being guided by the sole objective of activity recognition. To be specific, we learn

the activity recognition network S-3DG along with a pose estimator whose output is fed as an input to S-3DG. In this approach, we utilize the J-channel joint confidence maps (e.g., $\mathbf{S} \in \mathbb{R}^{H \times W \times J}$) which are available as intermediate outputs from a 2D pose estimator, rather than our initial 3-channel motion representations. The reason for this shift is that our motion representations are non-differentiable, making them unsuitable for end-to-end training. Furthermore, we opt for a different pose estimator, namely PifPaf [133], as an alternative to OpenPose. This choice is made to facilitate easy and seamless integration of the pose network with S-3DG network.

Figure 5.10(b) presents this pipeline. As shown in the figure, the PifPaf network comprises an encoder and a decoder. The encoder is based on the ResNet [105] model which employs 5 blocks of covolutional layers to process an input RGB image in order to produce two intermediate outputs – Part Intensity Field (PIF) and Part Association Field (PAF). PIF and PAF are very similar in notion to joint confidence maps, and joint association vector fields, respectively as produced by OpenPose. The decoder module uses these intermediate outputs to produce the final pose estimation results. However, for the purpose of this work, we only tap into PIF, thus not propagating any gradients through the decoder and the PAF module during end-to-end learning. This is indicated by having these modules shaded in dark colors in the figure.

To elaborate, at each output location (i, j), PIF makes 5 different predictions – a joint confidence score p_c^{ij}, a vector (p_x^{ij}, p_y^{ij}) with spread p_b^{ij} pointing to the nearest joint, as well

as a scale parameter p_{σ}^{ij} for the size of the joint. However, we are only interested in the joint confidence map $p_c \in \mathbb{R}^{\frac{H}{8} \times \frac{H}{8} \times J}$, which has a spatial down-sampling factor of 8 compared to the input image. Starting with p_c, we first upsample it using bilinear interpolation to match the size of the input image (i.e., $H \times W$), then pass it through $tanh$ activation to rescale the joint confidence scores in the range $[-1, 1]$, which is finally fed as input to the S-3DG network. To accommodate the J-channel joint confidence map p_c as input, the channel dimension of S-3DG's convolutional filters on the first layer is set to J. The whole pipeline is trained end-to-end. For additional details about the PifPaf network, please refer to the original work proposed in [133].

5.3 Experimental Setup

5.3.1 Dataset

We use the UCF101 [107] video activity recognition dataset to evaluate our proposed methodology. UCF101 stands out as one of the most diverse video activity recognition datasets, containing a total of $13,320$ videos spanning over 101 different activity categories. The resolution of the videos is 320×240 pixels. It exhibits significant variability in camera movement, object appearance, posture, object size, perspective, background complexity, lighting conditions, and more.

The videos for each activity in UCF101 is organized into 25 groups, with each group

Figure 5.11: Sample frames from the 52 activity categories selected from the UCF101 dataset. The green, red, blue, and purple colors around the frames represent the activity types, namely, 'Sports', 'Body-Motion Only', 'Human-Object Interaction', and 'Human-Human Interaction', respectively.

typically containing 4−7 videos. Videos within the same group often share common attributes like similar backgrounds and viewpoints. Moreover, the activity categories can be divided into five types – i) 'Human-Object Interaction', ii) 'Body-Motion Only', iii) 'Human-Human Interaction', iv) 'Playing Musical Instruments', and v) 'Sports'. However, for the purpose of this work, we specifically choose 52 out of the 101 activity categories that emphasize human body movement and demand less reliance on visual appearance details for differentiation. Our decision aligns with the primary goal of this research, which is to demonstrate the efficacy of alternative approaches for acquiring motion information in

activity recognition, as opposed to using optical flow. This subsampling process results in a dataset comprising 6,712 videos. In this dataset, for each activity category, the initial 23 groups of videos totalling to $5,956$ are designated for the training set, while the remaining 2 groups are designated for the test set containing 756 videos. Fig. 5.11 shows one sample frame from each of the 52 categories.

5.3.2 Performance Metric

The performance of the proposed methods is evaluated based on the test set using the standard metric for the video activity recognition task called Mean Classification Accuracy, which is defined as the mean of the class-wise classification accuracies over all categories. Class-wise classification accuracy for a particular class i, on the other hand, is measured as the proportion of samples belonging to class i that are correctly classified as class i. More formally, let $Total_i$, TP_i, and C denote the total number of samples belonging to class i, number of samples that are correctly classified as class i, and total number of activity categories. Then,

$$Mean\ Classification\ Accuracy = \frac{1}{C} \sum_{i \in \{1 \ldots C\}} \frac{TP_i}{Total_i} \tag{5.14}$$

5.3.3 Implementation Details

The activity recognition network S-3DG is randomly initialized and trained using a batch size of 6 per GPU, where each sample consists of a frame sequence generated by sampling 64 frames from the input video. Details about the sampling process appears in Sec. 5.3.4.1. The whole network is trained using Adam optimizer with an initial learning rate of 0.001, which is reduced by a factor of 10 at step 50k. The PifPaf network, on the other hand, is initialized from a pre-trained model on COCO keypoints dataset [130], and trained using a much lower learning rate of 0.0001 in order not to distort the pre-trained weights too much. During training, data augmentation is performed using random consistent left-right flipping, where either all or none of the 64 frames belonging to the input frame sequence are flipped horizontally. Additionally, temporal jittering is applied to randomly choose the frame to start the sampling of the 64 frames from an input video. However, no data augmentation is applied during evaluation. Training is performed on 2× NVIDIA RTX 3090 GPUs using Tensorflow [138]. For the *Bone Flow* algorithm, we set the distance threshold D_{th} to twice the length of the bone, the scaling factor *scale*= 0.25, and the threshold time gap $t_{\max} = 1$.

5.3.4 Results

In this section, we first present the results of video activity recognition based on the proposed motion representations, followed by results achieved using end-to-end learning of

Frame Sampling	Effective Temporal Footprint (frames)	Mean Classification Accuracy (%)		
		Bone Flow Map	Bone Orientation Map	Joint Color Map
every frame	64	48.54	53.78	52.23
every 2nd frame	128	55.12	61.32	60.96
every 4th frame	**256**	**67.13**	**75.46**	**71.10**

Table 5.2: Activity recognition accuracy based on the proposed motion representations. Results are reported on the test set.

pose and activity recognition. We then compare these results with the baseline approach based on optical flow. Finally, an analysis of run-time performance for the different methods is presented at the end.

5.3.4.1 Results for the Proposed Motion Representations

Table 5.2 presents the results of activity recognition using the proposed motion representations. For each representation, we assess the influence of temporal footprint of the input frame sequence on activity recognition by training and evaluating models at varying frame sampling rates while sampling a constant number of frames per video (i.e., 64 frames). Specifically, we explore three different sampling rates: sampling every frame, sampling every other frame, and sampling every fourth frame, resulting in effective temporal footprints of 64, 128, and 256 frames for the input frame sequence, respectively. We train all layers of the activity recognition network S-3DG from scratch.

As indicated in the table, higher temporal footprint correlates with increased accuracy across all three motion representations. This phenomenon can be attributed to the fact that,

with a larger temporal footprint, the 3D CNN model S-3DG is capable of capturing a greater amount of temporal context from the entire video. However, no further further performance improvement is noticed for sampling rates higher than 4, likely because subtle temporal cues between consecutive samples are diminished with greater sampling rates, especially for activities involving fast-moving actors (e.g., LongJump, HighJump).

Among the different motion representations, the Bone Orientation Map stands out as the best performer, achieving an overall classification accuracy of 75.46%. Following closely behind is the Joint Color Map, with an overall classification accuracy of 71.10%, while the Bone Flow Map lags behind with mean accuracy of 67.13%. The primary reason for the Bone Flow Map's lower performance can be attributed to the lack of clear visual features in the resulting images when there is minimal or no movement of persons between consecutive frames. Additionally, the *Bone Flow* vectors tend to spread over neighboring regions, thereby creating confounding signals in the resulting image representations, especially in situations where bones overlap with each other. These issues have been discussed in more detail in Sec. 5.2.2.4. These challenges are not present in the other two representations. Although the Joint Color Map is able to encode joint type information, it struggles to capture essential information about the overall structure of a person, which might have negatively impacted it's accuracy.

Mean Classification Accuracy (%)		
Only PIF	PIF + 'Block5'	PIF + 'Block5' + 'Block4'
72.42	74.54	**75.67**

Table 5.3: Results of End-to-End learning of activity recognition along with 2D pose.

5.3.4.2 Results for End-to-End Learning

Table 5.3 presents the results of end-to-end learning of activity recognition along with 2D pose estimation. Fixing on the optimal frame sampling rate from the above experiment (i.e, sampling every 4^{th} frame), we train the PIF module of the pose network PifPaf, along with a combination of few other top layers of the PifPaf encoder, namely ResNet 'Block5' and ResNet 'Block4'. The goal is to leverage the pre-trained weights of the pose network while gently modifying the weights of a few top layers with respect to the activity recognition task. This is particularly true since there is no ground-truth supervision provided for the pose network during training. As before, the activity recognition network S-3DG is fully trained from scratch.

As evident from the table, learning the PIF module along with 'Block5' and 'Block4' of the ResNet encoder produces the best results with an overall classification accuracy of 75.67%. Learning more ResNet blocks in the PifPaf encoder is observed to have reduced performance.

Mean Classification Accuracy (%)		
Optical Flow	Bone Orientation Map	End-to-End Learning
74.25	75.46	75.67

Table 5.4: Comparison with baseline approach based on optical flow.

5.3.4.3 Comparison with Baseline Approach

In this work, we consider activity recognition using optical flow as the baseline approach. To this end, we first compute dense and accurate optical flow using TV-L1 [108] algorithm based on the RGB frames from each video in the dataset and then train the activity recognition network S-3DG from scratch using the computed flow frames. Table 5.4, presents the results of activity recognition using optical flow and compares the performance with the top-performing methods from Table 5.2 and Table 5.3. Notably, both Bone Orientation Map and the End-to-End learning approach outperform the optical flow-based method, achieving an accuracy gain of 1.21% and 1.42%, respectively.

The performance improvement of Bone Orientation Map over optical flow is particularly interesting, given the fact that both models are based on the same activity recognition network (i.e., S-3DG) with identical training details. We conjecture that this gain is due to Bone Orientation Map's ability to better distinguish individuals in occluded and close proximity scenarios. It accomplishes this as it encodes enhanced skeleton information by focusing on the region of support over the bones, thereby exhibiting less susceptibility to the occlusion issue compared to optical flow. This phenomenon is visualized in Fig. 5.12. The

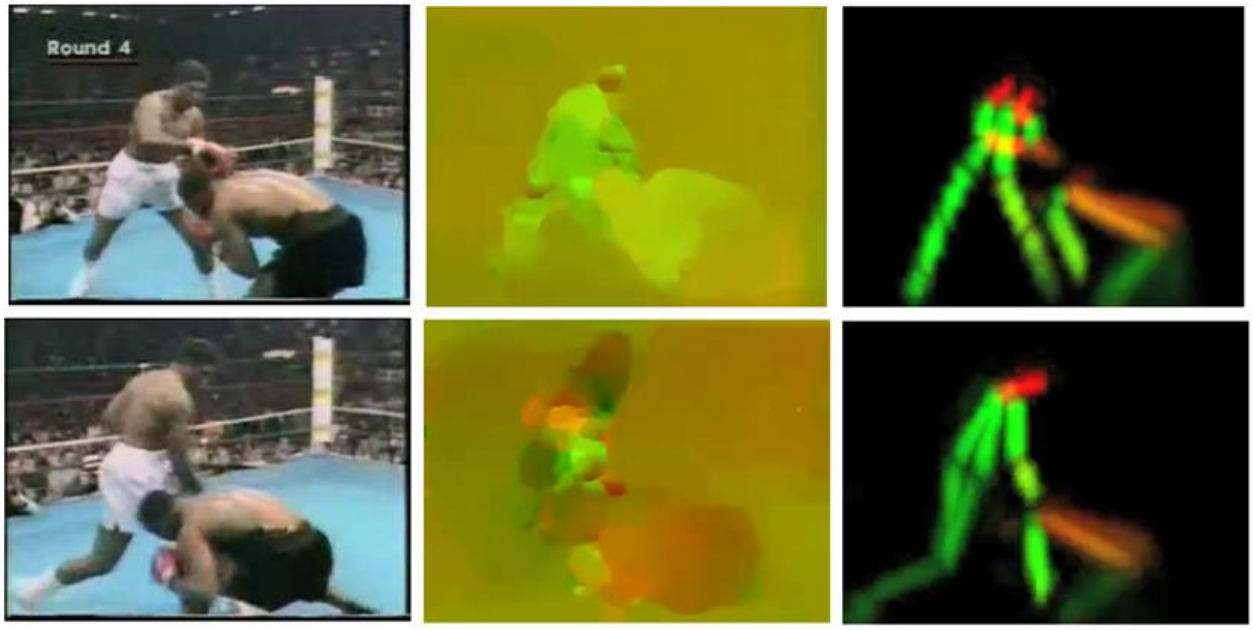


Figure 5.12: Visualization of optical flow and Bone Orientation Map for two example frames from the UCF101 dataset depicting occluded scenarios. The 3-channel image representation for optical flow is achieved by encoding the X and Y components of the flow vector in 'R' and 'G' channels, respectively, while setting the 'B' channel to 0. As obvious from the figure, the Bone Orientation Map demonstrates superior ability to distinguish individuals in crowded scenes compared to optical flow.

higher performance of the end-to-end approach, on the other hand, may be attributed to its increased model capacity.

5.3.4.4 Run-time Performance

In order to evaluate the run-time performance of the proposed methods along with the baseline approach (i.e., optical flow), we compute the total time taken by each method to process a single frame in the test set averaged over the whole set. The experiments are

Method	Computational Steps	Time (ms)/frame
Optical Flow	Extract dense optical flow	204.653
Bone Flow Map	Extract 2D skeleton + Compute Bone Flow Map	17.543 + 12.117 = 27.660
Bone Orientation Map	Extract joint association fields + Compute Bone Orientation Map	16.946 + 0.006 = 16.952
Joint Color Map	Extract joint confidence maps + Compute Joint Color Map	16.946 + 0.008 = 16.954
End-to-End Learning	Extract PIF fields	27.548

Table 5.5: Breakdown of running time of the different motion representation methods. All times are reported on GPU.

conducted on a Desktop computer equipped with an AMD Ryzen Threadripper 3960X 24-Core processor, an NVIDIA RTX 3090 GPU with CUDA 11.2, and 128GB of main memory, all running on the Ubuntu 20.04 Operating System (OS).

Table 5.5 shows the breakdown of the running-time for each method. Optical flow is computed using TV-L1 algorithm [108] based on the implementation provided in OpenCV [139]. On the other hand, the official implementation[1] of OpenPose [132] is leveraged to extract 2D skeleton, joint association fields and joint confidence maps, while the PIF fields for the End-to-End learning approach are extracted based on the official implementation[2] of PifPaf [134].

As evident from the Table, all of the proposed motion representations outperform optical flow in terms of running time. Specifically, Bone Orientation Map and Joint Color Map representations demonstrate a performance speedup of about 12 times over optical flow,

[1] `https://github.com/CMU-Perceptual-Computing-Lab/openpose`
[2] `https://github.com/openpifpaf/openpifpaf`

while Bone Flow Map and End-to-End learning approaches are about 7 times faster than optical flow. Notably, the running time of the proposed methods is mainly dictated by the computational demands of extracting 2D pose information (i.e., 2D skeleton, joint association fields, joint confidence maps, or PIF fields). Given the rapid advancements in GPU hardware speed, this is particularly promising since deep networks are embarrassingly parallelizable. However, this speedup with GPUs may not necessarily scale proportionally when it comes to dense and accurate optical flow computation, mainly due to the nature of computations involved.

5.4 Summary

This chapter proposes novel techniques to model actor motion in videos by leveraging human 2D pose information extracted from the video frames. The proposed methods are agnostic of the 2D pose estimators, hence enjoys the flexibility to capitalize on the state-of-the-art pose estimators. The proposed motion representation techniques demonstrate effectiveness in human activity recognition from videos, with the Bone Orientation Map and the End-to-End learning approach surpassing the performance of optical flow both in speed and accuracy.

Chapter 6

Conclusion

In this chapter, we first present the summary of the thesis in Sec. 6.1, followed by outlining the limitations and the potential future research directions in Sec. 6.2.

6.1 Thesis Summary

In this thesis, we have addressed the human activity detection problem in realistic videos with the goal of identifying the activities as well as the video segments that contain the activities of interest. We particularly address the problem in two different settings – activity detection in the TAD setting concerning the detection of activities in the temporal domain, and activity detection in the STAD setting involving detection of activities both in space and time. To this end, we have developed deep learning based models to advance the state-of-the-art in both problem settings. Additionally, we have proposed several novel techniques

to capture actor motion based on human pose information that are both computationally efficient and effective for video activity recognition.

To perform activity detection in the TAD setting, Chapter 3 introduces a single-stage deep learning model, referred to as the TAD framework. This framework leverages a 3D CNN to extract both appearance and motion information from the input video, and it employs a multi-scale temporal feature hierarchy to carry out activity classification and temporal localization. The end-to-end architecture of the TAD framework allows the learning of task-specific spatio-temporal features efficiently. We further propose a novel fusion method to combine the appearance and motion features in the middle of the processing pipeline, thus allowing the two streams of information to interact with each other. Notably, we integrate the learning of this fusion technique into the TAD framework to comprehensively capture the correlation between these two modalities. Together with this fusion method, the proposed TAD framework outperforms the existing state-of-the-art methods in activity detection in the TAD setting.

Chapter 4 extends the TAD framework to incorporate an innovative spatial localization module. This addition empowers the detection of activities in both space and time within a comprehensive end-to-end framework, referred to as STAD framework. Our approach leverages shared spatio-temporal feature maps to jointly optimize spatial and temporal activity localization, thereby enhancing the overall effectiveness and efficiency of the entire pipeline. As an extension of the TAD framework, the proposed STAD framework enjoys

direct regression on the temporal bounds of the activities which helps improve the temporal localization accuracy.

In Chapter 5, we introduce novel techniques for representing actor motion in videos, prioritizing computational efficiency. This is achieved through the extraction of 2D pose information from video frames, which is then used to capture human motion by considering factors such as bone movement, bone orientation, and body joint positions. Among these techniques, our best-performing method stands out due to its impressive computational efficiency and effectiveness in video activity recognition compared to optical flow based method.

6.2 Limitations and Future Work

Below, we outline the limitations of the various methods introduced in this thesis:

1. **Temporal Footprint Expansion:** In our TAD and STAD frameworks, we enhance the temporal context by inputting a longer sequence of frames into the 3D CNN feature extractor. This improves temporal localization accuracy but comes at the cost of increased GPU memory usage.

2. **Dependence on 2D Pose Estimation:** The motion representation techniques discussed in Chapter 5 rely on the accuracy of the 2D pose estimators. If the pose estimation network fails to provide accurate results, our methods will also suffer.

However, we anticipate that this limitation will diminish as 2D pose estimation techniques advance rapidly, and GPUs become more proficient at processing higher resolution images, resulting in enhanced accuracy for 2D pose estimation.

Below, we present the future work and the potential future research directions that are worth pursuing.

1. **Expanding Fusion for other Input Modalities:** The fusion method proposed in Chapter 3 is agnostic to input modality, making it potentially applicable to fusing other input modalities, thus supporting tasks beyond activity detection. One example could be fusing RGB and LiDAR inputs in the context of 3D object detection. To this end, a recent work [140] has demonstrated success by fusing RGB and LiDAR Bird's Eye View images using our proposed fusion method. In the same vein, fusion of RGB and thermal camera images is worth exploring in order to enhance the performance of 2D object detection, specifically in varying lighting conditions.

2. **Enhancing the STAD Framework:** The proposed STAD framework relies on extracting actor tubes to identify the activity regions. However, this is a surrogate measure since the ultimate goal is to pinpoint the activity regions rather than emphasizing the actor tubes. This realization suggests that research efforts should be directed toward developing more sophisticated methods to be able to trace activity flow in a holistic manner, as opposed to tracking actors.

3. **Extending the Motion Models:** Combining the motion models outlined in Chapter 5 with the appearance models and extending them to address the TAD and STAD problems would constitute a logical progression of this research. Moreover, given the success of the Bone Orientation Map representation, it would be interesting to explore the End-to-End approach discussed in Sec. 5.2.3.2 based on the raw joint association vector fields.

4. **Developing Representative Dataset for STAD:** While there are large-scale datasets available for the TAD problem, a notable gap exists when it comes to representative benchmark datasets for the STAD problem that accurately mirror real-world situations. Specifically, the existing datasets fall short in providing instances that depict scenarios where the same actor engages in multiple consecutive activities of the same and/or different types. This particular characteristic of an STAD dataset holds significant importance, especially in the context of supporting real-world applications like autonomous driving and retail analytics. These application scenarios often involve actors performing multiple activities sequentially, such as pedestrians waiting to cross the street followed by the actual crossing activity, or customers ordering food, paying for their orders, and subsequently picking up their food in a fast-food restaurant.

Bibliography

[1] "Hours of video uploaded to youtube every minute." https://www.statista.com/statistics/259477/hours-of-video-uploaded-to-youtube-every-minute/. Accessed: 2023-10-04. 1

[2] Cisco Technical Report, "Cisco Visual Networking Index: Forecast and Trends, 2017–2022." https://davidellis.ca/wp-content/uploads/2019/05/cisco-vni-feb2019.pdf. Accessed: 2023-09-28. 1

[3] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, 2014. 2, 14, 23, 26, 46, 75

[4] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Val Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *ECCV*, 2016. 2, 15, 23, 46

[5] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell, "Actionvlad: Learning spatio-temporal aggregation for action classification," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 101

[6] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 15, 23, 46, 75, 101

[7] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 2, 15, 101

[8] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy, "Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification," in *The European Conference on Computer Vision (ECCV)*, 2018. 2, 15, 27, 40, 56, 75, 101, 102

[9] J. C. Stroud, D. A. Ross, C. Sun, J. Deng, and R. Sukthankar, "D3d: Distilled 3d networks for video action recognition," *ArXiv*, vol. abs/1812.08249, 2018. 2, 40, 75

[10] S. Saha, *Spatio-temporal Human Action Detection and Instance Segmentation in Videos*. PhD thesis, School of Engineering, Computing and Mathematics Oxford Brookes University, UK, 2018. 4

[11] NIST, "Activities in extended videos prize challenge." `https://actev.nist.gov/prizechallenge`. Accessed: 2023-10-05. 4

[12] A. Rasouli, *The Role of Context in Understanding and Predicting Pedestrian Behavior in Urban Traffic Scenes*. Phd thesis, York University, Toronto, Canada, August 2020. Available at `https://yorkspace.library.yorku.ca/bitstreams/4aeeae77-8692-41f9-afca-7416b6d3e260/download`. 5

[13] H. Pham, Z. Dai, Q. Xie, M.-T. Luong, and Q. V. Le, "Meta pseudo labels," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 5

[14] A. Brock, S. De, S. L. Smith, and K. Simonyan, "High-performance large-scale image recognition without normalization," *arXiv preprint arXiv:2102.06171*, 2021. 5

[15] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur, "Sharpness-aware minimization for efficiently improving generalization," in *International Conference on Learning Representations*, 2021. 5

[16] H. Touvron, A. Vedaldi, M. Douze, and H. Jegou, "Fixing the train-test resolution discrepancy," in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019. 5

[17] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, 2015. 5, 12

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *ECCV*, 2016. 5, 14, 24, 27, 59

[19] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," *arXiv preprint arXiv:2103.14030*, 2021. 5

[20] S. Qiao, L.-C. Chen, and A. Yuille, "Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution," *arXiv preprint arXiv:2006.02334*, 2020. 5

[21] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10778–10787, 2020. 5

[22] Y. Fang, B. Liao, X. Wang, J. Fang, J. Qi, R. Wu, J. Niu, and W. Liu, "You only look at one sequence: Rethinking transformer in vision through object detection," *CoRR*, vol. abs/2106.00666, 2021. 5

[23] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7464–7475, June 2023. 5

[24] G. Jocher, A. Chaurasia, and J. Qiu, ""yolo by ultralytics"." `https://github.com/ultralytics/`. ultralytics, 2023. Accessed: 2023-10-05. 5

[25] G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T. Lin, E. D. Cubuk, Q. V. Le, and B. Zoph, "Simple copy-paste is a strong data augmentation method for instance segmentation," *CoRR*, vol. abs/2012.07177, 2020. 5

[26] Y. Fang, S. Yang, X. Wang, Y. Li, C. Fang, Y. Shan, B. Feng, and W. Liu, "Queryinst: Parallelly supervised mask query for instance segmentation," *arXiv preprint arXiv:2105.01928*, 2021. 5

[27] X. Wang, R. Zhang, T. Kong, L. Li, and C. Shen, "Solov2: Dynamic and fast instance segmentation," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, eds.), vol. 33, pp. 17721–17732, Curran Associates, Inc., 2020. 5

[28] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, *et al.*, "Segment anything," *arXiv preprint arXiv:2304.02643*, 2023. 5

[29] H. Xu, L. Yang, S. Sclaroff, K. Saenko, and T. Darrell, "Spatio-temporal action detection with multi-object interaction," *ArXiv*, vol. abs/2004.00180, 2020. 6, 18, 53, 54, 70

[30] L. S. S. Karaman and A. D. Bimbo., "Fast saliency based pooling of fisher encoded dense trajectories," in *ECCV THUMOS Workshop*, 2014. 12

[31] J. V. D. Oneata and C. Schmid, "The lear submission at thumos 2014," in *ECCV THUMOS Workshop*, 2014. 12

[32] Y. Y. Q. L. Wang and X. Tang, "Action recognition and detection by combining motion and appearance features," in *ECCV THUMOS Workshop*, 2014. 12

[33] Z. Shou, D. Wang, and S.-F. Chang, "Temporal action localization in untrimmed videos via multi-stage cnns," in *CVPR*, 2016. 12, 49, 50

[34] R. Girshick, "Fast r-cnn," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, 2015. 12, 38

[35] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Computer Vision and Pattern Recognition*, 2015. 12

[36] X. Dai, B. Singh, G. Zhang, L. Davis, and Y. Qiu Chen, "Temporal context network for activity localization in videos," 2017. 12, 49

[37] J. Gao, Z. Yang, and R. Nevatia, "Cascaded boundary regression for temporal action detection," *ArXiv*, vol. abs/1705.01180, 2017. 12, 49

[38] J. Gao, Z. Yang, K. Chen, C. Sun, and R. Nevatia, "Turn tap: Temporal unit regression network for temporal action proposals," in *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 12

[39] H. Xu, A. Das, and K. Saenko, "R-c3d: Region convolutional 3d network for temporal activity detection," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017. 12, 23, 49

[40] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, 2015. 12, 13, 23, 45

[41] Y. Bai, H. Xu, K. Saenko, and B. Ghanem, "Contextual multi-scale region convolutional 3d network for activity detection," *ArXiv*, vol. abs/1801.09184, 2018. 12, 23

[42] Y.-W. Chao, S. Vijayanarasimhan, B. Seybold, D. A. Ross, J. Deng, and R. Sukthankar, "Rethinking the faster r-cnn architecture for temporal action localization," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 13, 15, 22, 39, 49, 54, 61, 75

[43] H. Idrees, A. R. Zamir, Y. Jiang, A. Gorban, I. Laptev, R. Sukthankar, and M. Shah, "The THUMOS challenge on action recognition for videos "in the wild"," *CoRR*, vol. abs/1604.06182, 2016. 13, 14, 39

[44] V. Escorcia, F. C. Heilbron, J. C. Niebles, and B. Ghanem, "Daps: Deep action proposals for action understanding," in *ECCV*, 2016. 13, 47, 49

[45] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei, "End-to-end learning of action detection from frame glimpses in videos," *CoRR*, vol. abs/1511.06984, 2015. 13

[46] S. Ma, L. Sigal, and S. Sclaroff, "Learning activity progression in lstms for activity detection and early detection," in *CVPR*, 2016. 13

[47] B. Singh, T. K. Marks, M. Jones, O. Tuzel, and M. Shao, "A multi-stream bi-directional recurrent neural network for fine-grained action detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 13

[48] A. Montes, A. Salvador, S. Pascual, and X. Giro-i Nieto, "Temporal activity detection in untrimmed videos with recurrent neural networks," in *1st NIPS Workshop on Large Scale Computer Vision Systems*, 2016. 13

[49] S. Buch, V. Escorcia, B. Ghanem, L. Fei-Fei, and J. C. Niebles, "End-to-end, single-stream temporal action detection in untrimmed videos," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2017. 13, 22, 49, 54, 61

[50] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015. 13, 81

[51] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 14, 24, 27, 59

[52] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 14, 59

[53] T. Lin, X. Zhao, and Z. Shou, "Single shot temporal action detection," in *Proceedings of the 25th ACM International Conference on Multimedia*, 2017. 14, 22, 29, 30, 37, 40, 45, 49, 50, 54, 61

[54] F. Long, T. Yao, Z. Qiu, X. Tian, J. Luo, and T. Mei, "Gaussian temporal awareness networks for action localization," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 14, 22, 45, 49, 54, 61

[55] Y. Huang, Q. Dai, and Y. Lu, "Decoupling localization and classification in single shot temporal action detection," in *2019 IEEE International Conference on Multimedia and Expo (ICME)*, 2019. 14, 15, 22, 45, 46, 47, 48, 49, 50, 54, 61, 75

[56] B. Jiang, M. Wang, W. Gan, W. Wu, and J. Yan, "Stm: Spatiotemporal and motion encoding for action recognition," in *The IEEE International Conference on Computer Vision (ICCV)*, 2019. 15

[57] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 15, 23, 31, 35

[58] C. Feichtenhofer, A. Pinz, and R. Wildes, "Spatiotemporal residual networks for video action recognition," in *Advances in Neural Information Processing Systems 29*, 2016. 15, 23, 31

[59] E. Kazakos, A. Nagrani, A. Zisserman, and D. Damen, "Epic-fusion: Audio-visual temporal binding for egocentric action recognition," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 15, 23, 31

[60] M. Jain, J. Van Gemert, H. Jégou, P. Bouthemy, and C. G. M. Snoek, "Action localization with tubelets from motion," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 740–747, 2014. 15

[61] D. Oneata, J. Revaud, J. Verbeek, and C. Schmid, "Spatio-temporal object detection proposals," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), pp. 737–752, Springer International Publishing, 2014. 15

[62] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, 2013. 16

[63] S. Manen, M. Guillaumin, and L. Van Gool, "Prime object proposals with randomized prim's algorithm," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2013. 16

[64] H. Wang, A. Kläser, C. Schmid, and L. Cheng-Lin, "Action Recognition by Dense Trajectories," in *CVPR 2011 - IEEE Conference on Computer Vision & Pattern Recognition*, (Colorado Springs, United States), pp. 3169–3176, IEEE, June 2011. 16

[65] K. Soomro, H. Idrees, and M. Shah, "Action localization in videos through context walk," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 3280–3288, 2015. 16

[66] H. Wang and C. Schmid, "Action Recognition with Improved Trajectories," in *ICCV - IEEE International Conference on Computer Vision*, (Sydney, Australia), pp. 3551–3558, IEEE, Dec. 2013. 16

[67] G. Gkioxari and J. Malik, "Finding action tubes.," in *CVPR*, pp. 759–768, IEEE Computer Society, 2015. 16, 53

[68] G. Singh, S. Saha, M. Sapienza, P. H. S. Torr, and F. Cuzzolin, "Online real-time multiple spatiotemporal action localisation and prediction," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 16, 53, 64, 70

[69] Y. Ye, X. Yang, and Y. Tian, "Discovering spatio-temporal action tubes," *J. Visual Communication and Image Representation*, 2019. 16, 53

[70] J. G. Zhenheng Yang and R. Nevatia, "Spatio-temporal action detection with cascade proposal and location anticipation," in *Proceedings of the British Machine Vision Conference (BMVC)* (G. B. Tae-Kyun Kim, Stefanos Zafeiriou and K. Mikolajczyk, eds.), pp. 95.1–95.12, BMVA Press, September 2017. 16, 53

[71] X. Peng and C. Schmid, "Multi-region two-stream R-CNN for action detection," in *ECCV - European Conference on Computer Vision*, vol. 9908 of *Lecture Notes in Computer Science*, (Amsterdam, Netherlands), pp. 744–759, Springer, Oct. 2016. 16, 53

[72] S. Saha, G. Singh, M. Sapienza, H. S. P. Torr, and F. Cuzzolin, "Deep learning for detecting multiple space-time action tubes in videos," *BMVC*, 2016. 16, 53

[73] P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "Learning to track for spatio-temporal action localization," in *ICCV - IEEE International Conference on Computer Vision*, (Santiago, Chile), pp. 3164–3172, IEEE, Dec. 2015. 16, 53, 64, 70

[74] R. Hou, C. Chen, and M. Shah, "Tube convolutional neural network (t-cnn) for action detection in videos," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 16, 53, 70

[75] V. Kalogeiton, P. Weinzaepfel, V. Ferrari, and C. Schmid, "Action Tubelet Detector for Spatio-Temporal Action Localization," in *ICCV - IEEE International Conference on Computer Vision*, (Venice, Italy), pp. 4415–4423, IEEE, Oct. 2017. 16, 53, 64, 70

[76] S. Saha, G. Singh, and F. Cuzzolin, "Amtnet: Action-micro-tube regression by end-to-end trainable deep architecture," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 16, 70

[77] Y. Li, Z. Wang, L. Wang, and G. Wu, "Actions as moving points," in *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI*, (Berlin, Heidelberg), p. 68–84, Springer-Verlag, 2020. 16, 70, 71

[78] L. Song, S. Zhang, G. Yu, and H. Sun, "Tacnet: Transition-aware context network for spatio-temporal action detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 17, 54, 70

[79] X. Yang, X. Yang, M.-Y. Liu, F. Xiao, L. S. Davis, and J. Kautz, "Step: Spatio-temporal progressive learning for video action detection," in *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019. 17, 54, 70, 71

[80] J. Wu, Z. Kuang, L. Wang, W. Zhang, and G. Wu, "Context-aware rcnn: A baseline for action detection in videos," in *Computer Vision – ECCV 2020* (A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, eds.), (Cham), pp. 440–456, Springer International Publishing, 2020. 17

[81] J. Pan, S. Chen, M. Z. Shou, Y. Liu, J. Shao, and H. Li, "Actor-context-actor relation network for spatio-temporal action localization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 464–474, June 2021. 17

[82] J. Ni, J. Qin, and D. Huang, "Identity-aware graph memory network for action detection," in *Proceedings of the 29th ACM International Conference on Multimedia*, MM '21, (New York, NY, USA), p. 3437–3445, Association for Computing Machinery, 2021. 17

[83] G. Cheron, I. Laptev, and C. Schmid, "P-cnn: Pose-based cnn features for action recognition," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015. 19, 77

[84] I. Lillo, J. Niebles, and A. Soto, "A hierarchical pose-based approach to complex action understanding using dictionaries of actionlets and motion poselets," in *CVPR*, 2016. 19, 77

[85] C. Wang, Y. Wang, and A. L. Yuille, "An approach to pose-based action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013. 19, 77

[86] K. Soomro, H. Idrees, and M. Shah, "Predicting the where and what of actors and actions through online action localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 19, 77

[87] M. A. Gowayyed, M. Torki, M. E. Hussein, and M. El-Saban, "Histogram of oriented displacements (hod): Describing trajectories of human joints for action recognition," IJCAI '13, p. 1351–1357, AAAI Press, 2013. 19, 77

[88] Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 19, 77

[89] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. 19

[90] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "Ntu rgb+d: A large scale dataset for 3d human activity analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 19

[91] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Advances in Neural Information Processing Systems* (D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, eds.), vol. 21, Curran Associates, Inc., 2009. 19

[92] J. Liu, A. Shahroudy, D. Xu, and G. Wang, "Spatio-temporal LSTM with trust gates for 3d human action recognition," in *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), vol. 9907 of *Lecture Notes in Computer Science*, pp. 816–833, Springer, 2016. 19

[93] H. Yonghong, L. Zhaoyang, W. Pichao, and L. Wanqing, "Skeleton optical spectra based action recognition using convolutional neural networks.," in *In IEEE Transactions on Circuits and Systems for Video Technology*, 2016. 20, 77

[94] C. Caetano, J. Sena, F. F. Bremond, J. A. Dos Santos, and W. Robson Schwartz, "SkeleMotion: A New Representation of Skeleton Joint Sequences Based on Motion Information for 3D Action Recognition," in *AVSS 2019 - 16th IEEE International*

Conference on Advanced Video and Signal-based Surveillance, (Taipei, Taiwan), Sept. 2019. 20, 77

[95] P. Wang, Z. Li, Y. Hou, and W. Li, "Action recognition based on joint trajectory maps using convolutional neural networks," MM '16, p. 102–106, Association for Computing Machinery, 2016. 20, 77

[96] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu, "An end-to-end spatio-temporal attention model for human action recognition from skeleton data," AAAI'17, p. 4263–4270, AAAI Press, 2017. 20

[97] M. Zolfaghari, G. L. Oliveira, N. Sedaghat, and T. Brox, "Chained multi-stream networks exploiting pose, motion, and appearance for action classification and detection," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 20

[98] W. Wang, J. Zhang, C. Si, and L. Wang, "Pose-Based Two-Stream Relational Networks for Action Recognition in Videos," *arXiv e-prints*, p. arXiv:1805.08484, May 2018. 20

[99] F. Baradel, C. Wolf, and J. Mille, "Human Activity Recognition with Pose-driven Attention to RGB," in *BMVC 2018 - 29th British Machine Vision Conference*, (Newcastle, United Kingdom), pp. 1–14, Sept. 2018. 21

[100] Z. Shou, J. Chan, A. Zareian, K. Miyazawa, and S.-F. Chang, "Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos," 2017. 23

[101] J. B. Tenenbaum and W. T. Freeman, "Separating style and content with bilinear models," *Neural Computation*, 2000. 24

[102] C. Feichtenhofer, A. Pinz, and R. P. Wildes, "Spatiotemporal multiplier networks for video action recognition," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 31, 46

[103] J. Kim, K. W. On, W. Lim, J. Kim, J. Ha, and B. Zhang, "Hadamard product for low-rank bilinear pooling," in *5th International Conference on Learning Representations, ICLR*, 2017. 31, 32

[104] Z. Yu, J. Yu, J. Fan, and D. Tao, "Multi-modal factorized bilinear pooling with co-attention learning for visual question answering," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017. 31, 33, 34

[105] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. 35, 103

[106] "Mexaction2." `http://mexculture.cnam.fr/Datasets/mex+action+dataset.html`. Accessed: 2020-04-01. 39, 40

[107] K. Soomro, A. R. Zamir, M. Shah, K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *CoRR*, 2012. 40, 63, 93, 95, 99, 104

[108] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime tv-l1 optical flow," in *Proceedings of the 29th DAGM Conference on Pattern Recognition*, 2007. 40, 111, 113

[109] S. Das, M. Thonnat, K. Sakhalkar, M. F. Koperski, F. Bremond, and G. Francesca, "A New Hybrid Architecture for Human Activity Recognition from RGB-D videos," in *MMM 2019 - 25th International Conference on MultiMedia Modeling*, 2019. 43

[110] S. Buch, V. Escorcia, C. Shen, B. Ghanem, and J. C. Niebles, "Sst: Single-stream temporal action proposals," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 47, 49

[111] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, X. Tang, and D. Lin, "Temporal action detection with structured segment networks," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 49

[112] T. Lin, X. Zhao, H. Su, C. Wang, and M. Yang, "Bsn: Boundary sensitive network for temporal action proposal generation," in *The European Conference on Computer Vision (ECCV)*, 2018. 49

[113] C. Gu, C. Sun, D. A. Ross, C. Vondrick, C. Pantofaru, Y. Li, S. Vijayanarasimhan, G. Toderici, S. Ricco, R. Sukthankar, C. Schmid, and J. Malik, "Ava: A video dataset of spatio-temporally localized atomic visual actions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 54, 68, 70, 71

[114] S. Saha, G. Singh, and F. Cuzzolin, "Two-stream amtnet for action detection," *CoRR*, vol. abs/2004.01494, 2020. 54

[115] M. A. Rahman and R. Laganière, "Mid-level fusion for end-to-end temporal activity detection in untrimmed video," in *31st British Machine Vision Conference (BMVC2020), UK, Sept. 7-10*, 2020. 54, 61, 75

[116] G. Singh, *Online spatiotemporal action detection and prediction via casual representations*. Phd thesis, Oxford Brookes University, Oxford, UK, 2019. Available at `https://oxfordbrookes.on.worldcat.org/oclc/1338136872`. 55

[117] M. A. Rahman, P. Kapoor, R. Laganière, D. Laroche, C. Zhu, X. Xu, and A. Osman Ors, "Deep people detection: A comparative study of ssd and lstm-decoder," in *2018 15th Conference on Computer and Robot Vision (CRV)*, pp. 305–312, 2018. 58

[118] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," *CoRR*, vol. abs/1703.07402, 2017. 59, 65

[119] M. D. Rodriguez, J. Ahmed, and M. Shah, "Action mach: a spatio-temporal maximum average correlation height filter for action recognition," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2008. 63

[120] S. Saha, G. Singh, M. Sapienza, P. Torr, and F. Cuzzolin, "Deep learning for detecting multiple space-time action tubes in videos," 2016. 64, 70

[121] J. Carreira, E. Noland, C. Hillier, and A. Zisserman, "A short note on the kinetics-700 human action dataset," *CoRR*, vol. abs/1907.06987, 2019. 65

[122] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," in *CVPR*, 2017. 65

[123] X. Peng and C. Schmid, "Multi-region two-stream R-CNN for action detection," in *ECCV - European Conference on Computer Vision*, 2016. 70

[124] R. R. A. Pramono, Y.-T. Chen, and W.-H. Fang, "Hierarchical self-attention network for action localization in videos," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 70

[125] J. Zhao and C. G. M. Snoek, "Dance with flow: Two-in-one stream action detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 70

[126] G. Chéron, J.-B. Alayrac, I. Laptev, and C. Schmid, "A flexible model for training action localization with varying levels of supervision," in *Neural Information Processing Systems (NeurIPS)*, 2018. 70

[127] G. Singh, S. Akrigg, M. D. Maio, V. Fontana, R. J. Alitappeh, S. Khan, S. Saha, K. Jeddisaravi, F. Yousefi, J. Culley, T. Nicholson, J. Omokeowa, S. Grazioso, A. Bradley, G. D. Gironimo, and F. Cuzzolin, "Road: The road event awareness dataset for autonomous driving," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 1036–1054, 2023. 70, 71

[128] Z. Qiu, T. Yao, and T. Mei, "Learning spatio-temporal representation with pseudo-3d residual networks," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 75

[129] M. A. Rahman and R. Laganière, "Single-stage end-to-end temporal activity detection in untrimmed videos," in *17th Conference on Computer and Robot Vision (CRV2020), Ottawa, ON, Canada, May 13-15*, pp. 206–213, IEEE, 2020. 75

[130] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Doll'a r, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. 76, 79, 107

[131] T. L. Munea, Y. Z. Jembre, H. T. Weldegebriel, L. Chen, C. Huang, and C. Yang, "The progress of human pose estimation: A survey and taxonomy of models applied in 2d human pose estimation," *IEEE Access*, vol. 8, pp. 133330–133348, 2020. 76

[132] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 1, pp. 172–186, 2021. 76, 79, 80, 81, 113

[133] S. Kreiss, L. Bertoni, and A. Alahi, "Pifpaf: Composite fields for human pose estimation," *CoRR*, vol. abs/1903.06593, 2019. 76, 79, 103, 104

[134] S. Kreiss, L. Bertoni, and A. Alahi, "OpenPifPaf: Composite Fields for Semantic Keypoint Detection and Spatio-Temporal Association," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–14, March 2021. 76, 79, 113

[135] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2d human pose estimation: New benchmark and state of the art analysis," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 79

[136] D. B. West, *Introduction to Graph Theory*. Prentice Hall, 2 ed., September 2000. 83

[137] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," *CoRR*, vol. abs/1812.03982, 2018. 101

[138] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org. 107

[139] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000. 113

[140] Y. Massoud, "Sensor fusion for 3d object detection for autonomous vehicles," October 2021. Available at `https://ruor.uottawa.ca/bitstream/10393/42812/1/Massoud_Yahya_2021_thesis.pdf`. 118