Conception d'architectures compactes pour la détection spatiotemporelle d'actions en temps réel

Yu Liu

THESE DE DOCTORAT DE L'ETABLISSEMENT UNIVERSITE BOURGOGNE FRANCHE-COMTE

PREPAREE A L'UNIVERSITE DE BOURGOGNE

Ecole doctorale n° 37

Sciences Pour l'Ingénieur et Microtechniques (SPIM)

Doctorat de Instrumentation, Informatique de l'Image

Par

Yu LIU

Lightweight Architectures for Spatiotemporal Action Detection in Real-Time

Thèse présentée et soutenue à Dijon, le 25/05/2022

Composition du Jury :

Professeur Olivier SENTIEYS	Université de Rennes	Rapporteur
Professeur Stéphane CANU	INSA de Rouen	Rapporteur
Professeure Catherine ACHARD	Sorbonne Université	Examinatrice
Professeur Fabrice MERIAUDEAU	Université de Bourgogne Franche-Comté	Examinateur
Professeure Fan YANG	Université de Bourgogne Franche-Comté	Co-directrice de thèse
Professeur Dominique GINHAC	Université de Bourgogne Franche-Comté	Directeur de thèse

Declaration of Authorship

I, Yu LIU, declare that this thesis titled, "Lightweight Modeling of Spatiotemporal Action Detection in Real-Time" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 

Date: 15.08.2022

"Most memory of my Ph.D. study was filled with wonders when it would finish, until when I am actually close to complete then I wished I could start over and explore more."

Abstract

Title: Lightweight Modeling of Spatiotemporal Action Detection in Real-Time

In the last decade, the explosive growth of video content has driven numerous application demands for automating action detection in space and time. Aside from accurate detection, vast sensing scenarios in the real-world mandate incremental, instantaneous processing of scenes under restricted computational budgets. The main challenge here lies in dependence on heavy 3D Convolutional Neural Networks (CNN) or explicit motion computation (e.g., optical flow) to extract pertinent spatiotemporal information.

To this end, we propose three lightweight action detection architectures coupling various spatiotemporal modeling schemes with compact 2D CNNs. Our first intuition was to accelerate frame-level action detection by allocating computationally expensive feature extraction to only a sparse set of video frames while approximating the rest. Meanwhile, we accumulated multiple observations over time to efficiently model temporal variations of actions.

Subsequently, we explored processing a series of video frames and predicting the underlying action-specific bounding boxes concurrently (i.e., tubelets). Specifically, modeling of an action sequence was decoupled into multi-frame feature aggregation and trajectory tracking for enhanced action classification and localization, respectively.

Finally, we devised a flow-like motion representation that can be computed on-the-fly from raw video frames. Our aforementioned tubelet detector was extended into two-CNN pathways to jointly extract actions' static visual and dynamic cues. We demonstrate that our online action detectors progressively improve and obtain a superior mix of accuracy, efficiency, and speed performance.

Titre: Architectures compactes pour la détection spatiotemporelle d'actions en temps réel

Depuis la dernière décennie, la croissance explosive de vidéos fait naître un large éventail d'applications nécessitant l'analyse et la compréhension des actions humaines. Les recherches connexes actuelles se concentrent principalement sur l'amélioration des performances de détection de reconnaissance d'actions. Cependant, certains scénarios du monde réel exigent des réponses spontanées réalisées sur des systèmes embarqués avec des ressources limitées. Les méthodes existantes sont difficilement déployables dans ce contexte, puisqu'elles utilisent des architectures lourdes comme réseaux de neurones convolutifs 3D pour extraire les caractéristiques spatiotemporelles d'un vidéo ou calculent explicitement le flux optique des mouvement. Dans cette thèse, nous explorons la faisabilité de réaliser la détection spatiotemporelle d'action satisfaisant simultanément plusieurs contraintes d'applications grand publique : robustesse, temps réel, bas coût, ergonomie, bonne portabilité et longue autonomie énergétique.

Pour ce faire, nous proposons trois architectures de détection d'action couplant différents schémas de modélisation spatiotemporelle avec des CNN 2D compacts. La première réalise la détection au niveau d'une image statique en approximant les caractéristiques de la plupart des frames d'une séquence vidéo pour accélérer le traitement. Nous explorons ensuite un paradigme de détection multi-images pour traiter simultanément la détection temporelle et la prédiction des boîtes englobantes des actions spécifiques pour former des tubelets. Enfin, nous concevons une représentation de mouvement de type flux calculé à la volée à partir d'images vidéo brutes, et étendons l'approche de détection de tubelet à deux CNN pour extraire conjointement les caractéristiques spatiales et temporelles des actions. Les résultats expérimentaux obtenus sur des bases de données publiques montrent les améliorations progressives de nos approches en termes de précision, d'efficacité, et de vitesse de traitement.

Acknowledgements

First and foremost, I would like to express my earnest gratitude to my director Prof. Dominique Ginhac, and co-supervisor Prof. Fan Yang for their invaluable time, guidance, and support for my thesis. Dominique and Fan's openness to various state-of-the-art breakthroughs, and their unending trust in me approaching research problems steadily at my own pace, have motivated me to always explore and push myself beyond the limits. I am also grateful to their patience to discussions, thorough reviews on my works, as well as assisting me to handle different administrative activities. Moreover, I deeply appreciate their compassion and generosity to prolong my research duration under the influence of the COVID-19 pandemic, permitting me to focus on my research at ease.

I would also like to thank all the staffs of the ACHIEVE Innovative Training Network, which funds and grants me such an exceptional opportunity to pursue a doctorate well aligned with my area of interest. A special thanks to the project coordinator Ricardo Carmona, who devotes to gather such a talented group of young researchers, encourage collaborated research, and organize a variety of training events to hone our skills. Another special thanks to Walther Hernandez, a close colleague in the project for numerous discussions and advises on my research topics.

My acknowledgement would not be complete without mentioning a few names of my colleagues at the ImViA laboratory, who have instigated plenty of brainstorming moments and assisted me in different ways: Reda Belaiche, Deivid Botina, Yuly Castro, Abir Zendagui, Ramamoorthy Luxman, David Lewis, Rita Meziati, Romain Cendre, Antoine Leger, Arsalan Khawaja, Houda Rafi, and Youssef Skandrani. Due acknowledgement goes to my Master thesis supervisor Prof. Sen Wang, for his time and guidance that helped me formulate a healthy, persevering attitude toward research and gave inspiration to further deepen my scientific understanding through Ph.D. studies.

Last but not least, I am deeply thankful to my parents, George Liu and Cathy Lin, for their unconditional support in my decision of pursuing a doctorate. Their ceaseless encouragement has led me out of many challenging situations. I cannot imagine under any circumstance that I would be writing this manuscript today without them at my back. I would also like to specially thank my twin brother, Tso Liu, who happened to start a new career the same time as my Ph.D. studies began. Throughout the years, our continuous inspiration and motivation to each other at ups and downs help us grow stronger, stay positive and surmount countless obstacles together.

Contents

List of Figures

List of Tables

List of Abbreviations

CNN	Convolutional Neural Network
SVM	Support Vector Machine
RoI	Region of Interest
RPN	Region Proposal Network
FPS	Frame-per-Second
MAC	Multiply–Accumulate
IoU	Intersection-over-Union
R-CNN	Region-based CNN
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
mAP	mean Average Precision
SSD	Single Shot (Multibox) Detector
NMS	Non-Maximal Suppression

To my beloved parents, without whom none of my success would be possible.

Chapter 1

Introduction

1.1 Context

A video consisting of a series of ordered frames can encompass a vast variety of information that supersedes any single image. In recent years, acquisition and distribution of video data have become increasingly cheap via easily accessible mobile platforms and applications from consumers, recording from millions of closed-circuit cameras, as well as video broadcast of various events and a rapidly growing number of creative content (e.g., by Netflix and Amazon). In 2019, it was estimated that 500 hours of content were uploaded to YouTube every minute, and nearly 80% of the internet traffic was associated with transferring video data [1]. Given such an explosive growth of visual information to process anytime, everywhere in the world, it becomes crucial to interpret the underlying video content automatically by machines. Meanwhile, the field of artificial intelligence, more specifically, computer vision, has progressed significantly in the last decade with the aim of empowering machines to address specific vision-based tasks at human-level capacities. This makes computer vision techniques highly sought-after to solve the video understanding problem.

The introduction of deep learning, particularly Convolutional Neural Network (CNN) [2], has re-defined modern approaches toward computer vision problems. As the receptive fields of its filters progressively expand in deep layers of the network, a CNN hierarchically learns image representations from low-level shapes to abstract semantics. Due to their superior capacity to model highly complex visual cues, along with the rising availability of large-scale annotated image datasets and graphical processing units (GPU) for accelerating model training and inference, CNNs have become the de facto standard since 2012 [3] to approach the majority of computer vision tasks in the image domain (for example, image classification, object detection and segmentation).

Replicating the successes of CNNs in image-space to the realm of video understanding is a non-trivial task, however. Due to the additional temporal dimension to consider in videos, the span of a video understanding problem is much wider than that in images. Some of the most visited topics by the scientific community include but are not limited to human action understanding, video compression, object tracking, optical flow, and video captioning. These problems pose a number of different challenges such as variations in the viewpoint of the camera, occlusion of salient targets, motion blur, and changes in luminance, etc.

Among the variety of problems associated with video analysis, this thesis particularly focuses on **human action understanding**. We argue that most videos seeking automated analysis tend to be human-centric. In other words, they are created around humans and their activities, such as footage for surveillance monitoring, interactive marketing, and customer experience analysis. Additionally, the task of action understanding implicitly requires addressing several of the above-mentioned problems. For instance, an automated analysis system may need to simultaneously detect and track moving humans and possibly surrounding objects via optical flow in order to capture their dynamic interactions for action classification.

1.2 Types of action understanding tasks

We first and formally define a human *action* in our thesis as an event which is *intentionally* or *purposely* carried out by the human *actor* from which an observer could derive meaning, e.g., walking, running, jumping, skateboarding, etc. There are other terms such as an activity and an event that have also been used when describing human behavior. While some studies interpret an activity as an ensemble of actions [4], e.g., playing basketball, salsa dancing, etc., the distinction between an activity and action is actually weak and the two terms have been used interchangeably throughout literature [5][6]. Here, we choose to use *action* throughout this manuscript for consistency. On the other hand, an *event* is defined as an incident that does not necessarily take place intentionally by the actor (such as falling and yawning), which is related but not within the scope of this thesis.

1.2.1 Types of action understanding tasks

There are several sub-tasks falling under the umbrella of action understanding. We describe three major ones and pin-point their distinctions from the others as referred from the survey article by Hutchinson et al. [7].

1. Action recognition can be considered one of the most fundamental and studied problems in action understanding. In the most common setup, it classifies a complete video input by the action occurring in the video and outputs a video-level label. The task can be further categorized into trimmed or untrimmed action recognition, depending on whether the underlying action spans the entire duration of the video or not.

2. Temporal action detection/localization concerns identifying the temporal boundaries, i.e., start and end timestamps (frames) of each action instance in the video along with its action category. Multiple action instances can be present in a video with different temporal extents.

3. Spatiotemporal action detection/localization, corresponds to the task where the action category, temporal boundaries and spatial bounding boxes all need to be inferred for each action instance in the video. In this case, each action instance is represented as an *action tube* composed of a set of bounding boxes linked across time; the temporal extent of each instance can be determined by the start and end frames of the tube.

Other action understanding problems are typically derived from the above ones. For example, the task of early action prediction is a variant of action recognition. It requires predicting the action label of a video after only observing a few frames from the initial part of the video. Similarly, *online* spatiotemporal action detection concerns predicting the bounding boxes and label of an action tube only in the observed part of the video (only current and past video frames can be accessed). We illustrate some of these related tasks in Figure 1.1.

In this thesis, **we solely focus on addressing spatiotemporal action detection in an online fashion**. Note that the terms *detection* and *localization* are used interchangeably throughout this manuscript, while the prefix *spatiotemporal* is sometimes omitted to avoid redundancy. Overall, action detection is more challenging as it needs to cope with both spatial and temporal localization of co-occurring actions (possibly with different categories).

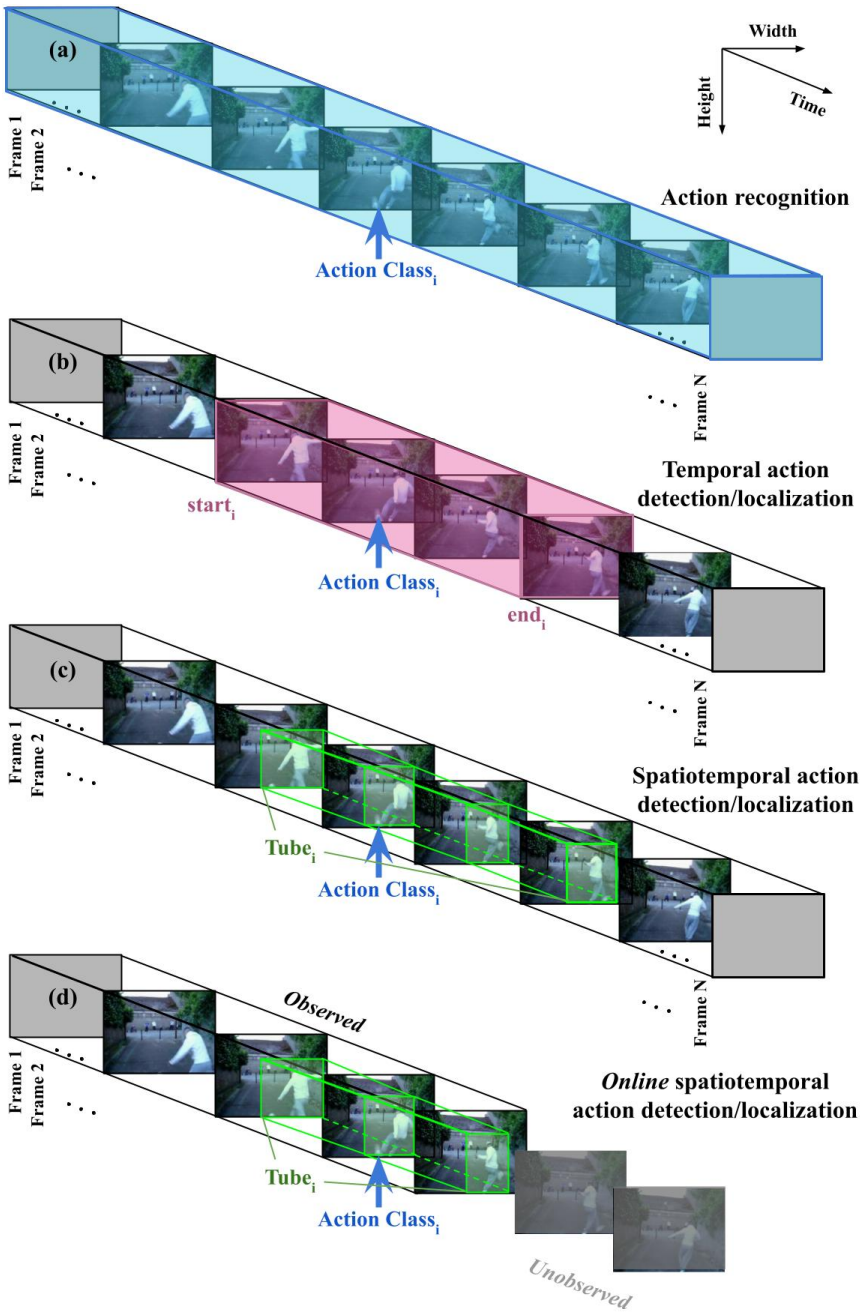

FIGURE 1.1: An overview of the main action understanding problems. A video is depicted as a 3D volume where N frames are stacked along the temporal axis. Action recognition assigns an action class label to the entirety of the input video (a). Temporal action detection/localization predicts the action-specific temporal regions bounded by the start and end frames (b). Spatiotemporal action detection/localization proposes action-specific tubes formed by bounding boxes linked across frames (c). Its online variant concerns predicting the bounding boxes and label of an action tube only in the observed part of the video (d). In the detection/localization problems, multiple co-occurring action instances can be present (each captured by a temporal region or action tube).

1.2.2 Why targeting online spatiotemporal action detection?

Our incentives to tackle online spatiotemporal action detection are three-fold. First, we consider action-tube detection in space and time a highly comprehensive action understanding problem. To produce co-occurring action tubes in a video involves several related tasks such as tracking of each actor, action recognition, and space-time tube proposal, etc. As opposed to providing only video-level labels, we believe that orienting toward an instance-based solution requires a much more thorough understanding of the video content that can benefit more application scenarios.

Second, we are convinced that the identification of actions is a stepping stone for understanding complex human behavior. In this sense, instance-level detection provides means of characterizing each actor ranging from the coordinates, static appearance to dynamic context descriptors. The instance-level information can subsequently serve as metadata for more complicated analysis such as person re-identification or modeling of group activities.

Finally, we target action-tube detection in an online mode where the video is processed *incrementally* for any arriving new frame (the detector could only rely on the present and past observations). Such a style of processing is crucial in real-world scenarios such as human-robot interaction and automated security where video contents are streamed continuously (rather than being stored on disk as complete video files).

1.3 Toward real-time and lightweight detection

In recent years, action detection has received much more attention in the research community as driven by numerous vision and internet-of-things (IoT) application demands, such as automated security systems, service robotics, unmanned aerial vehicle monitoring, autonomous driving, interventional healthcare, and unmanned stores, to name a few. Capacities of modern action-tube detectors have progressed significantly thanks to multiple advancements in related domains: object detection for precisely localizing humans and scene elements; two-stream CNN facilitates appearance-motion modeling on top of raw video frames and optical flow; 3D CNN enables direct spatiotemporal encoding from video volumes; the emergence of large-scale, labelled video datasets, and so on. Despite their promising progresses, existing approaches

have been tailored to obtain superior accuracy in public benchmarks. In return, their heavy architectural designs and detection pipelines typically incur long processing time and high computational cost.

Meanwhile, moving the computation closer to the sensor has become a fundamental system requirement for real-world deployment in order to manage the enormous data flow (i.e., video streams). Such a sensing paradigm shifts dependence from more capable, high-end systems of workstation machines (e.g,. NVIDIA Titan GPUs) to embedded/edge devices under a restricted computing and power budget (e.g., NVIDIA Jetson GPUs). Although embedded systems are more energy efficient, their computation capabilities fall short by an order of magnitude when compared to the server-level hardware with high core counts and memories. For instance, An NVIDIA Jetson TX2 has 12 times fewer CUDA cores and a theoretical slowdown of 9.23 times in total than a Titan X [8]. Inherently, current action-tube detectors fail to meet the realistic application specifications which mandate on-site, real-time and highly efficient interpretation of the scene. Deploying state-of-the-art action detection methods onto power-aware embedded vision systems while retaining accuracy and real-time performance remains a great challenge until this day.

1.4 Thesis overview

Based on the above insights, the core of this thesis is to investigate ultra-efficient CNN architectures and detection pipelines for solving spatiotemporal action detection. Ultimately, we seek to uncover a lightweight, real-time action-tube detection solution that is pertinent to the criteria of realistic applications and their deployment.

The rest of this manuscript begins with a thorough review on related works, followed by four contribution chapters in a chronological order as how the problem has been approached. It finishes upon our summary of contributions & future perspectives. In detail, the remaining chapters are structured as follows:

Chapter 2: Related Work. In this chapter, a literature review is first made to familiarize with two highly-associated tasks: action recognition and spatiotemporal action detection. We first study the advancement of video representation extraction, from hand-crafted descriptors to learned CNN features (two-stream CNN and 3D CNN) within the context of action recognition.

Next, we concentrate on modern progresses of spatiotemporal action detection based on the CNN approach, upon which we identify suitable research paths that meet our global objective of lightweight and high-speed detection. Following literature review, action-related datasets as benchmarks are introduced. Finally, we explain the evaluation protocols that are widely used in the research community and in this manuscript.

Chapter 3: Action detection framework based on flow-guided feature approximation and memory aggregation. This chapter presents our first proposed action detector aiming to enhance detection efficiency by exploiting the temporal coherence of continuous video frames. This is embodied by leveraging motion information to help generate visual features for the majority of frames in the video, mitigating redundant re-extraction of features from nearby frames that are visually similar. Also making use of motion cues, we explore implicit spatiotemporal modeling based on a multi-frame feature aggregation schema which recursively accumulates appearance context from sparsely sampled frames over time.

Chapter 4: Temporal feature exchange-difference network. Instead of inferring actions from a single frame at a time as presented in Chapter 3, in this chapter we propose to concurrently process a series of video frames and predict the underlying action tubelets (sequences of action-specific bounding boxes). Specifically, two lightweight temporal modules based on channel-exchange and spatial-displacement are applied on top of the input sequence's CNN features, aggregating actions' context and model their movement over time, respectively. We further introduce a tubelet linking algorithm to associate detection results in a timely and incremental manner for online action tube generation and spatiotemporal localization.

Chapter 5: Accumulated micro-motion features for real-time spatiotemporal action localization. Unlike the previous two chapters where we attempt to implicitly model actions' dynamics via raw video frames and their features, in this chapter we generate explicit micro-motion representations on-the-fly to facilitate temporal modeling. The proposed method (AMMA) adopts the tubelet detection paradigm introduced in Chapter 4 while partially leveraging a two-stream CNN architecture to fuse appearance and complementary motion cues at multiple scales. Moreover, we demonstrate the generalization ability of our method through successfully integrating with multiple lightweight CNN architectures built for embedded platforms.

Chapter 6: Diving more deeply into AMMA. This chapter extends from

Chapter 4 and 5, providing a more profound inspection to uncover our methods' accuracy, computation complexity, and speed performance. Multiple breakdown analysis on each detector's error rates, class-wise accuracy, computational cost and runtime bottlenecks are conducted and compared.

Chapter 7: Conclusion and Future perspectives. Finally, we summarise our contributions proposed in this thesis and discuss prominent research directions in both short-term and long-term perspectives.

Chapter 2

Related Work

In this chapter, we first review the advancement in action recognition, a highly related video understanding task which has established a concrete foundation for spatiotemporal information extraction from video clips. A brief overview is first presented on methods utilizing hand-crafted features in combination with a classifier. We then move on to prominent works based on learned features derived from designated objective functions, i.e., Convolutional Neural Networks (CNN). Following the above overview, recent progresses of spatiotemporal action detection and related topics are studied.

2.1 Action recognition

2.1.1 Recognition from hand-crafted features

Action recognition has been widely studied and most often tackled as a video classification problem. Traditionally, an action recognition pipeline can be broken down into three phases. In the first phase, local visual cues are extracted from hand-crafted image features and extended temporally into video representations. For instance, Laptev et al. [9] proposed the space-time interest points by extending the 2D Harris corner detector [10] into 3D space. Other local spatiotemporal features such as HOG3D [11], SIFT-3D [12], Extended SURF [13], HOF [14], and dense trajectory features [15] have also been explored. Subsequently, the extracted local features are combined into a fixed-sized video-level descriptor following high-order encodings such as Bag-of-Words [16] or Fisher Vectors [17]. Lastly, a classifier such as a Support Vector Machine (SVM) [10] is trained on the resulting video-level spatiotemporal representations to distinguish among different action categories.

Manual crafting of feature descriptors leaves room for improvement as the process is agnostic to the specific classification task (only the classifier "adapts" to distinguish different actions; no action-specific pattern is learned).

Consequently, traditional methods which are limited by these predefined set of features struggle to fully extract complex, action-relevant information embedded in videos.

2.1.2 Recognition from learned features: 2D CNNs

The advent of deep convolutional architectures [3][18][19] and their impressive performances to hierarchically extract complex, abstract spatial information makes 2D CNNs the dominating paradigm in the image domain (e.g., object classification and detection, semantic segmentation, and image captioning, etc.). Recently, 2D CNNs have also been adopted to tackle the video-based action recognition problem.

The early attempt by Karpathy et al. [20] proposed to fuse multiple frame information over various convolutional configurations. Nevertheless, their different fusion strategies did not lead to significant improvement from neither their single-frame baseline model, nor other top-performing methods based on hand-crafted representations. Such findings suggested 2D CNNs' limited capability to model temporal variations from video frames alone. Loosely inspired by the human visual cortex, Simonyan and Zisserman [21] later devised the two-stream CNN architecture, combining a spatial CNN to learn static visual cues from regular RGB frames, and a temporal CNN to model motion information from stacked optical flows. In such a framework (as illustrated in Figure 2.1), separate training was performed for the two CNNs; the softmax scores of both streams were combined by late-fusion.

A plethora of research have extended from two-stream CNNs. In lieu of applying late-fusion from two independently processed streams, Feichtenhofer et al. [22] investigated intermediate feature-level fusion strategies to jointly model actions' appearance and motion correspondences. Park et al. [23] proposed a multiplicative fusion paradigm to combine two-stream features which demonstrated better performance than the regular addictive fusion (e.g., feature concatenation or averaging CNNs' output scores). In order to model long-range temporal structures, Wang et al. [24] sparsely sampled frames spanning the whole video as inputs. Different segments are fed to the same CNN with shared parameters to predict action independently, followed by a segmental consensus module to aggregate their results. With a similar objective, Donahue et al. [25] and Ng et al. [26] both leverage Recurrent Neural network models (RNN), more specifically, Long Short-Term

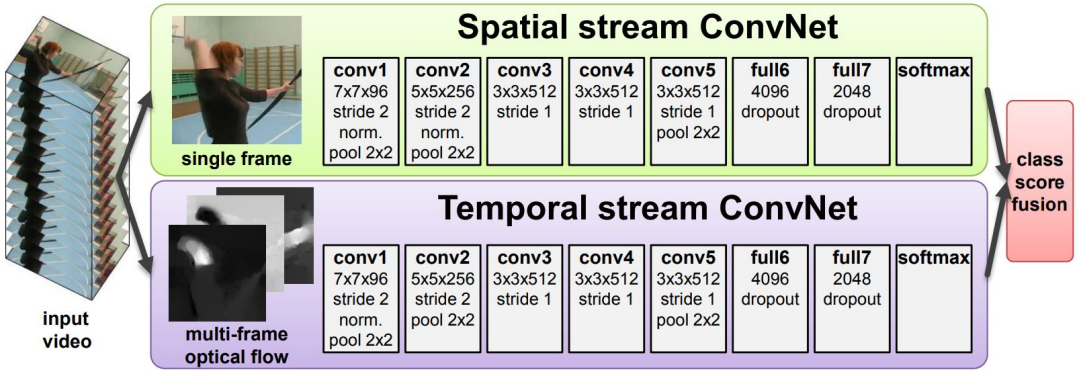

FIGURE 2.1: Two-stream CNN architecture for action recognition [21]. The spatial and temporal stream CNNs separately feed-forward RGB and stacked optical flow input to extract actions' appearance and motion cues. Late-fusion is applied to combine softmax activations of two streams for classifying actions.

Memory (LSTM) [27] RNNs on top of extracted two-stream CNNs features to capture long-temporal range context embedded in video clips.

Even though optical flow is found to be a useful motion representation, calculating optical flow externally remained inefficient and time-consuming (over 90% of the whole runtime [28]). To incorporate motion cues that are simpler to compute, Sun et al. [28] derived optical-flow-like features by taking the spatial and temporal gradients of video frames' feature maps using Sobel filters and element-wise difference operators. Other lightweight motion representation variants such as motion vectors [29], RGB difference [24], dynamic images [30], and feature-level displacement [31], etc. had also been explored to distill temporal information. Another line of works attempted to generate optical flow from CNNs [32][33], which not only exceeded handcrafted flow in terms of speed, but also permitted an end-to-end architecture that could jointly refine motion information and classify actions [34][35].

2.1.3 Recognition from learned features: 3D CNNs

Introducing the motion modality empowers 2D CNNs to incorporate temporal information and obtain competitive recognition accuracy. In the aforementioned methods however, visual cue extraction from video frames is isolated. Temporal modeling is also mostly restricted to late-fusion of abstracted RGB and motion cues. The pixel-level spatiotemporal evolution across successive video frames was not fully exploited.

A video sequence can be seen as a 3D volume of stacked RGB frames. To

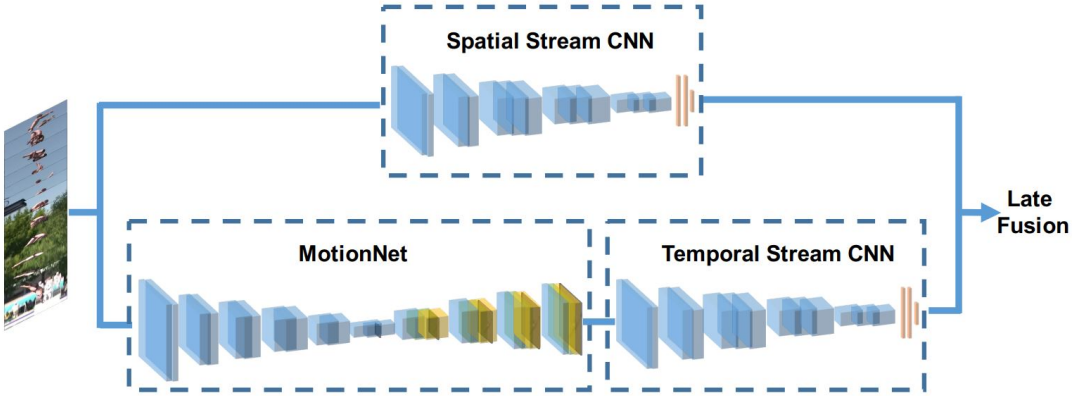

FIGURE 2.2: Illustration of an integrated two-stream CNN by Zhu et al. [35], an exemplary, end-to-end architecture where motion estimation can be jointly modeled and refined along with the designated task (i.e., action recognition). Here, MotionNet takes consecutive video frames as input and estimates motion. The rest of the architecture follows that of a standard two-stream CNN.

simultaneously encode the spatial and temporal pattern from the video volume, Ji et al. [36] and Tran et al. [37] extended the standard 2D convolutional and pooling kernels into 3D. Given a stack of frames as input, 2D convolution treats them as different channels and collapses the temporal structure, while applying 3D convolution results in another volume that preserves temporal information of the input. Nevertheless, early attempts based on 3D CNN architectures (e.g., C3D [37]) received limited performance gain mainly due to the high parameterization as well as the lack of labelled video data to properly generalize spatiotemporal feature learning.

Later, Carreira et al. [38] proposed the Inflated 3D CNN architecture (I3D) which inflated 2D convolutional and pooling kernels, as well as ImageNet pre-trained weights of existing 2D CNNs (i.e., by expanding into the temporal axis). This enabled spatiotemporal modeling on top of well-established ImageNet networks and their spatial feature extractors. Leveraging the two-stream configuration and their large-scale human action dataset Kinetics for pretraining, I3D had proved highly effective, making 3D CNNs the dominant approach for action recognition. Subsequent works such as Feichtenhofer et al. [39], took inspiration from two-stream CNN and proposed a dual 3D-CNN architecture, with each sub-network taking RGB frames of different frame rates as input to model varied extent of spatial and spatiotemporal cues. Wang et al. [40] inserted non-local blocks based on the self-attention mechanism in existing 3D CNN to enable capturing long-range temporal dependency in videos. Very recently, extensive research [41][42][43] have been conducted based the attention-derived transformer architecture [44].

2.1.4 Toward lightweight and efficient action recognition

Although 3D CNNs are suited for learning highly complex video representations, they introduce substantially more parameters, computational burden, and training difficulties than their 2D counterparts. For instance, C3D, which consisted of 11 layers of 3D convolutions, exceeded the model size of a 152-layer 2D-ResNet model ($\approx$ 321 vs. 235MB) [45]. Moreover, 64 and 128 GPUs were used for model training in [38] and [39], respectively.

Numerous works have aimed toward improving the efficiency of 3D CNNs with alternative convolutional blocks. Qui et al. [45] proposed the Pseudo-3D Residual Networks which decomposed a $3 \times 3 \times 3$ 3D convolution into separate spatial ($1 \times 3 \times 3$) and temporal kernels ($3 \times 1 \times 1$) interacted in various forms (e.g., stacking, parallel pathways, and residual connection, etc.). Similarly, Tran et al. [46] factorized 3D convolution into successive 2D spatial and a 1D temporal convolution (referred to as "R(2+1)D") and demonstrated that such sequential decomposition facilitated complex representation learning by doubling the number of nonlinearities.

Aiming also for computation reduction, Xie et al. [47] achieved the best accuracy-speed trade-off by replacing many 3D convolutions with 2D ones at the bottom of the I3D network, suggesting that temporal relation modeling is more effective on high-level semantics. Their resulting architecture (referred to as separable 3D CNN, or S3D) also leveraged the separate spatial-temporal convolutional block. Zolfaghari et al. [48] adopted a similar schema of mixing 2D/3D convolutions along with a sparse sampling strategy [24] for efficient online action recognition over long video sequences. The practice of decoupling 3D convolution had also been explored in the context of group convolutions, endowing two groups of filters to focus on modeling static and dynamic cues separately [49]. Other research such as Feichtenhofer's X3D [50] progressively expands a tiny 2D image classification architecture along multiple network axes (such as temporal duration, frame rate, spatial resolution, and network depth) to monitor different levels of computational complexities and memory usage.

Notably, architectures based on integrating dedicated temporal modules on top of 2D CNNs have received more attention in recent years. This general design aimed to achieve comparable spatiotemporal reasoning and accuracy of 3D CNNs while maintaining the computational complexity at 2D. Lin et al. [51] introduced the Temporal Shift Module (TSM) which partially shifted the feature channels of multiple-frame features along the temporal dimension. Inserting the shift module in hierarchies of 2D convolutional blocks induces

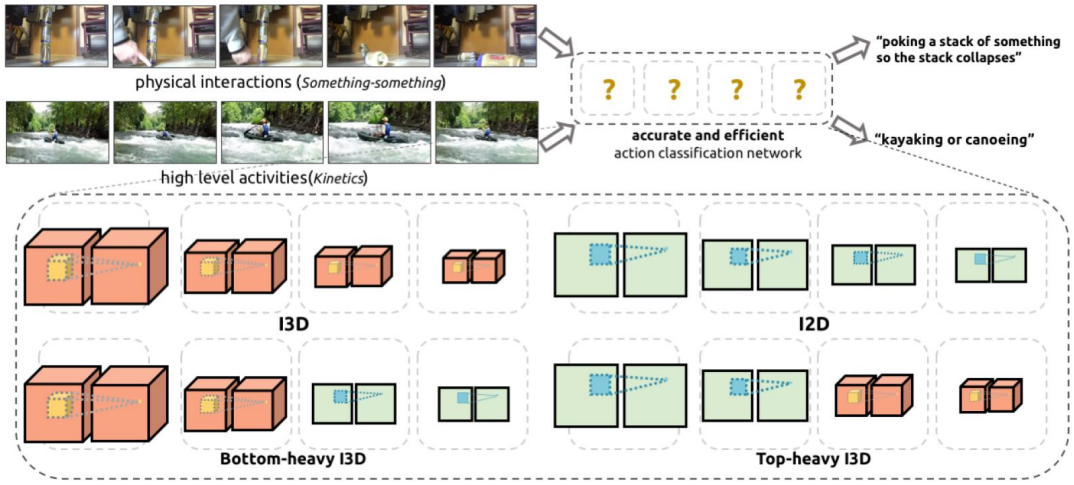

FIGURE 2.3: Illustrations of four architectural variants by Xie et al. [47] to extract spatiotemporal information. I2D (top-right) operates on multiple frames via a shared 2D CNN; I3D (top-left) is a full 3D CNN which convolves over space and time; Bottom-Heavy I3D (bottom-left) uses 3D convolution in the lower layers and 2D in the higher layers; Top-heavy I3D (bottom-right) employs 2D CNN in the lower layers and 3D in the upper ones. Top-heavy I3D achieves the best accuracy-speed trade-off, which implies that temporal relation modeling is more effective on high-level semantics.

feature interactions and forms temporal dependency across video frames. Lee et al. [52] proposed the Motion Feature Network (MFNet), which comprises hierarchies of motion filters to encode motion context between any two successive feature maps by aggregating their feature-level displacement in multiple directions. Building upon TSM, Jiang et al.[53] and Li et al.[54] further relaxed the shift operator into an 1D channel-wise temporal convolution with designated weight initialization, permitting a more adaptive and fully convolutional temporal aggregation schema. Similar to MFNet, efficient motion encoding is performed either via feature-level displacement [53] or motion-derived attention maps to highlight motion-salient regions [54]. Following the success of leveraging feature map displacement over time, Wang et al.[55] devised a two-level difference modeling paradigms to capture both high-resolution local motion pattern over consecutive frames and cross-segment spatiotemporal structures spanning long videos.

2.2 Action detection

Different from action recognition (i.e., video-level classification), spatiotemporal action detection/localization concerns generating space-time proposals to cover individual action instances. The detection pipeline broadly comprises three levels of sub-tasks: 1) frame-wise spatial localization of actors; 2) action classification, i.e., inferring the action category for each individual actor; and 3) temporal linking and trimming, i.e., temporally linking actions' bounding boxes and determining their temporal extent in the videos. Effective spatiotemporal modeling is essential in both action recognition and detection. Transitioning from hand-crafted video features to deep CNN paradigms (e.g., two-stream CNN or 3D CNN) had been prominent in both tasks thanks to CNNs' superiority in modeling complex scenes.

Here, we review recent progresses of spatiotemporal action detection within the context of CNN. Prior to that, a non-exhaustive overview on related object detection techniques is provided, which have been widely employed to spatially localize action instances. In later chapters, we will present more extensive reviews on the works relevant to our contributions.

2.2.1 Spatial localization (object detection)

Localizing human actors in each video frame is a fundamental step for spatiotemporal action detection. Naturally, this sub-task could be achieved using object detection techniques. Earlier attempts to detect salient objects relied on region-proposal algorithms based on local hand-crafted features, such as Selective Search [56] and EdgeBoxes [57]. In the last decade, object detection also benefits from the advancement of deep CNN architectures. Mainstream detectors make use of anchors (i.e., pre-defined proposals) sampled across every feature grid to expedite localizing objects at arbitrary locations in the image. For example, Faster R-CNN [58], R-FCN [59], and FPN [60] leverage a two-stage detection pipeline, first extracting potential regions-of-interest (RoI) from the convolutional Region Proposal Network (RPN). In the second stage, object features are pooled from each proposal region, which are then used to classify and refine the object-specific bounding box. Two-stage detectors achieve state-of-the-art accuracy. However, their sequential pipeline imposes a bottleneck to real-time inference speed.

Alternatively, other detectors such as YOLO [61], SSD [62], and RetinaNet [63] removed the intermediate step of region proposal. In a single forward-pass, they directly perform bounding box regression and classification from

anchors densely sampled across every grid of the image feature. In most scenarios, the number of background samples significantly outweigh that of salient objects during training. To address the imbalance between foreground and background instances, Lin et al. [63] designed a novel focal loss to automatically reduce the contribution of easy examples (i.e., background) and focus on hard examples. One-stage detectors are capable of real-time inference without compromising much accuracy, hence are widely considered when high speed is of priority.

More recently, anchor-free object detectors such as CornerNet [64], CenterNet [65] and FCOS [66] have gained more popularity as they are not bounded by the heuristic anchor design. These detectors generally tackle the detection problem by estimating class-specific "keypoints" in the image feature, upon which they regress objects' spatial sizes.

2.2.2 Spatiotemporal action detection/localization

Single-frame based methods

Early attempts of CNN-based action detection generally extracted potential actor regions in video frames leveraging salient-object proposal algorithms. Each actor proposal would be fed to the CNN for classification; frame-wise detections were then linked over time to construct action tube. For instance, Gkioxari and Malik [67] generated region proposals on each frame using Selective Search and inputted the concatenated appearance-motion features associated to each region to SVM classifiers. Following that, frame-wise detections over all video frames were linked in a timely manner using the Viterbi algorithm [68] to iteratively find optimal paths that maximize temporal coherence, i.e., action regions with high confidence scores were strongly linked if their spatial extent largely overlapped. Similarly, Weinzaepfei et al. [69] applied EdgeBoxes to obtain frame-level proposals and then link them via a tracking-by-detection paradigm. Temporal localization was achieved by a multi-scale sliding window strategy.

The above works followed an expensive multi-stage detection pipeline that consists of proposal generation by external modules, CNN fine-tuning and feature extraction for each proposal, caching of these features on disk, and action classification by SVMs. To overcome such inefficiency, Saha et al. [70] adopted Faster R-CNN to detect class-specific action instances in an end-to-end fashion. Their proposed pipeline included two parallel RPNs and detection networks to take into account both RGB and motion cues. Action

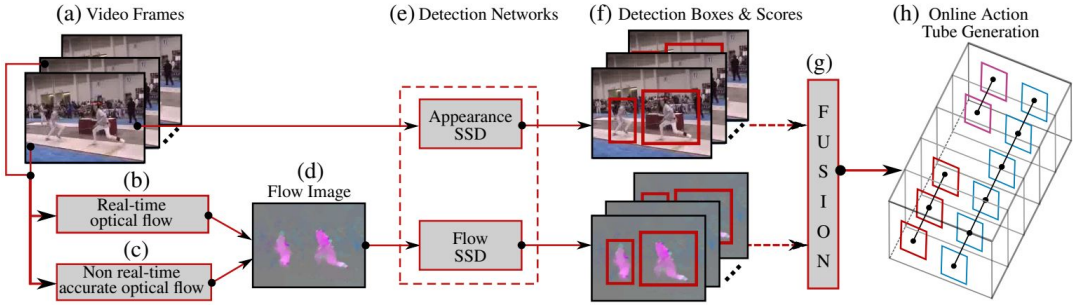

FIGURE 2.4: Illustration of the complete action detection pipeline (single-frame based approach) by Singh et al.[73].

tube construction was formulated into two energy maximization problems, ensuring optimal temporal coherence when linking frame-wise detections and smoothing the resulting links. Peng and Schmid [71] adopted a similar detection pipeline as in [70] while further introducing complementary information from stacked optical flow and human body parts. Further exploiting the sequential information across adjacent video frames to render more precise localization, Yang et al. [72] proposed a location anticipation network for inferring movements of action instances.

The advent of unified CNN object detectors greatly simplified the workflow of action detection; however, the two-stage detection scheme, optical flow computation, and two-pass tube generation algorithms over the entire video made the above methods intrinsically slow and offline. Aiming at real-world applications, Singh et al. [73] proposed the first real-time action detector, which leveraged the more recent SSD to acquire frame-wise action instances, a fast flow estimator to obtain motion cues, and an incremental tube linking & trimming algorithm to construct action tubes. Figure 2.4 summarized the workflow their detector. Behl et al. [74] extended the work in [73] by formulating a unified cost function to jointly solve the sub-tasks of linking, action labeling and temporal trimming.

Short-clip based methods

Temporal reasoning is constrained to fusing optical flow cues in the single-frame based approach. Distinguishing actions in such a way can lead to ambiguity, as the appearance evolution and temporal continuity in successive frames are not exploited. Kalogeiton et al. [75] surpassed the above limitation by temporally expanding the SSD and its 2D-anchor framework for 3D anchor cuboid regression. As depicted in Figure 2.5, their detector maps a series of consecutive video frames into a high-dimensional latent space upon

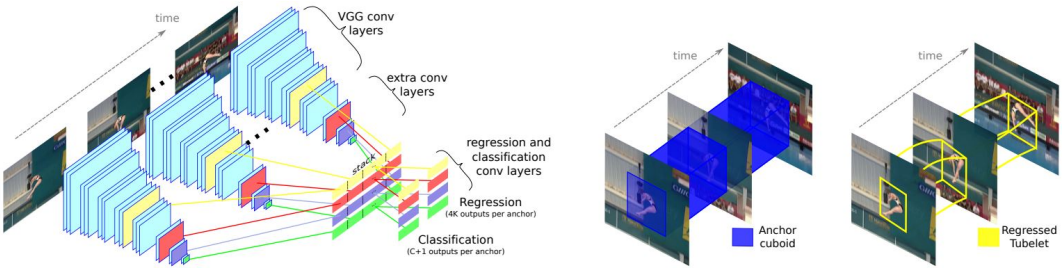

FIGURE 2.5: Illustration of the action tubelet detector by Kalogeiton et al. [75]. Given a sequence of frames, 2D convolutional features are extracted and stacked (left) to predict action scores and regress coordinates for the anchor cuboids (middle). The final outputs are regressed action tubelets (right) spanning the input frames.

which it directly detects action tubelets, a sequence of bounding boxes with associated confidence scores spanning the input frames. Saha et al. [76][77] proposed two similar tubelet frameworks as in [75] while relaxing the constraint of processing consecutive frames. Training and testing can therefore be conducted more flexibly on frame sequences of varied intervals according to the attributes of video data, such as various frame rates or whether densely-annotated groundtruths are available. Huang et al.[78] introduced an encoder-decoder block based upon Convolutional Gated Recurrent Unit [78] prior to the detection network to facilitate fusing the dynamic context across multiple clips.

A number of action tubelet variants have been proposed. For example, the above-mentioned methods first infer class-specific tubelets, and then link them over time into action tubes for each class independently. Alternatively, the works in [79][80][81][82] first processed a short-clip at a time to generate class-agnostic tubelet proposals. Multiple proposals are linked over time into class-agnostic tubes, from which spatiotemporal modeling takes place (such as 3D pooling or LSTM) to categorize each tube. A few other literature focused more on cost-effective detection pipelines. To avoid the heuristic anchor design of SSD, Li et al. [83] adopted the anchor-free CenterNet detector, modeling each action instance in time as a moving point. Aiming at reducing the computational cost associated with having two separate CNNs for visual and motion reasoning, Zhao et al. [84] jointly processed the RGB and optical flow streams in a single two-in-one CNN architecture for extraction of appearance information, while learning motion-conditioned maps to modulate the appearance features.

Clip-based 3D CNN representation for contextual support

In Section 2.1.3, we learned that 3D CNNs are able to effectively capture space-time information. Consequently, leveraging 3D CNNs to encode video representations have become the dominant approach for action detection in recent years. For example, Gu et al. [4] employed Faster R-CNN to first obtain actor proposals on the target frame. The authors then applied I3D on top of surrounding video sequence (e.g., 50 frames) near the target frame, aggregating action-specific video context for refining and classifying each proposal in the target frame. Kopuklu et al. [85] fuses 2D-Darknet and 3D-ResNet's CNN features extracted from the respective target frame and its neighboring video clip as context-augmented representations for frame-wise bounding box regression and classification. Li et al. [86][87] proposed two sparse-to-dense detection paradigms based on 3D CNN, first estimating coarse spatiotemporal action tubes spanning long video sequences, and then refining the tubes' precise locations from key timestamps. The adoption of the transformer architecture for modeling over long sequences also gains more attention in recent action detection research [88][89].

Lately, the AVA dataset [4] was released and contained challenging actions that required advanced contextual reasoning (e.g., interactions between actors and other actors/objects in the scene). To differentiate similar actions, spatiotemporal modeling in terms of relations between actors and scene elements have been investigated, such as global-context fusion [90], graph convolutional network [91], and non-local attention [92], etc. For instance, Sun et al. [90] treated every spatial grid of the video-level S3D feature as a weakly-supervised scene element. Specifically, they modeled pair-wise relations between any designated actor and the rest of scene elements via neural networks, generating relation features to enhance action classification. Wu et al. [92] focused on finding actor-actor interactions over a long temporal window. The authors stored actor features from the past in a long-term feature bank, which can be retrieved to augment present actors' features via feature pooling or non-local attention. Despite its outstanding spatiotemporal modeling and accuracy, this line of method relies on very deep 3D convolutions over long video clips to aggregate contextual information for target-frame detection. In most cases, they also require an additional 2D proposal network to locate potential actor regions in advance [4][90][92].

2.3 Related datasets

Labelled datasets provide means to train accurate and robust deep learning models. In addition, they serve as suitable benchmarks for a direct comparison between different methods, leading to better understanding of the algorithmic advantages and limitations. In this section, we describe some of the most widely used datasets in the action detection community.

2.3.1 Overview on action detection datasets

Datasets for spatiotemporal action detection require action tube annotations. Specifically, an arbitrary action tube consists of the underlying action category and frame-wise bounding box coordinates of the action instance. Inherently, each action instance's temporal extent can be inferred from the frame indices containing bounding boxes.

There exists a number of datasets that have been actively used in the action detection community. To name a few, UCF-24 [5] was produced for the THUMOS-2013 [93] challenge and contained mostly sports actions in untrimmed videos. JHMDB-21 [6] was initially created for pose-based action recognition. It primarily includes instant actions such as sitting, standing, and waving, etc. from short video clips. DALY [94], as the name suggests, focuses on daily activities such as applying make-up on lips, brushing teeth, etc. Upper body pose and bounding box annotations around object(s) taken part in actions are also provided. Recently, more complex architectures have pushed the requirements for large-scale video datasets such as Atomic Visual Actions (AVA) [4]. The AVA dataset annotates 80 atomic actions sourced from movie clips, where every person is localized and attached multiple labels corresponding to the actor's pose, interactions with objects and other actors.

2.3.2 Datasets used in this thesis

UCF-24 contains 3207 videos and 24 human action classes of sports-related action categories, such as Basketball, Fencing, IceDancing, and VolleyballSpiking, etc. Released for the THUMOS-2013 challenge, this dataset is a spatiotemporally labelled subset of UCF101 [5], a diversified action recognition dataset with 101 action classes and videos sourced from YouTube. In average, each video in UCF-24 lasts approximately 7 seconds (acquired at a fixed frame rate of 25). Note that the original annotations released by [5] was found

prone to errors, and corrected annotations were later released by Singh et al. [73], which are used throughout this thesis.

Even though every video is associated with only one action class, it may contain multiple action instances with different spatial and temporal boundaries. Here, a video is likely to contain multiple action instances in a couple of scenarios. First, there exist multiple actors in the scene. Second, a single action may be interrupted (e.g., actor being out of frame due to camera movement, or an action is repeated interleaved with pauses). In average, there are approximately 1.4 action instances per video, each action instance covering 70% of the duration of the video. In some classes, an instance's average duration can be as low as 30%. More detailed statistics of individual classes will be presented in Chapter 6. The untrimmed nature of UCF-24 allows us to evaluate temporal localization of our proposed methods.

JHMDB-21 is a relatively smaller dataset, consisting of 928 videos and 21 action classes. It is a subset of the HMDB-51 dataset [95]. Compared to UCF-24, JHMDB-21 is made up of shorter video clips (40 frames at maximum). Its actions are less associated with sports and instead include various classes that are more instantaneous (i.e., lasting only a few frames) and independent from the visual scene, such as Sit, Stand, and Walk, etc. Each video is trimmed to the action's duration and contains only a single action instance. In addition to bounding box annotation, 2D joint masks and human-background segmentation are also provided in this dataset but are not exploited in this thesis. JHMDB-21 is divided into three train-test splits; our experimental results are reported over the mean of all three splits.

2.4 Evaluation metrics

In this section, we briefly describe some standard evaluation metrics that are commonly adopted for assessing spatiotemporal action localization.

2.4.1 Frame-level mean Average Precision (frame-mAP)

Mean Average Precision (mAP) has been commonly adopted in measuring how an object detector performs. The frame-mAP for action detection is calculated similarly. To briefly recap, AP_c corresponds to the area under the precision-recall curve produced by the model (c denotes a designated class). The area can be easily approximated by the 11-point interpolation algorithm [96]. Precision is measured as $\frac{TP}{TP+FP}$, where TP and FP denote true-positive

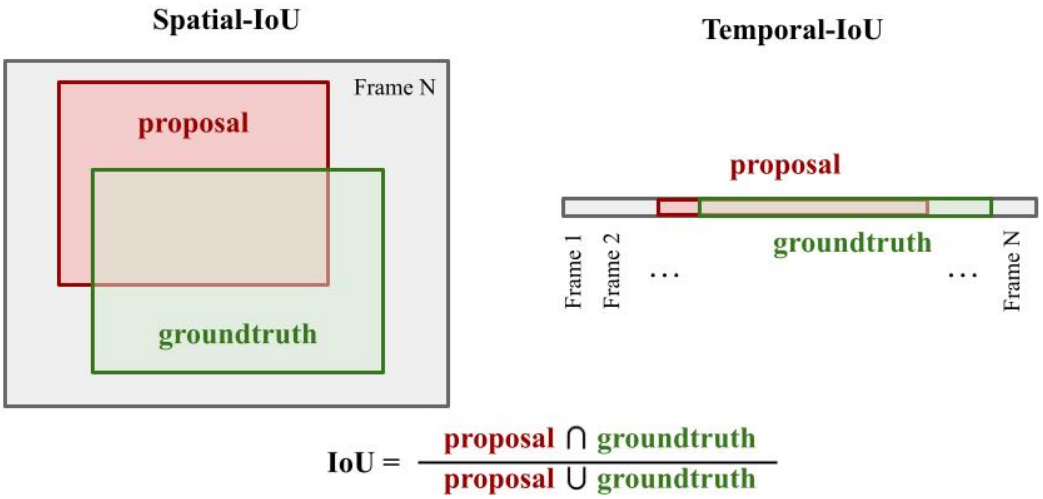

FIGURE 2.6: Illustrations of two types of intersection-over-union (IoU). The spatial-IoU (left) is used to compute frame-mAP, while both IoUs are used in video-mAP.

and false-positive detection, respectively. In the context of a detection problem, precision reflects how reliable the model's detection results are. On the other hand, the recall is computed as $\frac{TP}{TP+FN}$ which measures the model's ability to detect positive samples. The two metrics provide different perspectives; a detector with high-precision but low-recall misses detecting many objects in the scene, while a detector with high-recall but low-precision simply predicts many unreliable detections.

In detail, to calculate frame-AP_c, all the model's outputs of class c are ranked and assessed in decreasing confidence. The prediction at a particular rank is a true-positive detection if 1) there exists a groundtruth action instance of the same class in the same frame, 2) the predicted and groundtruth instances have a spatial overlap exceeding the designated threshold τ (typically at 0.5), and 3) the matched groundtruth had not been associated previously. Otherwise, the prediction is considered false-positive. Specifically, the extent of spatial overlap is determined by the intersection-over-union (IoU) as illustrated in Figure 2.6 (left).

At each rank, precision and recall can be derived from the calculated true-positive and false-positive metrics. The points to reconstruct the precision-recall curve can thus be obtained progressively as one iterates through the list of detection by the model. Once AP_c of each class is acquired independently, the frame-mAP metric is simply the arithmetic mean of them all.

2.4.2 Video-level mean Average Precision (video-mAP)

Video-mAP evaluates the model's space-time action proposals against groundtruth action tubes. The metric takes into account both frame-level prediction and the linking strategy which connects bounding boxes of the same class label in adjacent frames. Similar to the computation of frame-AP_c, once bounding boxes of class c are linked into tubes, every predicted tube of that class is ranked in decreasing confidence for computing video-AP_c. At any particular rank, a predicted tube is true-positive if there exists a groundtruth action tube of the same class against a spatiotemporal overlap threshold τ, and that groundtruth has not been associated previously. Otherwise, the prediction is considered false-negative.

Specially, the extent of overlap between predicted and groundtruth tubes is defined by spatiotemporal-IoU (ST-IoU). The ST-IoU is composed of the spatial-IoU and temporal-IoU computed separately. The spatial-IoU is calculated as the mean IoU between the groundtruth and detected bounding boxes of an action instance across their overlapped frames (the computation is the same as that for frame-AP). On the other hand, the temporal-IoU only concerns the frame-level temporal overlap between the groundtruth and predicted tubes. Both spatial and temporal IoUs are depicted in Figure 2.6.

Lastly, the product of spatial and temporal-IoU is taken to obtain the measure of ST-IoU. The final video-mAP metric is simply the arithmetic mean of video-AP_c of all action classes. In a typical evaluation setup, multiple overlapping thresholds (τ) are considered in video-mAP to inspect varied qualities of resulting action proposals. A common practice is to evaluate video-mAP at $\tau = 0.2, 0.5$, and 0.75. In addition, τ ranging from 0.5 to 0.95 (with an increment of 0.05) are examined and averaged as a measure of the global-scale performance.

2.4.3 Model efficiency

Our thesis focuses on well-balanced, efficient detection architectures that can be potentially deployed on power-restricted devices. As a result, measures of models' efficiency are needed. Specifically, we evaluate the speed performance (millisecond/frame or frame-per-second) of our proposed detectors in later chapters. Models' complexity is also measured by the number of trainable parameters and multiply–accumulate operations (MACs) at test time.

2.5 Recap on our research directions

For starters, the single-frame detection pipeline proposed by Singh et al. [73] achieved online and real-time action detection, conforming to realistic use cases. However, optical flow is the only means of temporal reasoning under such an approach, as frame features are extracted independently throughout the entire video. The temporal coherence embedded in continuous videos was not exploited. In Chapter 3, we will explore utilizing the temporal coherence among nearby video frames to guide the generation of new frame features, not only enhancing detection efficiency, but also enabling current observations to recursively accumulate precedent context for long-range temporal modeling.

Further, despite modeling actions from video clips which theoretically should also capture short-term dynamic cues, many tubelet detectors [75][77] [83][97] still rely on computationally-expensive optical flow to enhance accuracy. This implies insufficient temporal reasoning from stacking frame features alone. Alternatively, we present an adaptive, multi-frame feature aggregation schema based on partial channel-wise exchange, while exploiting dynamic information to trace moving actors in Chapter 4. Partially building upon this work, we devise an on-the-fly motion representation in Chapter 5 and 6 that accumulates motion boundaries as complementary action context, giving rise to a lightweight, real-time action detector with highly competitive accuracy.

On the other hand, as it hinders our objective of achieving lightweight and online action detection for power-constrained devices, the clip-based 3D CNN approach is not considered within the scope of this thesis.

To recapitulate, the theme of our thesis focuses on seeking lightweight architectures along with cost-effective pipeline for real-time (yet competitively accurate) action detection. To validate our reduced spatiotemporal models, we primarily evaluate on the UCF-24 and JHMDB-21 dataset. These two datasets, which will be detailed in the following sections, have been among the most representative and foundational benchmarks for assessing untrimmed and trimmed action videos since the beginning of this thesis.

Chapter 3

ACDnet: Action detection framework based on flow-guided feature approximation and memory aggregation

3.1 Introduction

The task of action detection explicitly addresses space-time localization of action instances in videos. In reality, a video can comprise a single or multiple instances, of the same or different actions, each can even begin or end independently. Rather than reasoning the global video-level action category, the most fundamental building block of a spatiotemporal action detector concerns modeling action-specific pattern from videos and acquiring frame-wise localization (i.e., bounding boxes) for the underlying action instances.

In this chapter, we tackle the action detection problem by building upon the latest CNN-based detectors equipped with spatial localization ability. Recently, advances in object detection have led to a number of state-of-the-art detectors such as Faster R-CNN [58], YOLO [61] and SSD [62], all of which can be potentially integrated within an action detection pipeline. Naively adopting image-based object detectors in the video domain is inherently insufficient to capture discriminitive video representations for actions. Instead, common practices for modeling action-specific representations leverage two-stream CNN or 3D CNN, as introduced in Chapter 2. However, the above inevitably raise computational requirements (time-consumption and computational cost) associated with optical flow computation or 3D convolutional operations, which we argue are sub-optimal (and even unfeasible) for practical deployment on low-end devices.

Most notably, consecutive video frames contain continuous and highly resembling appearance information; extracting frame-wise features without taking into account the intra-frame continuity and similarity incurs significant redundancy. With the above insight, we propose ACDnet, a real-time action detector which exploits the temporal coherence among nearby video frames to enhance detection efficiency. This is embodied by performing feature approximation at the majority of frames in a video, mitigating re-extraction of similar features for neighboring frames. Furthermore, we hypothesize that a less expensive framework can effectively extract meaningful temporal contexts. ACDnet adopts a multi-frame feature aggregation module, which recursively accumulates 2D spatial features over time to encapsulate long temporal cues. Such feature aggregation implicitly models temporal variations of actions and facilitates understanding degenerated frames with limited visual cues.

The proposed action detector leverages SSD to handle frame-level feature extraction and detection. It also employs a lightweight variant of FlowNet [32] to quickly generate optical flow for the feature approximation and memory aggregation modules. Integration of all the above CNN sub-networks formulates an efficient detection solution that is capable of accumulating spatiotemporal context over time and inferring actions well beyond real-time speed.

Related publication: The work presented in this chapter is published in Pattern Recognition Letters, 2020 [98]. The primary investigator in [98] is also the author of this manuscript.

Outline. The rest of the chapter is organized as follows. In Section 3.2, we first briefly review SSD, which consists the core feature extraction and detection block of ACDnet. Next, we present technical details of the proposed flow-guided detection framework, including the feature approximation and memory aggregation sub-modules, and how they are integrated with SSD in Section 3.2. We report various experimental validation of the proposed detector in Section 3.3. Finally in Section 3.4, the chapter is concluded by a summary of ACDnet and its limitations.

3.2 Review on SSD (Single Shot MultiBox Detector)

SSD performs object detection by enumerating over a collection of pre-defined proposals (i.e., anchor boxes) densely sampled at every image feature location. These anchors are defined with different scales and aspect ratios, serving to capture presence of all possible objects in the scene. Unlike earlier works based on a two-stage detection pipeline [58][59], SSD does not depend on the intermediate region proposal step to first sample potential object regions, therefore enabling significant improvement in speed performance.

To extract image features, the default SSD employs VGG16 [18] while discarding its fully connected layers. To handle objects of various scales, SSD leverages multiple feature maps of decreasing spatial dimensions to detect objects independently. This is achieved by appending 6 auxiliary layers (composing of 1×1 and 3×3 convolution layers) after VGG16. Among these feature maps of various spatial scales, i.e., (38×38), (19×19), (10×10), (5×5), (3×3), and (1×1), the largest one is used to locate objects of the smallest spatial scale whereas the smallest corresponds to objects of the largest scale.

After extracting the feature maps, SSD applies two separate 3×3 convolutional filters to conduct classification and regression predictions on every grid of each feature map independently. The classification branch outputs $M \times (C + 1)$ class confidence scores for M anchor boxes and $C + 1$ action classes (one of which is added to represent the *background* class). The regression head outputs $M \times 4$ coordinate offsets with respect to M anchor boxes, which can then be converted to actual bounding-box coordinates of objects. At test time, due to the nature of dense anchor design and placement, SSD tends to detect the same object multiple times from overlapping anchor boxes. To address this, non-maximal suppression (NMS) is applied to recursively remove less confident detection which largely overlap with those having higher confidence scores.

3.3 Overview: flow-guided detection framework

The proposed ACDnet which consists of the feature approximation and aggregation modules, is summarized in Figure 3.1. Our objective is to perform detection in an online manner for every incoming frame of a video. At the beginning of the video, action features are obtained from the first frame by the feature extraction sub-network N_{feat} (Figure 3.1 (a)). This initial frame

is also labeled as the key frame. Following the key frame are consecutive non-key frames, whose action features are not extracted from N_{feat}; instead, the flow sub-network N_{flow} estimates a pair of flow field and position-wise scale map between the non-key frame and its preceding key frame. The resulted flow field is used to propagate appearance feature of the preceding key frame to the current timestamp, which is then refined by the scale map via element-wise multiplication (Figure 3.1 (b)).

As new frames continue to arrive in a streaming video, we sample a new key frame for every temporal stride $T_{mem \to k}$. Two steps are taken at key frames (except for the initial one). First, appearance features are extracted by N_{feat}. They are then fused with those from the past key frames (memory features) via N_{flow} and the aggregation sub-network N_{aggr} (Fig. 3.1 (c)). Note that regardless of the way action appearances are obtained (extracted by N_{feat}, warped via motion, or aggregated in time), the resulting features will be fed to the detection sub-network N_{det} to obtain action bounding boxes and class-specific confidence scores. The fused feature is not only used for detection but also will be cached as the updated memory for the subsequent key frame.

Further technical details and demonstrations of ACDnet are elaborated in the following sections.

3.3.1 Feature approximation by motion guidance

In a video, the appearance content varies slowly over consecutive frames. This phenomenon is even more prominently reflected in the corresponding CNN feature maps which capture high-level semantics. Intuitively, the shared appearances among neighboring frames can help to propagate essential information for a given task. The practice of propagation by Zhu et al. [99] has established success to enhance object detection efficiency in videos, which motivates our feature approximation module.

Within our feature approximation paradigm, the heavier feature extraction sub-network N_{feat} only operates on a coarse set of key frames during inference. The features of successive non-key frames are obtained by spatially transforming those from their preceding key frames via two-channel flow fields. The workflow can be summarized by the following equations. Let $M_{i \to k}$ be the two-channel flow field capturing relative motion (horizontal and vertical direction) from the current frame I_i to its previous key frame I_k.

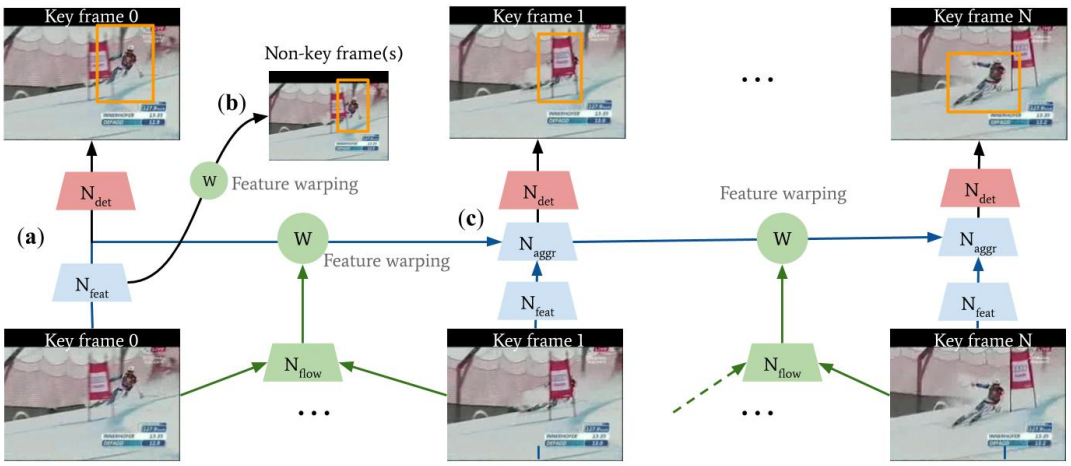

FIGURE 3.1: Illustration of ACDnet's inference pipeline. (a) At the **initial frame**, features are obtained from the feature extraction sub-network (N_{feat}). (b) For **non-key frames** (dense), the flow sub-network (N_{flow}) estimates a pair of flow field and position-wise scale map between any non-key frame and its preceding key frame. The resulted flow field is used to propagate appearance feature, which is then refined by the scale map via element-wise multiplication. (c) At **key frames** (sparse), new features are extracted. They are then aggregated with those from the past key frames (memory features) via N_{flow} and the aggregation sub-network (N_{aggr}). The fused features will be used for detection (N_{det}) and also passed along as the updated memory.

Then, feature approximation (also referred as feature propagation) is realized according to inverse warping:

$$F_i = W(F_k, M_{i \to k}), \tag{3.1}$$

where F_k is the key frame feature, and F_i is the newly warped feature corresponding to I_i. Here, W denotes the inverse warping operation to sample the correct key frame features and assign them to the warped ones. Inverse warping is necessary to ensure every location p at the warped feature can be projected back to a point $p + \Delta p$ at the key frame feature, where $\Delta p = M_{i \to k}(p)$. Concretely, the inverse warping operation W is performed as:

$$f_i^c(p_i) = \sum_{p_k} G(p_k, p_i + \Delta p) f_k^c(p_k). \tag{3.2}$$

In Equation 3.2, f_i^c and f_k^c denote the c^{th} channel of feature F_i and F_k, respectively; G denotes the bilinear interpolation kernel. Every location p_i in the warped feature map undergoes this warping scheme to sample features from key frames for each feature channel c independently. The warping operation is much lighter compared to layers of convolution resided in N_{feat}.

Consequently, by applying feature approximation on the dense set of non-key frames, computation is greatly reduced.

Previous works on action-related tasks mostly acquire motion information from traditional optical flow estimation methods [100]. These methods are generally time-consuming; computing optical flow in such a way inherently imposes a major speed bottleneck. On the other hand, pre-computing optical flow prohibits online detection and incurs additional storage space. Different from precedent approaches, ACDnet integrates a fast flow estimation sub-network, N_{flow}, to predict flow fields. In our case, optical flow serves to spatially transform CNN features; it does not need to capture fine-grained motion details and has the same spatial resolution as the target feature to be warped. Using such a learning-based flow estimator also allows it to be jointly trained with all other sub-networks specific to the task of action detection.

In the proposed detector, the flow sub-network takes a pair of frames (I_k, I_i) as input and generates a pair of motion field and position-wise scale map. Given that H, W, and C denote height, width and channel of F_k, the produced flow field $M_{i\to k}$ has dimension $H \times W \times 2$ (encoding horizontal and vertical offsets); the scale map has dimension $H \times W \times C$. Both motion-related tensors have spatial dimensions matching that of F_k to be warped.

After the inverse warping described by Equation 3.1, the warped feature F_i is refined by multiplying the scale map in an element-wise fashion. Any F_k and F_i would be fed to the shared detection sub-network, N_{det}, to regress action-specific bounding boxes. The above workflow is illustrated in Figure 3.1 (a) and (b).

3.3.2 Memory feature aggregation

Propagating features across frames in time reduces the computational cost associated with bottom-up feature extraction. However, since most features are now approximated, they are heavily dependent on the quality of the precedent key frame features. Moreover, the temporal receptive field of any frame's feature remains as 1, which lacks temporal cues to distinguish actions. To address both aforementioned limitations, we propose the memory aggregation module as inspired by [101] to encode video representation by accumulating multi-frame context.

Given incoming video frames, the core of memory aggregation is to reinforce features of the target frame by recursively incorporating discriminating

context from the past. This allows implicit spatiotemporal modeling without explicitly extracting motion features. In addition, in cases when the current frame is deteriorated, an action can still be inferred with the supportive visual cues from memory. Figure 3.1 (c) gives an example when such memory aggregation could be useful (when the actor is temporally not visible in the scene). In practice, memory aggregation shares the same warping operation used for feature approximation. Specifically, ACDnet takes a sparse and recursive approach to aggregate memory features only at key frames, due to similar appearances shared among nearby frames. Given two succeeding key frames I_{k1} and I_{k2}, where I_{k2} is the more recent one in time, memory aggregation is depicted in Equation 3.3:

$$F_{k2\_aggregated} = w_{k1} \otimes F'_{k1} + w_{k2} \otimes F_{k2}, \tag{3.3}$$

where $F'_{k1} = W(F_{k1}, M_{k2 \to k1})$ corresponds to the warped feature of I_{k1} to spatially align its position with that of I_{k2} according to their relative motion. The position-wise weights w_{k1} and w_{k2} both have the same height and width as F'_{k1} and F_{k2} ($\otimes$ denotes element-wise multiplication). These weights are normalized and determine the importance of memory feature (F'_{k1}) at each location p with respect to the target frame feature map (F_{k2}), i.e., $w_{k1}(p) + w_{k2}(p) = 1$. In our design, w_{k1} and w_{k2} are shared by all the channels of their respective features.

In detail, the weights w_{k1} and w_{k2} are adaptively calculated based on the similarity of memory and target features. We estimate feature similarity by first projecting them into an embedding space via convolutional layers, and then computing the cosine similarity between the embedded features. Finally at the current key frame, the weighted sum of the memory and current features will be fed to the detection sub-network and also passed along as the new memory.

3.3.3 Training ACDnet

ACDnet follows a three-frame training scheme as depicted in Figure 3.2. From each training mini-batch, frame I_i and two precedent video frames (I_k and I_{mem}) are sampled. The features of I_k and I_{mem} simulate the key frame and memory features, respectively. To select three frames for training, we randomize the offset between I_i and I_k between 0 and $T_{k \to i}$ while fixing the offset between I_k and I_{mem} at $T_{mem \to k}$.

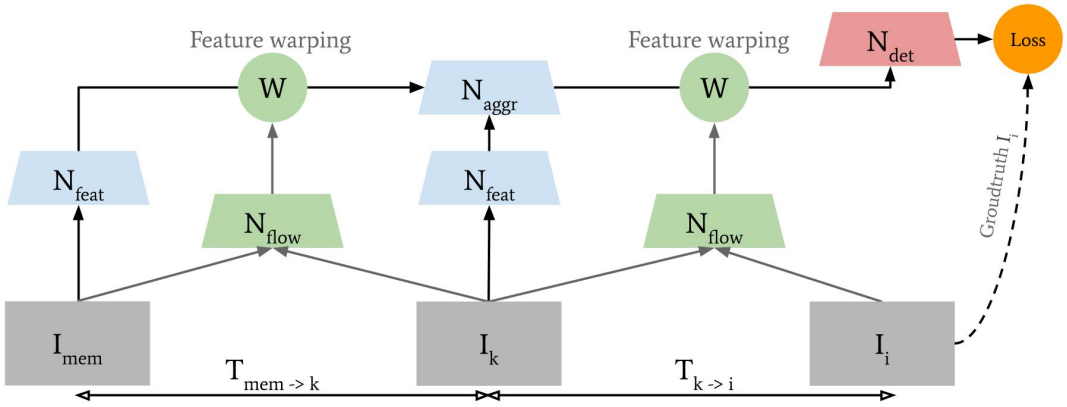

FIGURE 3.2: ACDnet's training procedure. Each mini-batch consists of three frames (I_{mem}, I_k, and I_i) and the groudtruth of I_i.

Both feature maps F_{mem} and F_k are first extracted from I_{mem} and I_k, respectively by N_{feat}. Two sets of flow fields, namely, the relative motion between I_k-I_{mem}, and I_i-I_k are estimated using N_{flow}. The former flow is used to propagate F_{mem} to F_k to simulate the occurrence of memory feature aggregation following Equation 3.3. Then, the fused feature is warped with the second flow (simulating feature approximation) following Equation 3.1, which will be the target feature map upon which N_{det} conducts action detection. Under this training paradigm, ACDnet leverages information from I_{mem} and I_k while predicting action instances on I_i; in other words, only the groundtruth of I_i is taken into account. The loss incurred by N_{det} is back-propagated to update all sub-networks of ACDnet. Specifically, ACDnet adopts SSD; as a result, its training objective in terms of the classification and regression loss follows that of Liu et al. [62].

3.3.4 Adaptation for multi-scale detection

The feature approximation and aggregation modules are designed to be generic and can be plugged into a wide range of detectors. In our design, ACDnet integrates SSD to fulfill our global objective of high-speed action detection potentially for embedded vision systems. In particular, the SSD300 model is chosen due to its superior inference speed.

As mentioned in Section 3.2, SSD progressively appends a set of auxiliary convolutional layers after the VGG16 base network to learn and extract object representations at multiple scales. The creation of multiple feature maps allows the detector to infer objects of various sizes. Consequently, adopting such a multi-scale framework in ACDnet requires feature approximation and memory aggregation to be handled for features at all scales.

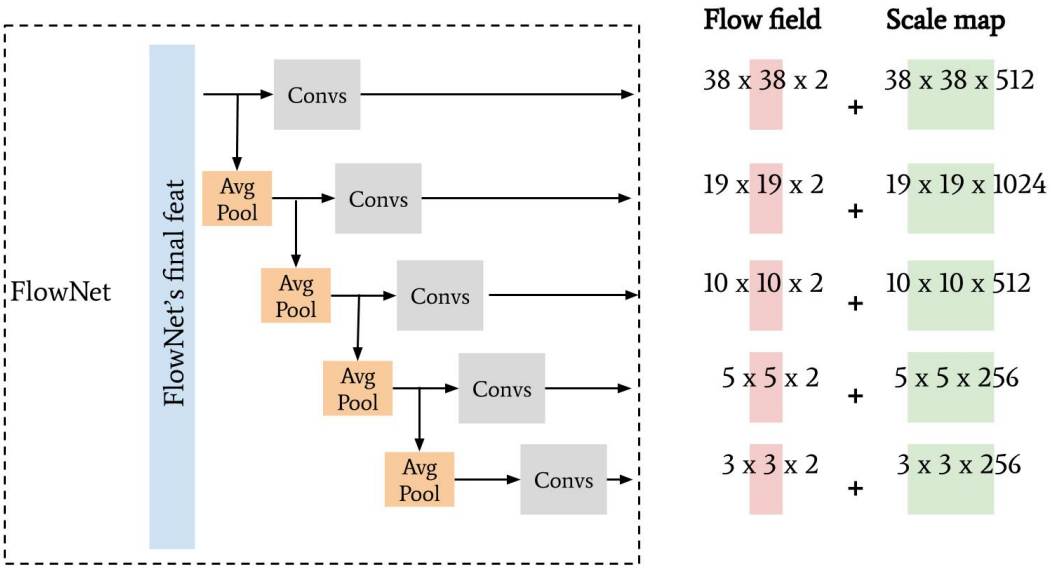

FIGURE 3.3: ACDnet's flow estimation sub-network adapted for multi-scale feature approximation and aggregation.

To carry out multi-level feature approximation, we duplicate N_{flow}'s flow prediction layer into several branches. The number of branches matches that of the feature maps upon which detection will take place. Prior to flow prediction, each branch's feature tensor is also progressively resized via average pooling. Finally, a pair of 2-channel flow field and scale map is reconstructed by every branch, each being used to handle feature approximation and aggregation at a designated scale level (refer to Figure 3.3).

To cope with multi-level feature approximation and aggregation, Equation 3.1 and 3.3 are also generalized to execute at each feature level independently. Note that the standard SSD300 applies detection at six feature scales. However, we only utilize the first five features, as the dimension of the last feature map becomes a 1D vector resulting from progressive resizing, which is no longer feasible for feature approximation governed by 2D spatial warping.

3.4 Experimental validation

The proposed detection framework has been evaluated on **UCF-24** and **JHMDB-21** datasets in terms of accuracy, efficiency, and robustness over several network configurations. The standard frame-level mean Average Precision (frame-mAP) and frame-per-second (FPS) have been used as the evaluation metrics. Specifically, we measure the FPS of the complete detection pipeline, including data loading and model inference using mini-batch size of 1. The

Intersection-over-Union threshold is set to 0.5 throughout all experiments. For brevity, we interchangeably refer to feature approximation and memory feature aggregation as **FA** and **MA** when presenting their results.

3.4.1 Implementation details

Network architectures. ACDnet (implemented in MXNet [102]) incorporates the following sub-networks: SSD300, FlowNet, and feature embedding. Our feature embedding sub-network comprises five branches for measuring feature similarity at five different scales. Each embedding branch has a bottleneck design of three 1×1 convolutional layers interleaved with ReLU non-linearity: $\text{feat}^{l}_{\text{channel}}/2$, $\text{feat}^{l}_{\text{channel}}/2$, and $\text{feat}^{l}_{\text{channel}} \times 2$, where the number of filters varies according to the number of channels at feature level l.

As addressed previously, our FlowNet is modified to also generate five sets of flow fields and position-wise scale maps, each pair being used for warping and refining designated features. We initialize the weights of the first two branches of flow generation layers using FlowNet's pre-trained weights. Considering that the remaining three flow outputs are spatially much smaller than that of the original FlowNet, we randomly initialize the weights of those branches.

Training and evaluation. All input frames are resized to 300×300 for training and inference. ACDnet is trained using the stochastic gradient descent optimizer. To address data imbalance among different actions (due to varied sequence length), from each training video clip of UCF-24, 15 frames spanning the whole video are evenly sampled as the training set. Since video clips of JHMDB-21 are generally short (i.e., maximum of 40 frames), we evenly sample 10 frames from each clip for training. Specifically, both temporal strides $T_{mem \to k}$ and $T_{k \to i}$ are set to 10 during training. These chosen values correspond to the key frame interval used during inference, which is also fixed at 10 in our experiments unless specified.

We apply different hyperparameters on the two datasets. Specifically, UCF-24 is trained for $100K$ iterations; the learning rate is initialized as $5e^{-4}$ and reduced by a factor of 10 after the $80K^{th}$ and $90K^{th}$ iteration. Weights of VGG16's first two convolutional blocks are frozen. For JHMDB-21, due to its smaller training and testing size, we observe that detection accuracy tends to fluctuate significantly between successive epochs. Hence, we empirically train this dataset for $20K$ iterations with learning rate initialized as $4e^{-4}$ and

TABLE 3.1: ACDnet's frame-mAP performances under different architectural configurations.

ACDnet	(a)	(b)	(c)	(d)	(e)
SSD	✓	✓	✓	✓	✓
FA		✓	✓	✓	✓
Scale map			✓		✓
MA				✓	✓
Frame-mAP					
UCF-24	67.32	65.84	67.23	68.06	**70.92**
JHMDB-21	47.90	46.65	46.69	49.37	**49.53**

reduced by a factor of 2 after the $8K^{th}$ and $16K^{th}$ iteration. During its training, the first three convolutional blocks of VGG16 are frozen. In addition, all layers of the modified FlowNet until the five flow generation branches are frozen to further reduce the risk of overfitting.

All sub-networks are trained jointly on an NVIDIA Quadro P6000 GPU using a training mini-batch size of 8. For the rest of hyperparameters and data augmentation setups, we follow the same configurations as those in the original SSD. The weights of VGG16 and FlowNet are pre-trained using ImageNet and the Flying Chair dataset in that order.

3.4.2 Impact of FA and MA

We first assess ACDnet's frame-mAP performance over various architectural configurations, which are reported in Table 3.1. In this experiment, the baseline method (a) consisting of only the stand-alone SSD is first established and compared against models incorporating FA/MA modules (b-d).

From the results of both datasets, we observe a consistent decrease in accuracy ($\sim$ 1.5 frame-AP) when feature approximation (b) is applied to replace the bottom-up per-frame extraction. The accuracy drop can be compensated by the addition of memory aggregation (d, e), which exceeds the accuracy of the stand-alone SSD. Figure 3.4 shows some examples of how the memory aggregation module improves detection by accumulating useful visual cues. Overall, we remark that aggregating multiple-frame features over time, even in a sparse manner, improves models' abilities to more confidently discriminate among different actions. In addition, we examine the effect of having separate branches of position-wise scale maps in N_{flow}, which aim to augment visual features with motion information. Our results (b vs. c and d vs.

FIGURE 3.4: Examples where ACDnet (FA, MA) improves the baseline SSD. Green/Red boxes correspond to correct/incorrect detection, respectively. Memory aggregation helps to accumulate useful visual cues over time and enhance detection consistency. For instance, ACDnet correctly and consistently detects SoccerJuggling in the bottom row despite the fact that strong visual cues pertinent to the action are present only at the beginning.

e) indicate that such refinement mildly improves detection accuracy. As elaborated in Figure 3.5, the scale maps serve as implicit attention maps which reinforce feature responses associated with moving actors.

Notably, even though ACDnet benefits from FA and MA modules as reflected in both datasets, the effect of memory aggregation appears less prominent in JHMDB-21. This could result from the fact that each video clip in JHMDB-21 is much shorter (40 frames or fewer). As MA is performed sparsely at every 10^{th} frame, its impact is limited to 2-3 aggregation per clip. Furthermore, we observe that motions in several JHMDB-21 clips are relatively small, and that key frames far apart still largely resemble. We suspect that in such sequences, memory features do not carry enough contextual variety over time to help distinguish among actions.

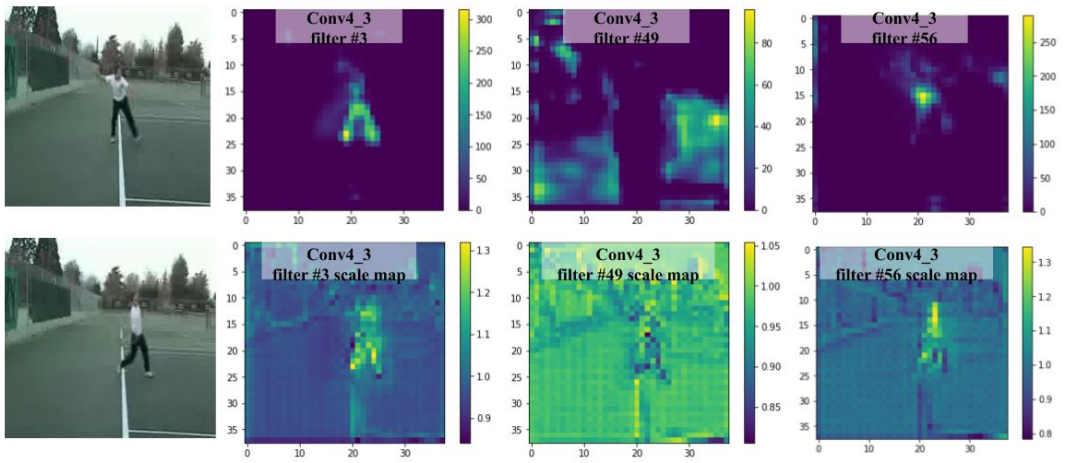

FIGURE 3.5: Position-wise scale maps produced by our modified FlowNet. The scale maps reinforce activation associated with moving actors. For example, filter 3 and 56 (TOP) capture visual cues of the actor, and are up-scaled by corresponding motion scale maps (BOTTOM) which express high responses near moving parts. On the other hand, scale map 49 does not significantly alter filter 49's activation that responds to background cues.

3.4.3 Efficiency analysis

Along with accuracy assessment, ACDnet is further evaluated in terms of its runtime performance and number of parameters under various configurations. Here, we conduct this part of the experiment with UCF-24 and assume the use of scale map refinement when feature warping takes place.

As shown in Table 3.2, ACDnet (SSD, FA, MA) outperforms the stand-alone SSD in both speed and accuracy. This suggests that it is crucial to exploit intra-frame redundancy and that long-range memory fusion is effective for collecting more discriminating features. Regarding the number of required parameters, the increase in ACDnet (SSD, FA) from stand-alone SSD is associated with the addition of FlowNet, which can be replaced by much lighter architectures in the future. Likewise, the increase with the addition of MA module corresponds to the extra embedding layers for measuring feature similarity at multiple scales. In terms of runtime, the speed drop with MA is incurred by the additional operations at key frames (except for the first one), where flow estimation, feature extraction, similarity measure and aggregation all take place.

To examine how our generic architecture performs on a different detection framework, we conduct the same experiments while integrating the R-FCN detector [59] with ACDnet. The results are presented in the bottom part of Table 3.2. Different from SSD, R-FCN is a two-stage detector which makes use of a intermediate region proposal network to first select feature regions

TABLE 3.2: ACDnet's performances under different configurations and detector backbones (on UCF-24).

	Frame-mAP	FPS	# param.
SSD	67.32	70	26.8M
ACDnet (SSD, FA)	67.23	85	50.8M
ACDnet (SSD, FA, MA)	70.92	75	57M
R-FCN	68.2	15	60M
ACDnet (R-FCN, FA)	66.19	34	85.7M
ACDnet (R-FCN, FA, MA)	68.31	32	89.6M

more likely to contain objects. One can observe that even though the runtime performance of R-FCN is significantly lower (15 FPS), the improvement brought by feature approximation is much more evident (34 FPS, which is 2.6 times faster) as compared to that in SSD (1.2 times faster). This is due to R-FCN leveraging the much deeper ResNet101 backbone and larger input frame size (800×600) for detection, incurring more time to extract feature from scratch. Furthermore, the number of additional parameter needed to carry out memory aggregation is less too for R-FCN, as it is designed to perform prediction on a single-scale feature map (i.e., only one branch is needed for the embedding and flow sub-networks). Meanwhile, both detection frameworks obtain similar accuracy under most configurations except when including memory aggregation, where the SSD variant outperforms the R-FCN by 2.61 mAP.

Overall, we perceive a more prominent efficiency gain when the feature approximation and memory aggregation modules are integrated with the computationally more costly R-FCN backbone. However, when taking into account runtime, memory consumption, and accuracy as a whole, our results still strongly favor the SSD-based ACDnet.

3.4.4 Impact of varied temporal strides at train/test time

Concerning the generalization ability of our ACDnet models trained with fixed temporal strides $T_{mem \to k}$ and $T_{k \to i}$ (at 10), we evaluate their performances under various key frame intervals (k) during inference. Such an experimental setup allows us to understand how robustly ACDnet copes with video streams having unexpected movement or varied frame rates. Figure 3.6 shows the frame-mAP results on both datasets where k ranges from 2 to 20.

In the case for UCF-24, both our models (with and without MA) retain their frame-mAP when $k \leq 10$ (reaching maximum at $k = 6$). Beyond $k = 10$,

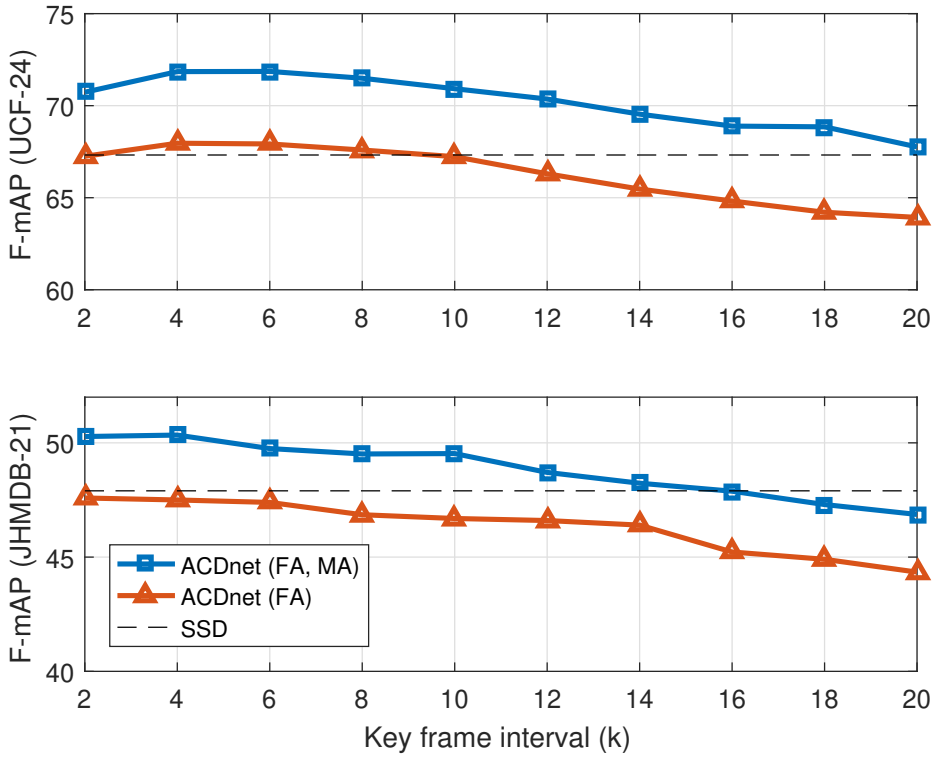

FIGURE 3.6: Robustness evaluation: ACDnet's Frame-mAP under varied key frame intervals. The key frame interval is adjusted only at test time.

we observe a steady drop in accuracy, which can be explained by the fact that flow fields' ability to correctly encode pixel correspondence is compromised under large motions. Similar behaviors can be seen on JHMDB-21. Note that even when k is large, ACDnet with MA still retains decent accuracy which outperforms or is comparable with the best scores of other configurations, reflecting its adaptive capacity toward certain degree of motion variations in videos.

We conduct runtime analysis under the same setting as the above one. The results are shown in Figure 3.7. It can be observed that ACDnet (FA, MA) exceeds the speed of SSD starting around $k = 8$, while the FA-only model is consistently faster. In theory, larger key frame intervals intuitively should only lead to further speed boost, as higher ratio of frame features are approximated. Interestingly, we observe that this pattern is neatly presented when $k \leq 10$. After that, the runtime of the examined models begin to saturate. This phenomenon is associated with two factors. On the one hand, as key frame interval continues to increase, the ratio between the number of key and non-key frames alters more slowly in a video. On the other

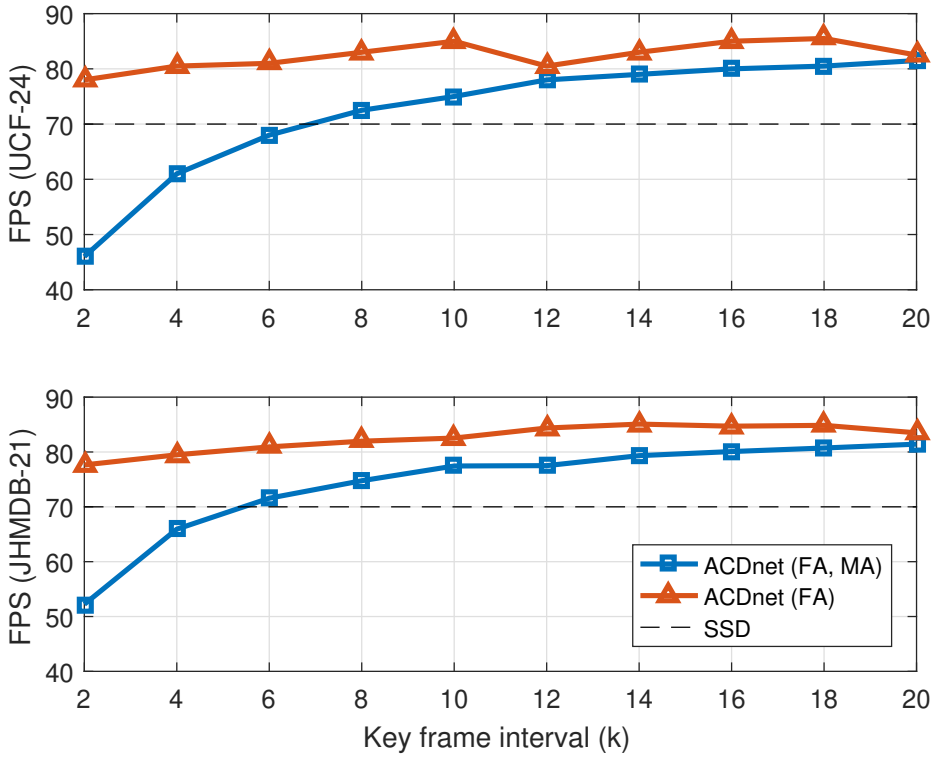

FIGURE 3.7: Robustness evaluation: ACDnet's FPS under varied key frame intervals. The key frame interval is adjusted only at test time.

hand, larger key frame intervals introduce more motion which could compromise the quality of approximated features. This results in an increase of low-confidence predictions, incurring a longer duration for SSD to remove via NMS post-processing.

3.4.5 Global detection performance and comparison

To demonstrate the strength of our detection framework, we compare the full ACDnet (including FA and MA) against several state-of-the-art methods as presented in Table 3.3. Since the proposed detector targets lightweight action inference for realistic deployment (rather than solely obtaining superior accuracy), only top-performing methods which highlight both accuracy and detection speed are considered for fair comparison.

Alongside detection performance, we briefly summarize each method's architectural and input configuration in Table 3.4 to better visualize distinctions between ACDnet and the others. Under this table, methods such as ACT, STEP, and MOC perform clip-based detection. In other words, they take clips of multiple RGB frames and predict action tubelets (i.e., sequence

TABLE 3.3: State-of-the-art comparison in frame-mAP and runtime (FPS).

Method	Frame-mAP@0.5		FPS
	UCF-24	JHMDB-21	
ROAD-RTF[73]	65.7	−−	28
ROAD-AF	68.3	−−	
ACT[75]	69.5	65.7	25
Two-stream YOLO[103]	71.7	−−	25
STEP[97]	75	−−	21
YOWO-Res[85]	80.4	74.4	34
YOWO-Shuffle	71.4	55.3	−−
YOWO-Mobile	66.6	52.5	−−
MOC-RGB[83]	73.1	−−	53
MOC-TS	78.0	70..7	25
ACDnet	70.9	49.5	75

TABLE 3.4: State-of-the-art comparison in architectural and input configurations. *For any key frame, ACDnet fuses the precedent key frame feature from the past with the current one (considered 2RGB implicitly). For any non-key frame, its feature is approximated based on the preceding key frame feature (considered 1RGB). In the table, "AF", "RTF", and "TS" denote accurate flow, real-time flow and two-stream CNN architecture, respectively.

Method	+2^{nd}-stream OF	+3D CNN	No. input frames : det
ROAD-RTF[73]	Kroeger[104]	✗	(1RGB+1OF) : 1
ROAD-AF	Brox[100]	✗	(1RGB+1OF) : 1
ACT[75]	Brox	✗	(6RGB+30OF) : 6
YOLO-TS[103]	FlowNet2[33]	✗	(1RGB+1OF) : 1
STEP[97]	Brox	3x3D Conv. layers	(6RGB+30OF) : 6
YOWO-Res[85]	✗	3DResNet101	16RGB : 1
YOWO-Shuffle	✗	3DShuffleNetV2	16RGB : 1
YOWO-Mobile	✗	3DMobileNetV2	16RGB : 1
MOC-RGB[83]	✗	✗	7RGB : 7
MOC-TS	Brox	✗	(7RGB+35OF) : 7
ACDnet	✗	✗	2 (1) RGB* : 1

FIGURE 3.8: Examples of false detection in JHMDB-21. (a) Correct action: Jump. (b) Correct action: Sit. (c). Correct action: Stand; ACDnet incorrectly predicts two actions (Stand and Run).

of bounding boxes) spanning these RGB frames. In opposition to clip-based detection, methods such as YOWO gather supportive contextual cues from multiple frames to augment the target one. Along with RGB frames, single/stacked optical flow frames ("OF") are optionally fed to a secondary CNN and fused with the appearance information.

As shown in Table 3.3, ACDnet largely outperforms the other methods in terms of speed. This is ascribed to the feature approximation module and our less complex architectural configuration overall. Methods such as ROAD, ACT, STEP, and MOC-TS all adopt the two-stream CNN framework, which depends on additional optical flow estimation and motion feature extraction (from a separate CNN). In fact, producing optical flow via Brox [100] or FlowNet2 [33] is particularly time-consuming; as a result, most methods employing the flow stream do not take into account optical flow acquisition when measuring runtime (except for ROAD). In contrast, ACDnet only encodes low-resolution optical flow for abstract feature warping, lifting off the need to compute fine-grained flow fields nor two-stream CNN inference. In a similar spirit, YOWO models based on 3D CNNs also prove effective to model actions by accumulating spatiotemporal cues over 16 consecutive frames. However, such an approach inevitably raises the processing cost and time; not only from the model inference aspect, but also data loading (which is excluded in their reported speed performance). It is also worth mentioning that ACDnet achieves comparable accuracy when YOWO employs lightweight 3D CNN architectures (YOWO-Shuffle and YOWO-Mobile), implying the necessity of deeper 3D CNN architectures to effectively reason spatiotemporal context.

In terms of accuracy, ACDnet scores competitively on UCF-24. On the other hand, its performance on JHMDB-21 is less impressive compared to the other methods. As opposed to UCF-24, whose classes of sports activities

are visually more distinctive at each frame, JHMDB-21 contains more classes sharing ambiguous visual context (for example, Sit v.s. Stand, and Run v.s. Walk, etc.). Figure 3.8 demonstrates a few falsely detected examples by our model which result in lower frame-mAP in JHMDB-21. Such ambiguity in the scene is challenging even for human to confidently infer the correct action unless viewing consecutive frames. As shown in column 4 of Table 3.3, ACDnet applies detection on frames far fewer than other methods, which limits its ability to model detailed variations of visual cues over time. In addition, JHMDB-21 consists of short clips for which sparse memory aggregation can only take place minimally. The above factors lead to ACDnet's less satisfactory accuracy on JHMDB-21. This visual ambiguity could generally be mitigated when examining more frames at once (i.e., via clip-based/tubelet-based methods).

3.5 Summary and limitations

Summary. In this chapter, we present ACDnet, a compact action detection network with real-time capability. By exploiting temporal coherence among video frames, it utilizes feature approximation on frames with similar visual appearances, which significantly improves detection efficiency. Additionally, a memory aggregation module is introduced to fuse multi-frame features, enhancing detection stability and accuracy. The combination of the two modules and SSD implicitly reasons temporal context in an inexpensive manner. ACDnet demonstrates real-time detection (up to 75 FPS) on public benchmarks while retaining decent accuracy against other best performers at far less complex settings, making it more appealing to edge device deployment in practical applications.

Concretely, our contributions of this work is three-fold:

- We propose an integrated detection framework, ACDnet, to address both detection efficiency and accuracy. It combines feature approximation and memory aggregation modules, leading to improvement in both aspects.
- Our generalized framework allows for smooth integration with state-of-the-art detectors. When incorporated with SSD (single shot detector), ACDnet could reason spatio-temporal context well over real-time, more appealing to resource-constrained devices.

- We conduct detailed assessment in terms of our models' accuracy, efficiency, robustness and qualitative analysis on public action datasets UCF-24 and JHMDB-21.

Limitations. The proposed action detector produces remarkable runtime performance compared to the previous state-of-the-arts while retaining decent accuracy in scene-related actions (e.g., UCF-24). Nevertheless, it still suffers from limitations, which are described below.

First and foremost, ACDnet localizes action instances only at the frame-level. In order to fully achieve spatiotemporal localization for any given action in a video, proposals in the form of action tubes need to be constructed from the frame-wise detection. Our proposed method is currently not equipped to build action tubes on top of its detection results, which is also reflected in the lack of video-mAP assessment for ACDnet.

Second, ACDnet captures limited temporal cues from the memory aggregation module which recursively accumulates visual cues from past frames. Our inclusion of optical flow only serves to carry out feature warping, thus contributing minimally to reason action-specific pattern. Hence, the detector is challenged to perform effective temporal reasoning when coping with actions embedded with weak appearance context (e.g., JHMDB-21).

Third, even though the proposed feature approximation module aims to raise detection efficiency by reducing the bottle-top feature extraction in most video frames, attempting to detect action instances one frame at a time is intrinsically inefficient and redundant (not to mention the need for an additional optical flow module).

Looking ahead. The above-mentioned limitations provide us several insights to approach the action detection problem in a different manner. In the next chapter, we propose to overcome all the above limitations with the help of a tubelet-based action detection framework and coarse detection paradigm, along with an online detection linking algorithm (addressing limitation 2, 3, and 1, respectively). We will cover these aspects in detail in the next chapter.

Chapter 4

TEDdet: Temporal feature exchange-difference network

4.1 Introduction

As established in previous chapters, the task of spatiotemporal action localization concerns constructing space-time action proposals to capture individual action instances in videos. The problem is inherently challenging owing to actions' inter-class variety and intra-class ambiguity, as well as action tubes' deformable nature in time. It becomes even more complex when detection needs to take place in a real-time and continuous (online) manner on streaming videos, which are considered crucial criteria in a host of scenarios such as unmanned surveillance and human-robot interaction.

Toward the end of Chapter 3, we highlight the limitations of our first action detector, ACDnet. First, frame-wise bounding-box results need to follow a linking strategy in order to formulate long-range action proposals for spatiotemporal action localization, which is missing from ACDnet. Second, we observe that the single-frame approach lacks sufficient feature-level interactions and modeling capacity to effectively reason temporal information embedded in action sequences. Consequently, the accuracy of ACDnet remains low when coping with temporal-related actions (e.g., JHMDB-21), making the detector incompatible with real-world applications.

In this chapter, we devise a new online detection framework to overcome the above limitations. First, instead of the single-frame approach, we concurrently process a series of video frames and predict action tubelets (i.e., a sequence of bounding boxes of the same action category) spanning the input sequence. This allows our detector to learn and infer from video-level representations across successive frames. Specifically, we integrate two lightweight temporal modules on top of multi-frame features to aggregate actions' context and model their movement over time. The above are jointly performed

on cooperative detector branches derived from the CenterNet detector [65] (as opposed to the SSD detector used in the previous chapter). Lastly, an online tubelet linking algorithm is introduced to associate detection results in a timely manner for action tube generation and spatiotemporal action localization.

The proposed action tubelet detector directly operates on a single RGB stream and does not depend on 3D CNN nor optical flow. We refer to it as the **T**emporal feature **E**xchange and **D**ifference action **det**ector, or TEDdet. Throughout the chapter, we validate TEDdet's design choice via a series of experimentation and show that it achieves competitive accuracy at an unprecedented speed (110 FPS).

Related publication. The work presented in this chapter is published in the IEEE Access journal, 2021. The primary investigator in that piece is also the author of this manuscript.

Outline. The rest of the chapter is organized as follows. In Section 4.2, we first evaluate some of the less-desirable qualities of SSD for tubelet adaptation. The foreseen drawback can be alleviated by the anchor-free CenterNet, which is briefly reviewed. Next, we present an overview of our action tubelet detector and its two temporal modeling sub-modules in Section 4.3. We then describe the integration of our temporal sub-modules with the cooperative detector branches derived from CenterNet to form TEDdet's complete detection framework, in Section 4.4. In the same section, we also elaborate how TEDdet carries out online tube generation at test time. We report various experimental validation (quantitative, qualitative, multiple evaluation metrics, etc.) of the proposed detector in Section 4.5. Finally in Section 4.6, the chapter is concluded by a summary of TEDdet and its limitations.

4.2 Review on CenterNet

Limitations of SSD. Even though SSD is widely employed for spatiotemporal action localization thanks to its well-balanced accuracy and speed performance, its detection workflow heavily revolves around pre-defined anchors. More specifically, localizing objects depends on classifying and regressing over numerous anchors densely sampled across every location of the image feature. The anchor-based paradigm not only incurs complicated IoU calculation when matching anchors and groundtruth objects during training, but also generates an enormous amount of highly-overlapped bounding boxes to

be filtered by the non-maximal suppression (NMS) algorithm. Often, achieving precise localization on specific datasets also hinges on heuristic anchor design and placement, hampering the generalization ability of the detectors.

Extending SSD for tubelet detection has been realized and proven successful [75][77][84]. However, we argue that constructing tubelets from an anchor-based backbone is sub-optimal, especially when efficiency is of priority. In most existing tubelet detectors, action tubelets are regressed from a dense collection of 3D cuboids created by expanding 2D image-space anchors in the temporal dimension. One can infer the escalating computational complexity associated with calculating IoUs. For example, any positive cuboid sample used for training is determined by first calculating the mean IoU between all its enclosed anchors and groundtruth action tubes. Similarly at test time, the NMS filtering now concerns computing the IoUs among tubelets in order to remove those that are densely overlapped (rather than individual bounding boxes in image-space). In consequence, we hypothesize that a detector free of anchor operations is conceptually easier when adapted for tubelet detection.

CenterNet. Unlike mainstream CNN-based detectors which infer objects from pre-defined anchors, CenterNet represents objects by their bounding boxes' center points, converting detection to a keypoint estimation problem. Concretely, given an input RGB frame $I \in \mathbb{R}^{3\times H\times W}$ where H and W correspond to the height and width of the input image, CenterNet extracts high-resolution feature maps from an encoder-decoder architecture: a designated 2D CNN backbone followed by an up-scaling decoder block. The resulted feature $F \in \mathbb{R}^{C\times \frac{H}{R}\times \frac{W}{R}}$, where C and R denote the feature channel and down-sampling ratio, is used to predict object centers in the form of a multi-channel keypoint heatmap. Within this heatmap of dimension $cls \times \frac{H}{R} \times \frac{W}{R}$ where cls corresponds to the number of object classes, each peak represents a potential object center of a certain class. Meanwhile, detected centers allow objects' bounding boxes to be regressed directly from the corresponding locations in image feature F. In practice, keypoint estimation and size regression are handled by two separate branches (commonly recognized as **Center** and **Box** branch, respectively). As each feature location is no longer manifested by overlapping anchors, CenterNet's detection pipeline mainly involves the feed-forward inference without the need for the exhaustive NMS filtering to remove redundant detection.

The notion of keypoint estimation is extended in our action tubelet detector to capture moving action instances' centers over time, which will be

elaborated in the remaining of this chapter.

4.3 Overview of TEDdet and temporal sub-modules

4.3.1 Overview

The design of TEDdet builds upon architectural components of CenterNet and two temporal modeling modules, namely, the Temporal Feature Exchange (TE) and Temporal Feature Difference (TD) modules. The TE module performs partial feature exchange among neighboring frames, adaptively aggregating supportive and contextual information from adjacent frames to the target one on which actions will be detected. While TE induces interaction among multiple frame features, TD serves to approximate relative motion based on feature-level displacement to facilitate tracking actors' location shift over the input sequence. The full architecture of TEDdet is illustrated in Figure 4.1.

Given a series of T frames with a temporal stride δ, TEDdet leverages their spatiotemporal features aggregated by TE to categorize and localize action instances' centers on the target frame (Center branch). To model movement of actions, we exploit stacked pair-wise motions produced by TD to estimate actions' trajectories and refine their center locations over the input sequence (Trajectory branch). Finally, each action instance's bounding box on any of the input frame is regressed at the predicted center location of its feature map (Box branch). The above sub-tasks are carried out cooperatively by their respective detector branches.

Combining our coarse-detection scheme (when $\delta > 1$) and online tube generation algorithm, TEDdet is capable of proposing action tubes from online video streams at high localization precision in real-time. We will delve into the technical details of each build block in the following sections.

4.3.2 Temporal Feature Exchange: multi-frame feature aggregation

It has been established that even though 3D CNNs excel to capture rich spatiotemporal context over successive frames, they tremendously raise the computational cost and training complexity. Alternatively, we present the Temporal Feature Exchange (TE) module to facilitate modeling action-specific pattern on top of 2D CNN. Given a set of successive video frames, the TE

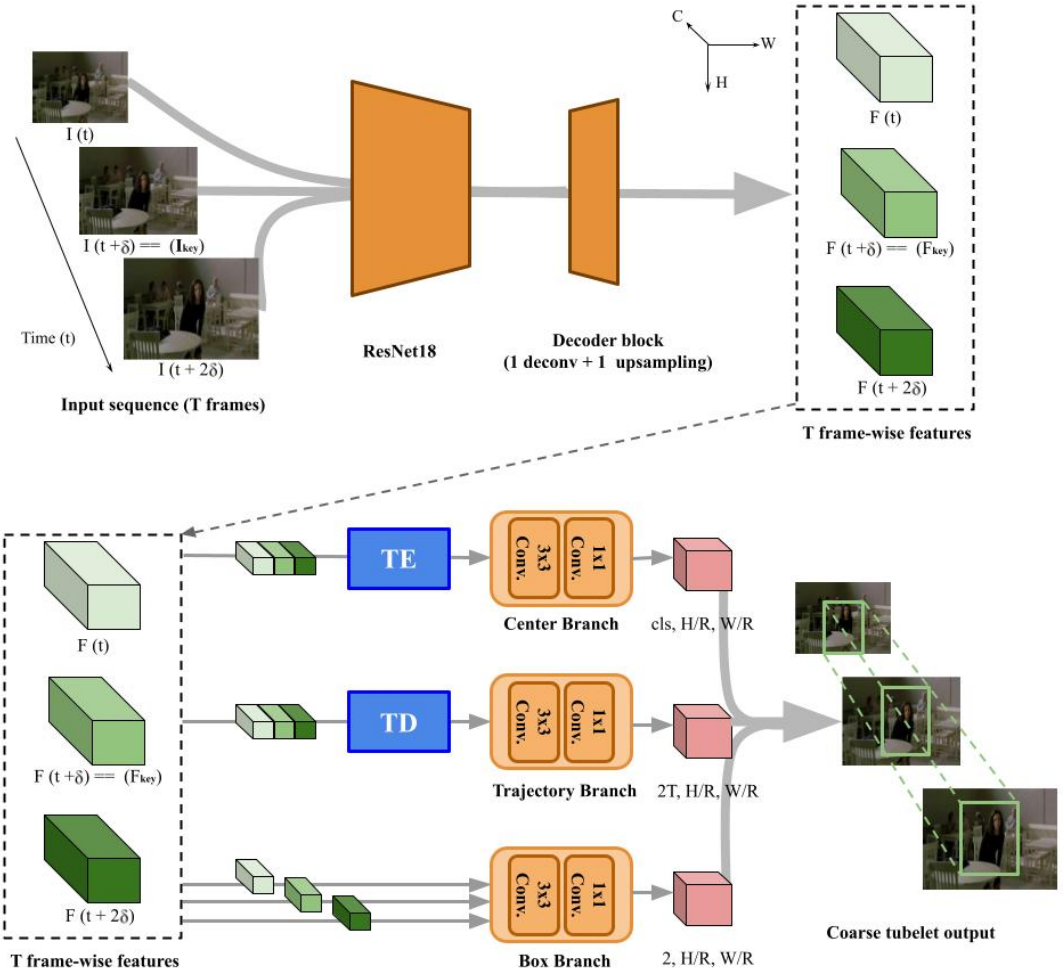

FIGURE 4.1: Overview of TEDdet. Input to the model is a series of T RGB frames ($T = 3$ in this figure) with temporal stride δ. Similar to CenterNet, TEDdet's backbone consists of ResNet18 followed by a decoder block (for adaptive up-sampling). TEDdet's detector head comprises three branches. The TE and TD modules are inserted prior to Center and Trajectory branch, respectively for action center prediction on the key frame and trajectory modeling over the input sequence. Box branch regresses actions' spatial extent independently on each input frame. The outputs of TEDdet are coarse tubelets (according to δ). Note that we also insert an additional TE module between ResNet18 and the decoder block (omitted in this figure) to endow feature interaction upon up-sampling.

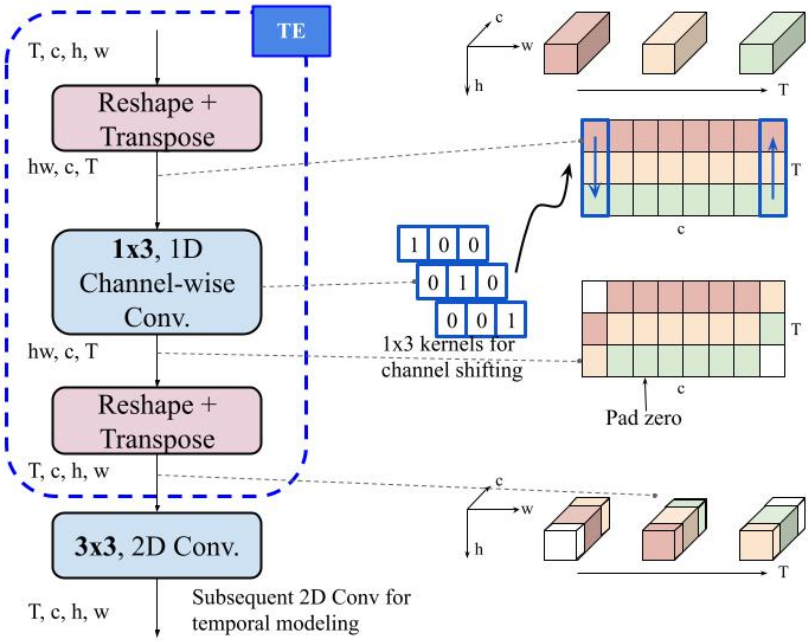

FIGURE 4.2: Architecture of Temporal Feature Exchange module. The left displays the workflow of TE, while its effect on features is illustrated on the right. The grids in white represent features filled with zeros (due to zero-padding at temporal borders).

module aggregates supportive context among nearby frames by partially exchanging their spatial features in a channel-wise manner. The resulting features have now interacted with those at different time steps; therefore, any subsequent 2D convolution can implicitly reason spatiotemporal information from these temporal-aware 2D features.

Figure 4.2 illustrates the architecture of our TE module. Formally, given a series of T RGB frames, they can be fed to a standard 2D CNN in parallel (concatenated in the batch axis) and transformed to abstract feature tensor $F \in \mathbb{R}^{T\times c\times h\times w}$, where c, h and w denote the number of channels, height and width of the feature. The TE module takes such a tensor as input and operates as follows. First, F is reshaped and transposed to $F' \in \mathbb{R}^{hw\times c\times T}$, where spatial dimensions h and w are collapsed into one. Feature exchange between adjacent frames is then carried out by a 1D channel-wise temporal convolution defined by kernel $K \in \mathbb{R}^{c\times 1\times 3}$. Here, each 1×3 kernel of K convolves with a feature channel of F' independently. The weight of these kernels are specifically initialized as: $[1,0,0]$, $[0,0,1]$ or $[0,1,0]$, each corresponding to a temporal forward-shift, backward-shift, and no-shift operation, respectively. The proportion of the three shift operators dictates the extent of interaction among nearby features. The above operation can be summarized as follows:

$$
\begin{aligned}
F'_x &= K * F' \quad (K \in \mathbb{R}^{c\times 1\times 3},\ F' \in \mathbb{R}^{hw\times c\times T}) \\
K[c_f, 1, :] &= [1, 0, 0], \qquad forward - shift \\
K[c_b, 1, :] &= [0, 0, 1], \qquad backward - shift \\
K[c_n, 1, :] &= [0, 1, 0], \qquad no - shift
\end{aligned}
\tag{4.1}
$$

$$
\begin{aligned}
1 \leq & c_f < c/div, \\
c - c/div \leq & c_b < c, \\
& c_n \in c - \{c_f, c_b\},
\end{aligned}
$$

where * and div denote the convolutional operation and feature exchange ratio. In Equation 4.1, c_f and c_b indicate the channel indices where forward- and backward-direction feature exchange take place. The rest of the channels (c_n) does not exchange with those of neighboring features. When taking $div = 4$ as an example, Equation 4.1 depicts $1/2$ of the total feature channels interacting with those of adjacent frames, i.e., $1/4$ forward and $1/4$ backward.

During the exchange, we perform zero-padding accordingly to fill up feature channels at the temporal borders. Finally, the resulted F'_x is transformed back to the original shape as F (i.e., $T \times c \times h \times w$). Additional convolution can be seamlessly appended after F'_x to further extract visual cues of higher semantics. As each of the T features in F'_x has partially interacted with others in adjacent times, the temporal receptive field of any subsequent 2D convolution is implicitly expanded by 3 for spatiotemporal modeling.

4.3.3 Temporal Feature Difference: pair-wise displacement as motion

Similar to the notion of CenterNet's keypoint estimation, we predict action instances by their centers on the target input frame (i.e., key frame) while modeling their movement to other frames via an additional regressor. To achieve this, we propose the Temporal Feature Difference (TD) module for encoding relative motion between any two designated frames. Specifically, the relative motion between frames is encoded by the displacement in their abstract feature maps.

Figure 4.3 depicts the workflow of TD, whose objective is to model action instances' offset with respect to the key frame. Given a feature tensor F

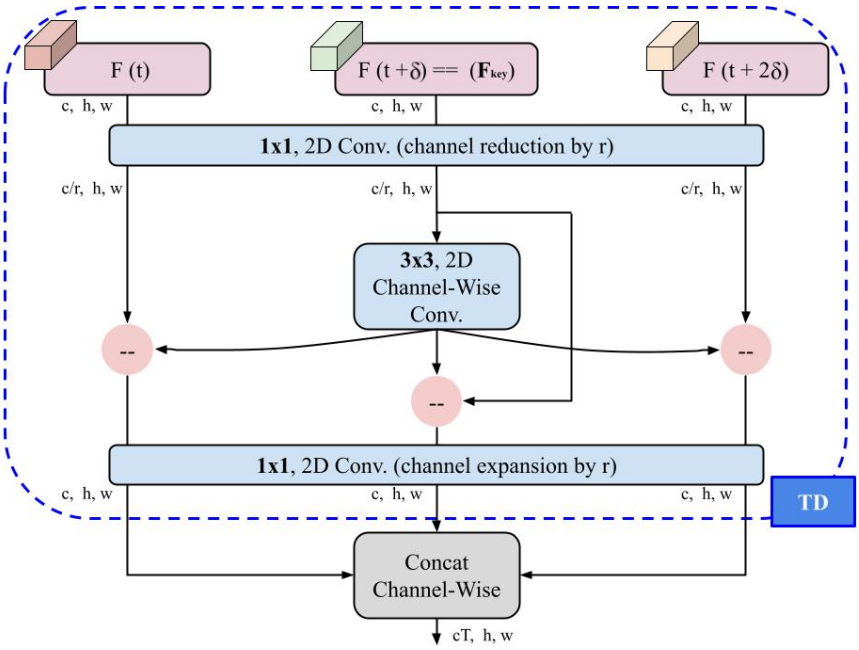

FIGURE 4.3: Architecture of Temporal Feature Difference module.

$\in \mathbb{R}^{T\times c\times h\times w}$ extracted from T frames, a 1×1 2D convolutional layer is first applied to transform F into a compressed latent space for efficiency ($F^r \in \mathbb{R}^{T\times \frac{c}{r}\times h\times w}$). Next, we slice F^r into T portions along the temporal axis, resulting in T features of dimension $\frac{c}{r} \times h \times w$. For F^r_{key} and any F^r_t, which denote the key and t^{th} frame features, we compute their spatial displacement by element-wise subtraction. Note that prior to the subtraction, we follow the standard practice of applying a 3×3 2D channel-wise convolution to all F^r_t. This additional operation has been proven useful for spatially aligning and matching high-level instances acquired at different time steps [53][54][55]. The above procedure is described in Equation 4.2:

$$F^r_{dsp}(t) = conv_{trans} * F^r(t) - F^r_{key},\ 1 \leq t \leq T, \tag{4.2}$$

where $conv_{trans}$ denotes the 3×3 2D channel-wise convolution. Here, F^r_{dsp} $\in \mathbb{R}^{T\times \frac{c}{r}\times h\times w}$ corresponds to all the displacement features between the key frame and others. Finally, we apply another 1×1 convolution to restore the channel number of F^r_{dsp} back to c. We will describe how the displacement feature is utilized for modeling the trajectory of input sequence in the following section.

4.4 Temporal Feature Exchange-Difference action tubelet detection framework

The full architecture of TEDdet, which integrates two complementary temporal modules (TE and TD) and three detetor branches, is summarized in Figure 4.1. To keep TEDdet effective yet compact, we employ the most lightweight ResNet variant, ResNet18 as the feature backbone. The input to TEDdet is T successive frames: $[I_t, I_{t+\delta}, ..., I_{t+(T-1)\delta}]$, where δ denotes the temporal stride between any two sampled frames. Unlike precedent methods which predict tubelets densely across consecutive frames [75][83], our detector does not restrict δ to 1. In TEDdet, we select the middle frame of an input sequence as the key frame (I_{key}), which is the target frame for keypoint heatmap estimation.

4.4.1 TE and Center branch

The TE module can theoretically be inserted prior to any 3×3 convolutional layer in ResNet18. This will enable the following 2D filters to exploit temporal-aware features from precedent layers while still benefiting from ImageNet pre-trained weights. However, unlike previous studies focusing on video-level action recognition [51][54], performing temporal exchange in early CNN layers risks distorting well-learnt spatial cues that could be essential for the localization task in hand. Instead, inspired by [47][48] which apply spatiotemporal modeling only on top of features of higher semantics, we mainly insert the TE module right before Center branch to aggregate abstract multi-frame context for the key frame.

In practice, we implement two variants of temporal exchange: TE_{bi} and stacked TE_{uni}. The former conducts bi-directional exchange to simultaneously collect $1/2$ adjacent-frame features for the key frame (i.e., $1/4$ forward and $1/4$ backward by setting $div = 4$ in Equation 4.1). The latter performs two uni-directional exchange that separately aggregates $1/2$ forward/backward information into two key frame features ($div = 2$), which we then fuse by stacking along the channel dimension. Note that while a single TE module increases the temporal receptive field by 3, stacking multiple TE modules can further enlarge the receptive field for longer-range temporal modeling, as illustrated in Figure 4.4.

Our TE module is composed of learnable shift-operators to adaptively accumulate multi-frame visual context upon which Center branch uses for

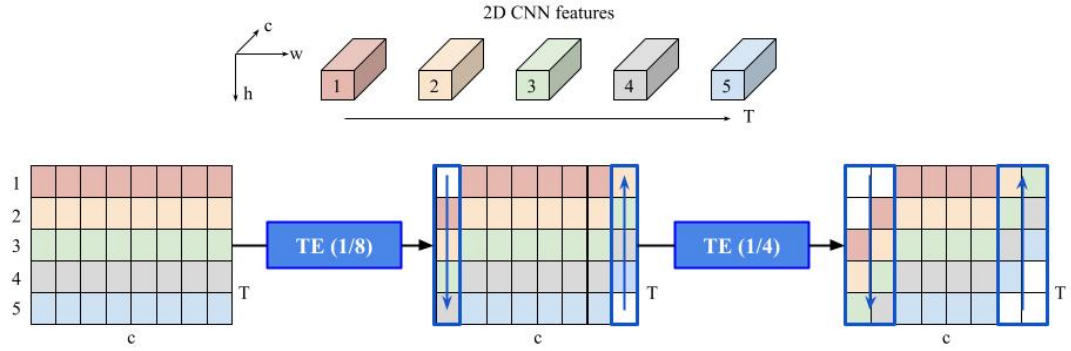

FIGURE 4.4: Stacking multiple TE modules. The above example sequentially applies two TE with different exchange ratios to enlarge the temporal receptive field of key frame feature (row 3) by 5.

keypoint heatmap estimation. The design of Center branch follows that of CenterNet with minor adjustment. It consists of a 3×3 and 1×1 convolutional layer interleaved with ReLU non-linearity. The number of filters is set to 256 and *#actionclasses* for the 3×3 and 1×1 convolution, respectively. The training objective for Center branch (l_{Center}) follows the same focal loss as in the related work by Li et al. [83]. At test time, the obtained heatmap is further filtered to only keep local peaks that are greater than their 8-connected neighbors. The top remaining N peaks across all classes are considered candidate action centers where N is fixed to 100 in our study.

4.4.2 TD and Trajectory branch

The predicted keypoint heatmap encodes centers of action instances in the key frame; it does not guarantee actions' locations in the rest of input frames. To address precise localization over the entire input sequence, we introduce Trajectory branch to track the movement of action instances with respect to the key frame. Prior to this branch, we insert our TD module which generates T displacement features $F_{dsp} \in \mathbb{R}^{T \times C \times \frac{H}{R} \times \frac{W}{R}}$. Each displacement feature estimates pair-wise offset between the corresponding frame from the key frame. To model the trajectory of the whole sequence, we stack every pair-wise feature along the channel dimension ($CT \times \frac{H}{R} \times \frac{W}{R}$) and feed as input to Trajectory branch.

The design of Trajectory branch follows that of our Center branch, consisting of a 3×3 and 1×1 convolutional layer interleaved with ReLU non-linearity. The output of Trajectory branch is the movement map $\hat{m}^{I_{key}} \in$

$\mathbb{R}^{2T\times\frac{H}{R}\times\frac{W}{R}}$, where $2T$ denotes the center offsets (in X and Y directions) between $[I_t, I_{t+\delta}, ..., I_{t+(T-1)\delta}]$ and I_{key} sequentially. In other words, each location in the movement map encodes horizontal and vertical offsets used for repositioning action centers on the non-key frames.

To train Trajectory branch, we first acquire groundtruth action centers on each input frame the same way as for Center branch. Then, the groundtruth movement ($m^{I_{key}}$) of any action instance with respect to I_{key} is simply the offset between its center at I_{key} and those at other frames. Finally, Trajectory branch is optimized based on L1 loss as follows:

$$l_{Trajectory} = \frac{1}{n}\sum_{i=1}^{n}|\hat{m}_i^{I_{key}} - m_i^{I_{key}}|, \tag{4.3}$$

where i indicates the i^{th} out of n action instances.

4.4.3 Box branch

In TEDdet, we extend CenterNet's Box branch to regress the spatial extent of action instances on multiple frames. We assume that class-agnostic bounding box generation does not benefit from temporal modeling. Consequently, Box branch takes every single frame's feature as input and regresses actions' bounding boxes independently for each frame. Specifically, it generates a size map of dimension $2 \times \frac{H}{R} \times \frac{W}{R}$; each location of the map encodes the height and width of a potential action instance. We optimize this branch based on L1 loss as suggested in [65].

The overall training objective of TEDdet is shown in Equation 4.4, where hyperparameter a, b, and c are set to 1, 1, and 0.1 respectively in accordance with [83]. At test time, action centers over input frames are first deduced by Center branch and then refined by Trajectory branch. Finally, the spatial extent of potential action instances are regressed by Box branch based on the localized action centers.

$$l_{TEDdet} = al_{Center} + bl_{Trajectory} + cl_{Box}. \tag{4.4}$$

4.4.4 Coarse-tubelet inference

Given an action video of sufficient length, TEDdet predicts coarse action tubelets for every sequence of T frames (i.e., $[I_t, I_{t+\delta}, ... I_{t+(T-1)\delta}]$). The next sequence of length T frames to detect begins at $I_{t+\delta}$ where there exist $T-1$ overlapping frames between the former and latter. As each frame is

propagated through the shared 2D CNN backbone independently for feature extraction, obtaining overlapping tubelets at two time steps can be performed at an extremely low cost by reusing $T-1$ previously obtained features cached in the buffer. In other words, only the new features corresponding to the current (latest) frame needs to be extracted from the 2D CNN backbone.

The retrieved and newly captured visual cues can be simply fed to the detector branches together for action keypoint heatmap, trajectory and size estimation. Once the resulting tubelets are acquired, TEDdet's feature buffer updates the cache to keep the latest $T-1$ features while dequeueing the oldest one. Conducting action inference in such a way well conforms to the online detection paradigm where input video frames are streamed continuously.

4.4.5 Online tubelet linking and tube generation

We adopt the online linking algorithm similarly employed by Kalogeiton et al. [75]. Given an input video stream, TEDdet detects N initial tubelets from which the top 10 (in terms of confidence scores) are initialized as "active" action links. As the video continues to be streamed and new tubelet candidates are detected, TEDdet enumerates through active links in descending order of their confidence scores and associates them with new tubelet candidates when matched.

In detail, whenever a collection of new tubelet candidates temporally overlaps with an active link, we associate the best-matched tubelet to that link in accordance with the mean-IoU of their bounding boxes on overlapped frames. Additionally, two conditions are respected during the linking process. First, the best-matched tubelet should have a mean-IoU exceeding threshold $\tau = 0.5$. Secondly, each candidate tubelet can only be assigned to an active link. After including a new tubelet, each link's confidence score is updated as the average score of all appended tubelets. An active link stops extending and is terminated ("inactive") either when there no longer exist temporal overlap with newly detected tubelets, or the video stops being streamed.

The final action tubes are derived from all the inactive action links. The temporal extent of any action tube is determined by the starting frame of the initialized tubelet and the end frame of the lastly linked tubelet. Finally, we discard any resulting action tube having either a low confidence score or a

short temporal duration. Action tube generation is carried out independently for each class.

Bounding box interpolation. When the temporal stride (δ) of the input sequence exceeds 1, a fully linked action tube starting at frame 1 comprises coarse detection across [I_1, $I_{1+\delta}$, $I_{1+2\delta}$, ..., etc.]. To obtain dense frame-wise detection, we calculate bounding box results for intermediate frames using a simple coordinate-wise linear interpolation between any two available detection. Such a simple approach reasonably assumes that actions are smooth and continuous in videos.

4.5 Experimental validation

We evaluate the proposed detection framework on **UCF-24** and **JHMDB-21** datasets. As TEDdet is equipped with tubelet linking capability for temporal action localization, we employ the standard frame-mAP as well as video-mAP to measure the accuracy of generated action tubes. We remind our readers that the former metric examines the IoU between detected and groundtruth boxes for each frame separately and does not depend on the online linking strategy. For frame-mAP, the IoU threshold τ is fixed at 0.5 throughout all experiments. On the other hand, video-mAP inspects spatiotemporal overlaps between linked action tubes and groundtruth tubes at multiple IoU thresholds: $0.2, 0.5, 0.75, [0.5 : 0.05 : 0.95]$. Besides accuracy, we also measure TEDdet's detection efficiency in terms of its model size (number of trainable parameters), MACs (number of multiply-accumulate operations), and runtime (FPS: frame-per-second).

4.5.1 Implementation details

We implement TEDdet in Pytorch [105]. Note that the original CenterNet attaches a decoder block of three deconvolution layers at the final convolutional output of ResNet [19]. This serves to adaptively up-scale highly abstracted feature maps by 8 times (each deconv layer spatially up-scales the feature by 2) to better detect small/overlapped objects. Different from object detection, we assume the likelihood of small actors emerging densely in a scene is low. Aiming to conduct highly accelerated and efficient detection, TEDdet's backbone re-uses ResNet18 but reduces the decoder block to one deconvolution layer followed by a bilinear upsampling layer. We also insert

an extra TE module before the decoder block to introduce additional temporal modeling upon feature up-scaling.

Each RGB frame of an input sequence is resized to 288×288. Propagating the input sequence of T frames (input tensor: $3T \times 288 \times 288$) through ResNet18 and our reduced decoder block transforms the input to its video representation (of dimension $256T \times 36 \times 36$). Prior to the detector branches, we apply a 1×1 convolutional layer to reduce the number of channels by a factor of 4 for efficiency gain.

Training details. TEDdet is trained with the Adam optimizer. We set the initial learning rate to $2.5e^{-4}$ for both datasets while initializing ResNet18 with COCO pre-trained weights. On JHMDB-21, we train our model for 10 epochs, during which we reduce the learning rate by a factor of 10 at the 5^{th} epoch. Likewise, UCF-24 is trained for 10 epochs; the learning rate is reduced by half at the end of each epoch since the second one. During training, we freeze ResNet18's first convolutional layer (to reduce chances of overfitting) and apply the same data augmentation as in [75]: photometric transformation, scale jittering, random cropping/expansion and location jittering, etc. In our experiments, all training has been conducted on an NVIDIA Titan V5 GPU with a training mini-batch size of 16.

4.5.2 Effect of feature aggregation and tracking

We first conduct ablation study to validate inclusions of the TE module and Trajectory branch (including TD). A baseline tubelet detector with no feature aggregation nor action center refinement is first established. In other words, given T frames as input, the baseline directly predicts the keypoint heatmap from the key frame and assumes that action centers remain at the same location in time. In this ablation study, we report frame-mAP as the evaluation metric on JHMDB-21; temporal stride δ and T are fixed to 5 and 3, respectively.

Table 4.1 summarizes varied configurations and performances of TEDdet. From the top of the table, it can be observed that all configurations benefit from Trajectory branch by approximately 2.5 frame-mAP. As Trajectory branch does not concern any aspect of classification, these results highlight the importance of refining action centers in time to cope with moving actors in videos. Notably, the increase of model parameters and GMACs are fairly minimal when adding the TD module with Trajectory branch, as they only

TABLE 4.1: Accuracy, MACs, and model size comparison over variants of TEDdet (JHMDB-21). TEDdet performs best in terms of accuracy when incorporating stacked TE_{uni} and Trajectory branch.

CenterNet	✓	✓	✓	✓	✓	✓
TE_{bi}			✓	✓		
stacked TE_{uni}					✓	✓
TD+Traj. branch		✓		✓		✓
Frame-mAP	51.33	53.98	55.82	58.15	58.34	**61.15**
GMACs	3.44	3.48	3.44	3.48	3.45	3.49
Param. (M)	13.74	14.48	13.74	14.48	13.89	14.63

consist of a few convolutional layers and operate on low-resolution feature maps.

When equipped with TD and Trajectory branch, TEDdet's accuracy boosts significantly from the baseline when incorporating feature exchange (nearly 7 and 10 mAP in TE_{bi} and stacked TE_{uni}, respectively). This phenomenon is expected. When $T = 3$, a bidirectional exchange essentially aggregates $1/4$ features each from the forward and backward direction into the key frame. On the other hand, stacked TE_{uni} is able to aggregate and retain more information from all three frames, endowing more context to distinguish among actions. Since TE only introduces the shift-operators implemented by 1D channel-wise convolution, it is highly efficient and barely increases computation and model size.

4.5.3 Effect of sequence coverage

Intuitively, videos spanning a longer duration can embed richer and more discriminative spatiotemporal context. However, longer sequences could potentially introduce irrelevant background cues, as well as raising difficulty to track actions' trajectories. To investigate how video duration affects the accuracy of the proposed detector, we conduct experiments on both JHMDB-21 and UCF-24 by varying the input sequence length T and temporal stride δ.

Figure 4.5 summarizes results of the above experiments. From there we observe that when $T = 3$, TEDdet's accuracy continues to arise on JHMDB-21 when the temporal stride expands ($\delta = 3, 5, 7, and 10$). This not only reflects the advantage of accumulating more diverse context, but also validates Trajectory branch's ability to track action centers further away from the key frame. We also experiment increasing T to 5 (while keeping δ at 5). Even though this configuration essentially has the same temporal coverage as $(T, \delta) = (3, 10)$, it slightly improves the accuracy due to incorporating more

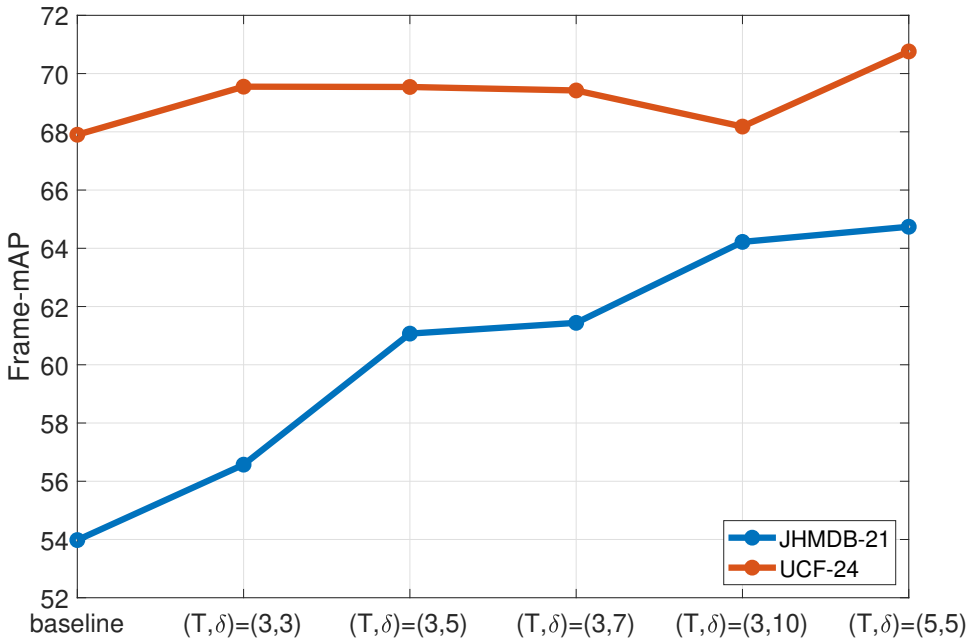

FIGURE 4.5: Accuracy comparison over varied input length (T frames) and temporal stride (δ).

intermediate frame features. To verify whether TEDdet correctly models action's temporal structure (rather than naively gathering contextual information from multiple frames), we repeat the experiment for $(T, \delta) = (5, 5)$ while reversing the order of the input video at test time. The resulting frame-mAP of each action class is displayed in Figure 4.6. It can be noted clearly that actions that depend more on static visual cues (e.g., BrushBair, ClimbStairs, Golf, and ShootBow, etc.) remain unaffected. On the other hand, those relying on strict temporal modeling suffers when the testing video sequences are reversed (e.g., Sit, Stand, Pick, and ShootBall, etc.), confirming that TEDdet correctly models the temporal relation of actions.

For UCF-24, our detector also improves from the baseline method via spatiotemporal feature aggregation. In opposition to JHMDB-21, increasing the temporal stride for UCF-24 exhibits little influence in terms of accuracy, suggesting that strong spatial context around the key frame more or less suffices to determine the corresponding actions. UCF-24 is recognized having strong scene-related cues [53] where background information highly correlates with an action category. Hence, the efficacy of temporal modeling saturates quickly. Lastly, we observe that setting (T, δ) as (5, 5) significantly outperforms (3, 10) by 2.5 mAP even though the two configurations share the

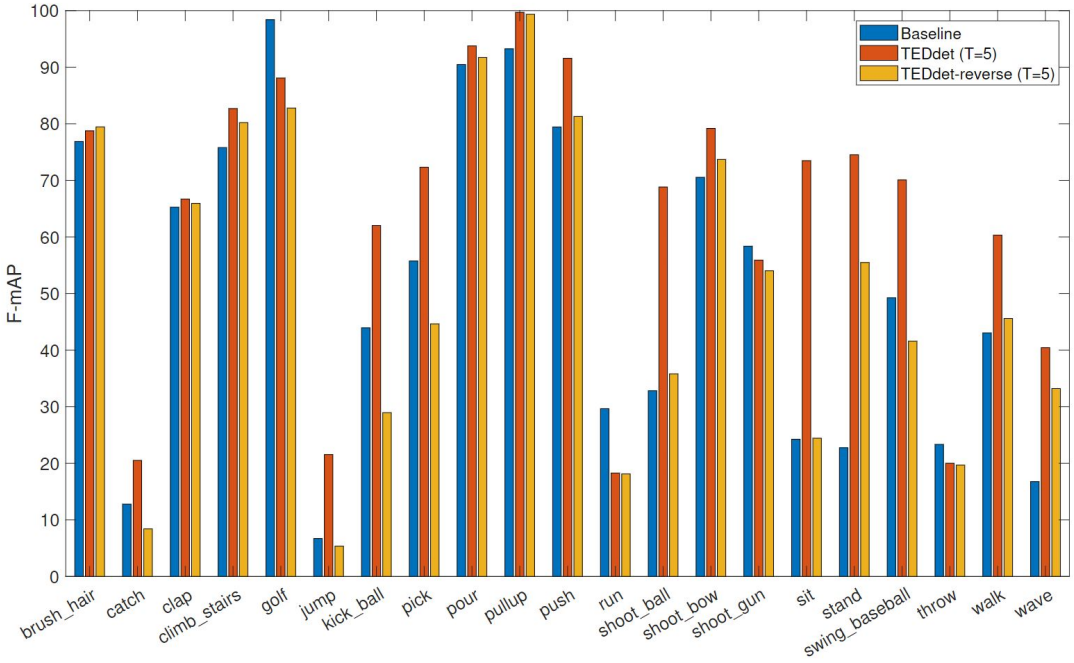

FIGURE 4.6: Per-class frame-mAP performance on forward (correct) and reversed testing input sequence (JHMDB-21). The baseline performance is also included for ease of comparison.

same temporal coverage. We suspect that in the former case, the TD module and Trajectory branch manage to trace action centers' movement more precisely from the extra intermediate frame-features.

4.5.4 Effect of varying sequence coverage at train/test time

In the previous experiment, we train and evaluate target models with matching δ. To assess the robustness/generalization ability of the trained models during inference, we examine them (i.e., models trained by $\delta_{tr} = [3, 5, 10]$) on JHMDB-21 by varying δ at test time ($\delta_{te} = [3, 5, 7, 10]$). JHMDB-21 is selected as it more prominently reflects the effect of temporal modeling according to our previous experiments.

The robustness experiment is reported in Figure 4.7. We observe that models trained at larger δ_{tr} consistently performs comparably or better than others when tested on different δ_{te}. This remains true even when δ_{tr} and δ_{te} are far apart, e.g., δ_{tr}, δ_{te}) = (10, 3). It can also be seen that at $\delta_{tr} = 5$ and $\delta_{tr} = 3$, our models manage to adapt at first when tested at slightly larger temporal strides, but soon degrade in accuracy. These results imply that training with sparsely annotated frames (especially at higher temporal strides) potentially introduces a higher variety of visual pattern for TEDdet to learn and robustly discriminate actions. Such an attribute not only helps

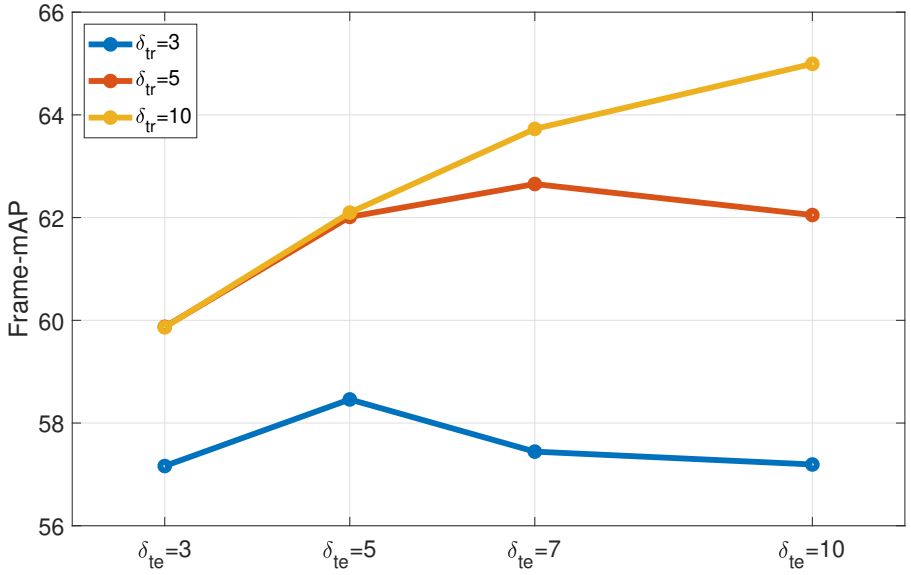

FIGURE 4.7: Accuracy comparison (JHMDB-21) over trained models ($\delta_{tr} = 3, 5, 10$) tested with varied temporal strides ($\delta_{te} = 3, 5, 7, 10$).

to reduce training complexity over long video sequences, but also relaxes our detector's reliance on densely annotated groudtruth boxes.

4.5.5 Action tube generation and runtime

To evaluate TEDdet's spatiotemporal localization capability, we apply the online tube generation algorithm and compute video-mAP based on our top-performing configuration (i.e., T and δ as 5). We also measure runtime (FPS) to support our claim of real-time detection. These results along with frame-mAP are summarized in Table 4.2. Specifically, detection runtime is recorded over the complete time span to obtain action tubes for all videos (including data loading, tubelet inference & linking, and intra-frame interpolation) and then divided by the total number of frames. We set the testing mini-batch size as 1 to simulate processing a continuous video stream.

TEDdet significantly accelerates tubelet prediction and linking with its coarse-detection strategy. It achieves an overall inference speed greater than 110 FPS on both datasets. One may observe that even though the total runtime is similar between the two datasets, the time distribution is notably different. Specifically, data loading takes longer in JHMDB-21 as this dataset comprises shorter videos where TEDdet can not fully exploit feature caching-dequeuing. Instead, it needs to frequently clear its feature buffer and await for a new sequence of T frames upon any new video. On the other hand,

TABLE 4.2: TEDdet's runtime, frame-mAP, and video-mAP performance. The total duration of action tube detection is broken down into three phases (top three rows under "Runtime") and reported in ms (millisecond/frame). We remind our readers that video-mAP is evaluated based on four detection thresholds (i.e., 0.2, 0.5, 0.75, and [0.5:0.05:0.95]).

	JHMDB-21				UCF-24			
Runtime (ms)								
Data loading	3.76				1.09			
Detection	1.63				1.60			
Tube generation	3.67				6.08			
Speed (FPS)	110				114			
Accuracy								
Frame-mAP	64.74				70.76			
Video-mAP	67.86	67.39	53.74	44.71	74.57	50.41	21.82	25.04

tube generation takes perceivablely longer in UCF-24. This is attributable to the fact that tubelet linking and intra-frame detection interpolation handle each action class independently. Unlike previous methods leveraging multi-thread CPU to complete this task [73][75], our current implementation uses only a single CPU thread, thus taking longer to process UCF-24 (24 classes) than JHMDB-21 (21 classes). We provide more qualitative analysis on TEDdet's video-mAP in the following section.

4.5.6 Global detection performance and comparison

After the ablation study, we compare the best configuration of TEDdet with state-of-the-art action detectors to have a holistic view of its performance. Similar to Chapter 3, only the methods which explicitly consider both accuracy and runtime performance are taken into account for fair comparison. Results of frame-mAP and video-mAP are reported in Table 4.3 and Table 4.4 for JHMDB-21 and UCF-24, respectively. Beyond accuracy, we also present comprehensive summaries of these top-performing methods (i.e., backbones and input types) along with their runime (FPS) in Table 4.5.

As shown in Table 4.3 and 4.4, even when prioritizing detection speed and low computation in its design, TEDdet retains decent accuracy on both datasets (more reflected in frame-mAP). Without relying on optical flow and two-stream CNN, our detector obtains comparable and even higher scores than ACT and Zhang et al. Notably, MOC-TS achieves impressive accuracy on both datasets but still counts on optical flow inputs as well as a stronger 2D backbone. When we adapt their pipeline into MOC-lite that is closer to

TABLE 4.3: State-of-the-art comparison on JHMDB-21. See Table 4.5 for architectural and input configuration.

Method	JHMDB-21				
	Frame-mAP@0.5	Video-mAP			
		@0.2	0.5	0.75	0.5:0.95
MR[71]	58.5	74.3	73.1	−−	−−
ROAD-AF[73]	−−	73.8	72.0	44.5	41.6
ROAD-RTF	−−	67.5	65.0	36.7	38.8
ACT[75]	65.7	74.2	73.7	52.1	44.8
AMTnet-TS[77]	−−	73.5	72.8	59.7	48.1
Two-in-One[84]	−−	−−	58.0	42.8	34.6
Two-in-One-TS	−−	−−	74.7	53.3	45.0
Zhang et al.[106]	37.4	−−	−−	−−	−−
YOWO[85]	74.4	87.8	85.7	58.1	−−
MOC-TS[83]	70.8	77.3	77.2	71.7	59.1
MOC-lite	57.3	59.8	61.6	55.4	44.7
TEDdet	64.7	67.9	67.4	53.7	44.7

TABLE 4.4: State-of-the-art comparison on UCF-24. See Table 4.5 for architectural and input configuration.

Method	UCF-24				
	Frame-mAP@0.5	Video-mAP			
		@0.2	0.5	0.75	0.5:0.95
MR[71]	−−	73.5	32.1	2.70	7.30
ROAD-AF[73]	−−	73.5	46.3	15.0	20.4
ROAD-RTF	−−	70.2	43.0	14.5	19.2
ACT[75]	69.5	76.5	49.2	19.7	23.4
AMTnet-TS[77]	−−	78.5	49.7	22.2	24.0
Two-in-One	−−	75.5	48.3	22.1	23.9
Two-in-One-TS	−−	78.5	50.3	22.2	24.5
Zhang et al.[106]	67.7	74.8	46.6	16.7	21.9
YOWO[85]	−−	75.5	48.8	−−	−−
MOC-TS[83]	78	82.8	53.8	29.6	28.3
MOC-lite	68.8	76.3	49.1	23.7	25.1
TEDdet	70.8	74.6	50.4	21.8	25.0

TABLE 4.5: State-of-the-art methods' speed, architectural and input configurations. In the table, "AF", "RTF", and "TS" denote accurate flow, real-time flow and two-stream CNN, respectively. Methods with * output action tubelets over multiple frames (those without * perform frame-wise detection). Having repeated backbones and two input streams both indicate adoption of the two-stream CNN.

Method	Backbone	Input frames	FPS
MR[71]	VGG16×2	1RGB+5OF	4
ROAD-AF[73]	VGG16×2	1RGB+1OF	7
ROAD-RTF	VGG16×2	1RGB+1OF	28
ACT*[75]	VGG16×2	6RGB+30OF	30
AMTnet-TS*[77]	VGG16×2	2RGB+10OF	21
Two-in-One*[84]	VGG16	6RGB	25
Two-in-One-TS*	VGG16×2	6RGB+30OF	12.5
Zhang et al.[106]	VGG16×2	3RGB	38
YOWO[85]	Darknet19+3DResNext101	16RGB	34
MOC-TS*[83]	DLA34×2	7RGB+35OF	25
MOC-lite*	ResNet18	7RGB	24
TEDdet*	ResNet18	5RGB	110

TEDdet's lightweight setting (i.e., replacing DLA34 by ResNet18 and removing the optical flow stream), the strength of TEDdet becomes evident. YOWO also demonstrates superior spatiotemporal modeling capacity through fusing 2D and 3D CNN features. However, their detection in every target frame depends on extracting contextual information from 16 nearby RGB frames by a very deep 3D CNN.

Specifically, we observe that the proposed action detector has more rooms for improvement in its video-mAP at lower detection thresholds (more prominent in JHMDB-21). The relatively low performance in this metric mainly results from TEDdet retaining a higher number of low-confidence action tubes after tubelet-linking (which contributes to more false-positive samples). Typically, low-confidence tubelet predictions tend not to be consistent in time and can be progressively discaarded during the linking phase by dense-tubelet detectors ($\delta = 1$) [75][77][83][84]. In contrast, TEDdet coarsely detects from an extended video sequence followed by intra-frame detection interpolation. Such a coarse-to-fine approach presumes the consistency of any action instance within a longer duration, making TEDdet more difficult to suppress false-positive detection later via tubelet linking/association.

In addition to retaining more low-confidence detection, it comes to our attention that in a few extreme cases when actors undergo drastic location

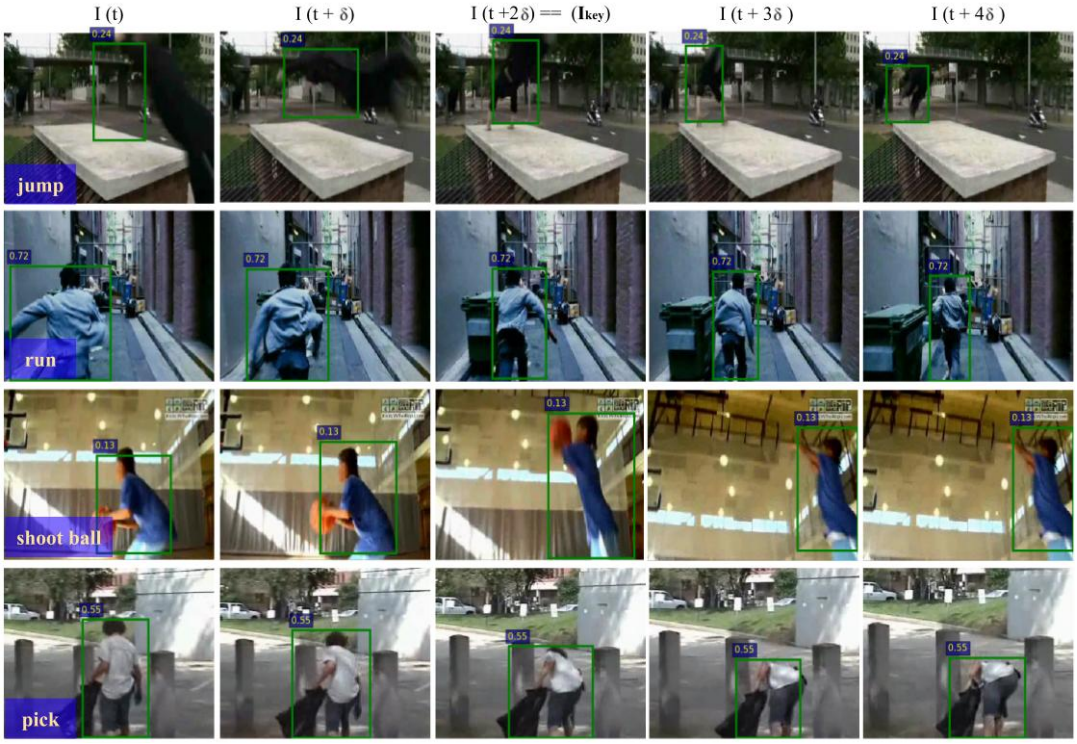

FIGURE 4.8: Examples of action sequences where actors undergo significant location shift. Here, we only show predicted tubelets of the correct class (with confidence score above 0.1). In the first row, TEDdet could not precisely track the "Jump" instance throughout the whole sequence due to drastic motion (blur) and ambiguous background. On the other hand, other actions (Run, ShootBall and Pick) with perceivable location shift are tracked properly. In the above examples, both T and δ are set to 5.

shift in time, TEDdet's Trajectory branch is challenged to precisely track action centers away from key frames (refer to the top-most example in Figure 4.8). In this case, the IoU-based linking strategy no longer guarantees the validity of linked action tubes, which negatively impacts video-mAP accuracy. Overall, TEDdet still demonstrates high robustness coping with most other situations where highly perceivable shift in actors' locations is present).

Lastly, it can be observed from Table 4.5 that TEDdet significantly outperforms others in terms of runtime. This is mainly attributable to its coarse-to-fine detection paradigm which accelerates action tube generation. Among all detectors listed in the table, TEDdet is also equipped with the most lightweight backbone and does not depend on a second set of CNN to extract optical flow features (nor calculating optical flow). Note that any method that utilizes repeated backbones adopts the two-stream CNN framework and makes use of optical flow (denoted by "OF" in the table). Last but not least, our model leverages the least number of input frames to carry out action inference, inherently reducing computational cost (i.e., MACs) by many folds. Overall, TEDdet's lightweight design, real-time inference capability and competitive accuracy makes it more compatible with computation-constrained devices and appealing to deployment in real-world applications.

4.6 Summary and limitations

Summary. In this chapter, we propose a lightweight action tubelet detector coined TEDdet. Its Temporal Feature Exchange module facilitates channel-wise feature interaction by aggregating action-specific visual patterns over successive frames, enabling spatiotemporal modeling on top of 2D CNN. To address actors' location shift in the sequence, our Temporal Feature Difference module approximates pair-wise motion among designated frames in their abstract latent space. The two modules are integrated with an existing anchor-free detector (CenterNet) to cooperatively model action instances' categories, sizes and trajectories for precise tubelet generation. TEDdet exploits larger temporal strides to efficiently infer actions in a coarse-to-fine and online manner. Our experimental results on the public UCF-24 and JHMDB-21 datasets demonstrate that without relying on any 3D CNN or optical flow, our action detector achieves competitive accuracy at an unprecedented speed (110 FPS), suggesting a much more feasible solution pertinent to realistic applications.

The main contributions of our work can be summarized as follows:

- We present two lightweight temporal modeling modules: Temporal Feature Exchange (TE) and Temporal Feature Difference (TD) to facilitate learning action-specific spatiotemporal pattern and trajectory.
- We propose TEDdet, an integrated action tubelet detector on top of 2D CenterNet and TE-TD plug-in. Our detector operates in a coarse-to-fine manner; alongside the online tube generation algorithm, TEDdet's detection speed well exceeds real-time requirement (110 FPS).
- We conduct comprehensive analysis and comparison in terms of TEDdet's accuracy, robustness, and efficiency on public UCF-24 and JHMDB-21 datasets.

Limitations. We claim that TEDdet provides a balanced spatiotemporal localisation performance between accuracy and efficiency. Its architectural design composing a lightweight 2D CNN backbone, cooperative detector branches and coarse detection scheme makes it more compliant with computationally constrained devices in real-world deployment. Meanwhile, we observe two less desirable attributes as described below.

First, TEDdet's performance in video-mAP is somewhat compromised by the manifestation of false-positive detection which can't be easily removed during the linking process. Leveraging successive RGB frames improves action modeling at a large margin when compared to the single-frame approach (e.g., ACDnet), but it remains challenging to distinguish among action categories that rely less on scene-related cues.

Second, scaling up TEDdet is a non-trivial task. More specifically, the classification task is performed on top of partially-exchanged features produced by the TE module. Such a partial exchange mechanism retains incomplete visual cues from each frame. In other words, the longer an action sequence is (i.e., more participating frames), the less discriminative information each individual frame contributes after feature exchange.

Looking ahead. In the next chapter, we address the aforementioned drawbacks by devising a closer-to-optimal formulation for spatiotemporal action detection. Mainly, we take a step back to revisit the well-established two-stream CNN framework modeling appearance-motion correspondences. The inspiration leads to our following detector which captures action-specific pattern simultaneously from visual and explicit motion cues. As will be presented in Chapter 5, the new approach further helps to distinguish challenging action categories and reduce false-positive detection.

Chapter 5

AMMA: Accumulated micro-motion features for real-time spatiotemporal action localization

5.1 Introduction

So far in this manuscript, we have covered two single-stream action detection methods (based on the RGB modality) for trimmed/untrimmed videos in an online fashion. The single-frame detection approach such as ACDnet presented in Chapter 3 induces little interaction among action sequences over time, thus offering limited temporal modeling capacity. The above can be alleviated by concurrently inferring actions from a series of video frames (such as TEDdet in Chapter 4), whose combined visual cues encode the spatiotemporal nature of the underlying actions. The latter, also referred to as the tubelet-based approach, demonstrates significant improvement capturing actions' contextual evolution, and therefore is widely adopted by the research community of spatiotemporal action localization.

In spite of taking into account the contextual information from RGB sequences at once, our experimental results from Chapter 4 still indicate notable presence of miss-classified action instances. Moreover, it may appear unorthodox that most existing action tubelet detectors [75][97][83] need to leverage both consecutive RGB frames and complementary optical flow fields to enhance detection accuracy. Intuitively, the motion of actions should already be embedded and can be inferred from consecutive RGB frames. The dependence on the additional optical flow suggests that motion cues carry relevant spatiotemporal information that cannot be implicitly reasoned from RGB series alone.

Even with the insight that explicit motion representations are critical for action inference, directly re-adopting optical flow is out of the question for

real-time, online action detection. More specifically, it is timely and computationally costly to acquire dense optical flow on-site. Instead, we attempt to first identify the most essential information carried by optical flow that helps to distinguish among actions. Once these subtle yet highly relevant features are clarified, an alternative, easy-to-compute motion representation retaining these critical attributes can be devised in lieu of optical flow.

Following the clarification of optical flow's role in actions, we present **A**ccumulated **M**icro-**M**otion **A**ction detector, or AMMA in this chapter. AMMA is a real-time tubelet detector based on 2D CNN backbones. It adopts a coarse-level tubelet detection paradigm similar to TEDdet from Chapter 4, acquiring actions' spatiotemporal context from sparsely sampled visual cues, while additionally incorporating their complementary motion dynamics. Specifically, to lift reliance on the computationally expensive optical flow, we design a learnable, implicit motion approximator that accumulates short-term motion dynamics ("micro-motion") of actions. In AMMA, micro-motion is computed on-the-fly from RGB frames. Motion features can then be extracted and adaptively fused with the appearance ones at multiple scales via lateral connections to produce temporal-aware features based on the 2D CNN backbone.

Unlike the single-stream approaches introduced in Chapter 3 and Chapter 4, AMMA partially adopts the two-stream CNN framework which induces learning motion-appearance correspondence from both visual and dynamic cues. Beyond its spatiotemporal feature backbone, AMMA aggregates multiple temporal-aware features at its detector head to permit long-range action modeling. Precisely, the detector head consists of three cooperative branches for coarsely recognizing and localizing action instances, modeling their movement over time, and regressing their sizes. In addition, AMMA's detection pipeline is generic and can be easily integrated with ultra-lightweight CNN architectures for resource-constrained edge devices.

Related publication. The work presented in this chapter aims to be submitted to Elsevier's Neurocomputing journal.

Outline. The rest of the chapter is organized as follows. In Section 5.2, we briefly retrace how motion cues (specifically optical flow) facilitate action understanding. The finding inspires us to devise an alternative motion representation that is effective yet more efficient. Next, we provide a high-level overview and illustration of AMMA in Section 5.3. In Section 5.4 and 5.5, technical details of AMMA's backbone and detector branches are elaborated in that order, followed by description of the complete tubelet inference

and linking procedures in Section 5.6. It is worth mentioning that in these sections, we re-visit (and even expand upon) several concepts/techniques shared with TEDdet from Chapter 4 in order to make this chapter more self-contained. Similarly, we report various experimental validation (quantitative, qualitative, multiple evaluation metrics, etc.) of the newly proposed detector in Section 5.7. Finally, the chapter is concluded by a summary and the proposed approach's limitations in Section 5.8.

5.2 How optical flow facilitates action understanding?

Optical flow encodes apparent motion of moving objects in the scene. When combining optical flow with RGB frames as input, two-stream CNN networks consistently achieve better accuracy than their counterparts inputting only RGB frames. As a result, selecting optical flow as the designated motion representation has been widely exploited for video action recognition [21][23][24][38]. Such a common practice has also been extended to many works in spatiotemporal action detection [73][75][84][83].

At first glance, it may seem natural to use explicit motion estimation for action-related tasks. As a counter point, however, Sevilla-Lara et al. [107] argue that the underlying interaction between optical flow and action recognition is unclear and rarely studied. Hence, the authors conduct a series of experiments with regards to clarifying the relation between motion cues and actions. In their experimental setup, one RGB frame and a block of five optical flow fields are used as separate modalities within a two-stream CNN framework.

Sevilla-Lara et al. observe that in the temporal stream, when temporal coherence is disrupted (e.g., by randomly shuffling the order of the flow fields) while flow fields still capture the shapes of moving objects, recognition accuracy only marginally decrease. Further examination such as altering the colormap of the input demonstrates that recognition accuracy is greatly compromised only when the modified target is an RGB frame. These results suggest that beyond the temporal structure embedded in this motion cue, optical flow serves to improve action recognition largely due to it being appearance invariant. In brief, optical flow results in simpler learning and generalization, as the classification model does not need to learn from vast inter-class appearance variations popularized in action videos.

Furthermore, current approaches tend to follow a sequential procedure when applying motion cues for action recognition. In this case, optical flow is separately estimated and then used as input (in parallel with RGB frames) to the subsequent action recognition module. Such a two-stage paradigm assumes that more accurate flow (measured by end-point-error, or EPE, which computes the Euclidean distance between the estimated and groundtruth flow) is associated with superior recognition accuracy. On the contrary, several studies found that this correlation is weak [35][34][107]. Instead, the unified approach which fine-tunes flow estimation in accordance with the learning objective of action recognition (i.e., cross-entropy, or CE), yields superior accuracy (assuming that the optical flow module is learnable such as [32][33]). When the two sets of flow fields are compared, namely, either being optimized against EPE or CE loss, the most salient differences consistently take place around motion boundaries and where humans are located.

To summarize, optical flow is found useful for action recognition mainly because **it is invariant to appearance** (even when flow vectors are inaccurate). In addition, **numerical improvements on action recognition accuracy mainly arise from changes in the flow near boundaries of the human body**. These observations serve as the basis of our newly devised micro-motion representation, which will be elaborated in the following sections.

5.3 Overview of the detection framework

The newly proposed action detector termed Accumulated Micro-Motion Action detector (AMMA) is an end-to-end 2D-CNN-based tubelet detector. As summarized in Figure 5.1, AMMA takes multiple short video clips as input and produces coarse action tubelets spanning the input sequence. Each video clip comprises t consecutive frames. From each video clip, appearance information is extracted from the latest frame (I_t) via the 2D CNN backbone. Alongside appearance feature extraction, each clip is fed to our micro-motion module which generates and accumulates short-term dynamics of actions. From the micro-motion, temporal cues are further extracted and fused with the appearance features of I_t via multiple lateral connections to encode short-term spatiotemporal context for the clip.

To model longer spatiotemporal structures across multiple video clips, AMMA aggregates their respective short-term temporal-aware features by stacking them in the channel dimension at its detector head. In essence, the

aggregated features are fed to three branches to recognize and spatially localize action instances' centers, model the trajectories of action centers over time, and regress their spatial extent (e.g., height and width). Cooperative modeling of the three branches produces action tubelets that are temporally coarse, where detection takes place only at I_t of each clip. Action tubelets can be incrementally detected and linked over time following the designated matching strategy, forming long-range action tubes for spatiotemporal localization. Finally, dense frame-wise detection is acquired by intra-frame interpolation between any two detection. The following sections describe each working module of AMMA in detail.

5.4 AMMA - Backbone

As pointed out earlier, the fact that existing action tubelet detectors still depend on optical flow to reach competitive accuracy implies insufficient spatiotemporal information embedded in dense RGB frames alone. Alternatively, our proposed tubelet detector learns and infers actions by coarsely extracting appearance features spanning a longer sequence, thus incorporating richer contextual variations. Further, instead of the computationally more expensive optical flow, AMMA's backbone extracts short-term dynamics of actions in the form of accumulated micro-motion, enabling 2D CNN to adaptively model appearance-motion correspondences and acquire temporal-aware appearance features.

5.4.1 Clip-level appearance information

We define an input video clip V_{cp} to contain t consecutive RGB frames, where $V_{cp} = [I_1, I_2, ..., I_t]$. The dimension of each frame is $H \times W \times 3$. Since neighboring frames share highly resembling visual cues, we only extract appearance information from I_t via a 2D CNN. Formally, we adopt a reduced variant of the encoder-decoder architecture used by Zhou et al.'s CenterNet [65] as the 2D backbone. In their work, three deconvolution layers have been added at the end of ResNet's final convolutional layer as the decoder block. This serves to adaptively project highly abstracted features onto a spatially larger feature map to facilitate dense detection of small/overlapped objects. Different from object detection, it can be reasonably assumed that the likelihood of actors emerging densely in a scene is low. With this insight, AMMA's backbone decoder is implemented with only one deconvolution layer followed

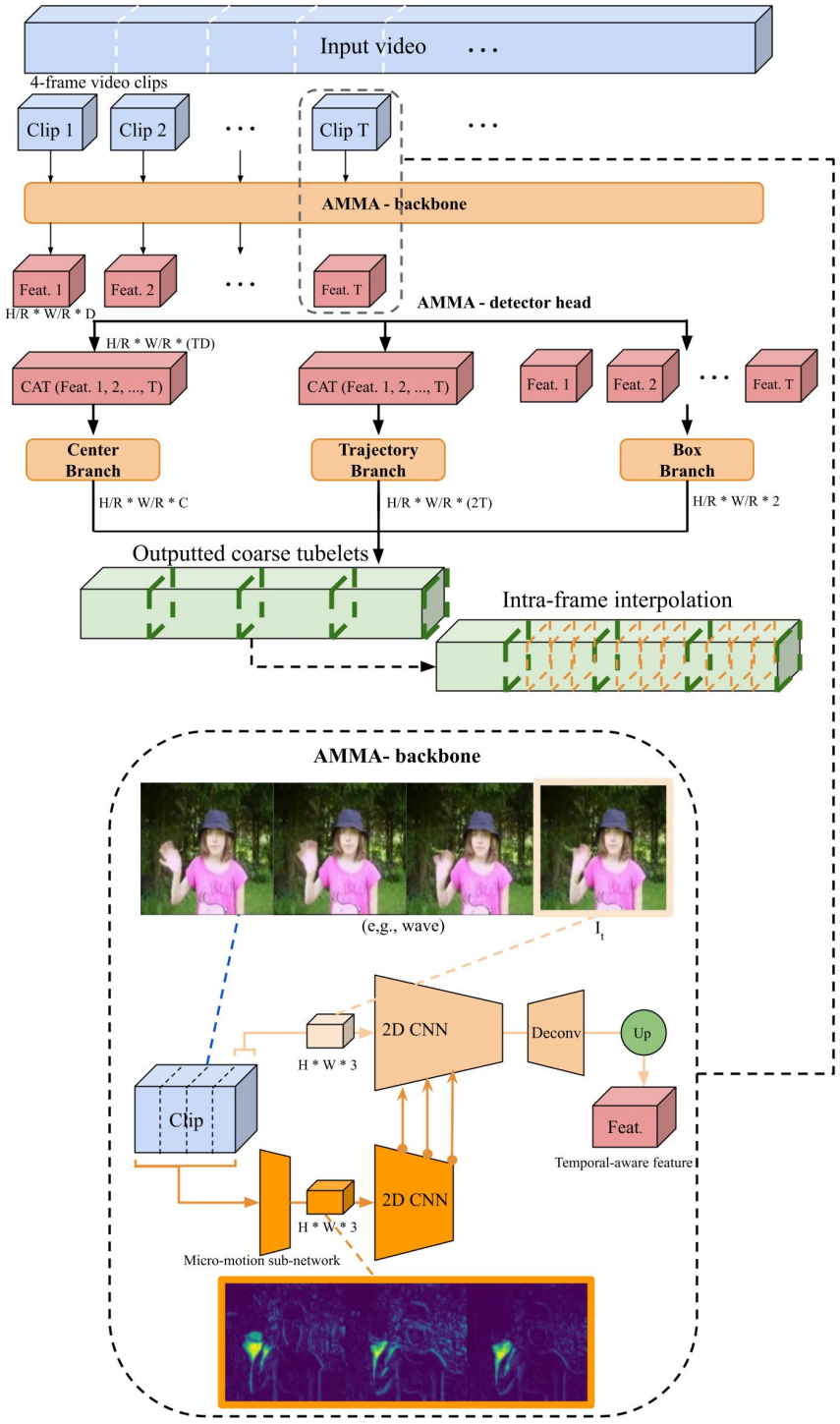

FIGURE 5.1: Overview of AMMA. AMMA's backbone (BOTTOM) takes an input clip of t frames at a time ($t = 4$ in this study), encodes short-term action dynamics as accumulated micro-motion, and outputs a motion-aware feature tensor by merging appearance (from I_t) and complementary motion information via lateral fusion. (TOP) Beyond a single clip, AMMA enables long-range spatiotemporal modeling by aggregating multiple clip-level features at its detector head consisting of three cooperative branches. After merging results of the detector branches, the predicted tubelets are coarse in time. From the coarse tubelets, dense frame-wise detection can be interpolated between any two clips in a later stage.

by bilinear upsampling. The resulted appearance feature is a tensor with dimension $\frac{H}{R} \times \frac{W}{R} \times D$, where D corresponds to the channel dimension of the feature. In practice, R and D are 8 and 256, respectively.

A single clip of t frames only covers a narrow temporal window, which could fail to encode distinctive spatiotemporal patterns of certain actions. To address this, we aggregate clip-wise features from T consecutive clips, enabling richer spatiotemporal context to be captured from a longer sequence. The aggregated features are further processed at AMMA's detector branches to infer action tubelets, which will be covered in depth in Section 5.5.

5.4.2 Accumulated micro-motion: clip-level action dynamics

A short sequence of t frames still potentially embeds crucial dynamic information which I_t alone does not carry. Alternative to optical flow which is commonly prepared in advance due to its high computational cost, we devise a simpler, adaptive motion cue which highlights the small displacements of motion boundaries, as derived from the observations mentioned in Section 5.2. Specifically, we uncover motion information of a clip by simply accumulating the appearance variation between I_t and its precedent frames in the shallow-CNN feature space. The implicit motion representation is referred to as accumulated micro-motion.

Shallow-CNN features tend to reflect local patterns (e.g., edges or textures) with low receptive fields. The difference map between two such low-level features within close temporal proximity inherently encapsulates the temporal evolution of various general patterns. Likewise, the recent study by Zhang et al. [31] demonstrates that motion boundaries derived from deeper-CNN features with large receptive fields do not improve recognition well, as high-level features have been overly abstracted and lost critical information on the spatial boundaries of moving targets.

Formally, we define the shallow convolutional block, $Conv_{5\times5}$, as eight 5×5 convolutions with strides of 1 and paddings of 3. The input to the convolutional block is any clip V_{cp} where all its frames are first downsampled by two via a max pooling layer. The downsampling operation comes from our observation that the difference map between two shallow features within close temporal proximity retains very small values in most areas, i.e., it only contains high responses in motion salient regions. As the difference map exhibits high sparsity, it is more efficient to process it in a low-resolution space

without much loss of information. Concretely, the above steps are described as follows:

$$[F_1, F_2, ..., F_t] = Conv_{5\times5}(MaxPool([I_1, I_2, ..., I_t])), \tag{5.1}$$

$$MM_i^d(x,y) = F_t^d(x,y) - F_i^d(x,y),\ for\ i = 1 : t-1, \tag{5.2}$$

where in Equation 5.1, $F_1, F_2, ..., F_t$ represent shallow-CNN features of frames in V_{cp}, each with a dimension of $\frac{H}{2} \times \frac{W}{2} \times 8$. In Equation 5.2, the $F^d(x,y)$ denotes the intensity of a feature at its d^{th} channel and pixel location (x,y). As expressed in this equation, each micro-motion MM_i corresponds to the feature-level difference between the respective frame I_i and I_t.

To efficiently encode motion variation across different feature spaces and time steps, all MM^d are first accumulated into one channel to manifest the motion magnitude, as shown in Equation 5.3:

$$AMM_i(x,y) = \sqrt{\sum_{d=1}^{8} (MM_i^d(x,y))^2}\ ,\ for\ i = 1 : t-1. \tag{5.3}$$

Note that unlike optical flow fields which typically encode horizontal and vertical motion vectors, our motion cue is an appearance-invariant saliency map reflecting small displacement of motion boundaries which satisfies the observations brought up in Section 5.2. To further incorporate temporal structure, we concatenate resulted micro-motion acquired at different time steps (i.e., $t-1$ AMM_i) in the channel dimension, followed by a bilinear upsampling operation so that the final clip-level micro-motion matches the spatial dimension of the original frames (we first downsample the frames via max pooling in Equation 5.1). In practice, clip length t is defined as 4; the accumulated micro-motion of a clip thus has the same dimension as an RGB frame (i.e., $H \times W \times 3$).

5.4.3 Multi-scale spatiotemporal fusion

In AMMA's backbone, multi-scale lateral fusion is utilized to incorporate micro-motion features into the appearance ones. We partially duplicate the 2D CNN from the RGB stream, and apply the network on accumulated micro-motion. With the above setup, each CNN backbone dedicates to extracting clip-level spatial and temporal information. Then, lateral connections, which are common practices in action-related domains [39][84] are attached

between designated layers of the two CNNs, where weighted summation is carried out to fuse their respective features. Specifically, we devise uni-lateral fusion connections; only the spatial CNN is aware of the complementary motion context when a clip is inputted.

In AMMA, the weights to sum spatial and temporal information are learnable scalars that add up to 1. In addition, the number of lateral connections dictates the extent of fusion between actions' visual and dynamic cues, which will be studied more thoroughly in Section 5.7. Since feature fusion is conducted by summation at the spatial CNN, the dimension of the short-term motion-aware features remains $\frac{H}{R} \times \frac{W}{R} \times D$.

5.5 AMMA - Detector branches

Once clip-level features are extracted from their respective clips, AMMA's detector head aggregates multiple of them for long-range spatiotemporal modeling and action tubelet inference. The detector head is composed of three branches similar to those of TEDdet from the previous chapter: Center branch, Trajectory branch, and Box branch. The function of each branch is reviewed in Figure 5.2.

5.5.1 Center branch

Given a sequence of T clip-level features, Center branch aggregates all clips' spatiotemporal information and locates action instances by their centers at the end of the sequence. In other words, it finds centers of actions with respect to the final frame of the T^{th} clip (i.e., I_t^T). To aggregate clip-wise context, all T features are first stacked along the channel dimension to form video representation $F_{stack} \in \mathbb{R}^{\frac{H}{R} \times \frac{W}{R} \times TD}$. Afterwards, F_{stack} is fed to a standard 3×3 and 1×1 convolutional layer in that order interleaved with ReLU non-linearity, outputting action heatmap $\hat{L} \in \mathbb{R}^{\frac{H}{R} \times \frac{W}{R} \times C}$ for I_t^T, where C corresponds to the number of action classes. Each value of $\hat{L}_{x,y,c}$ indicates the probability of detecting action instance of class c at location (x, y) of the heatmap.

We train Center branch following the setup in [65]. In detail, the groundtruth heatmap $L \in \mathbb{R}^{\frac{H}{R} \times \frac{W}{R} \times C}$ associated with a T-clip sequence is derived from the groundtruth center location (x_{c_i}, y_{c_i}) of I_t^T, where c_i corresponds to the true class of action instance i. We set heatmap $L_{x,y,c} = 0$ for all classes except for the true class. When $c = c_i$, a Gaussian kernel is applied to generate soft

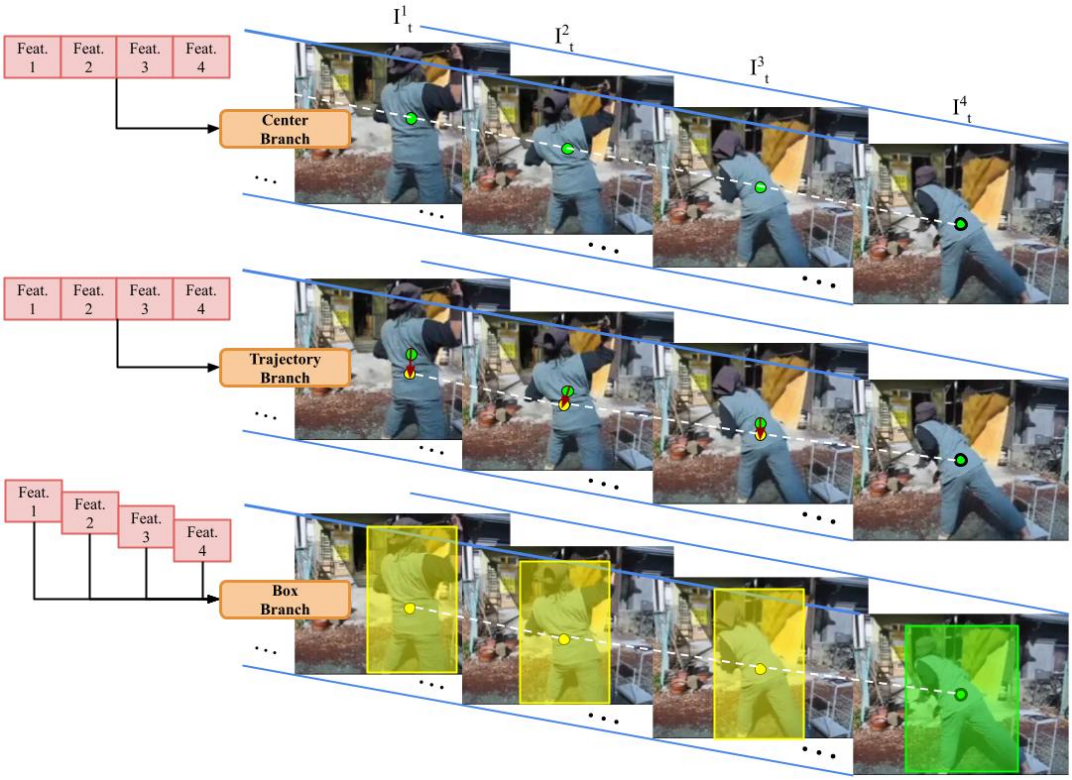

FIGURE 5.2: Overview of AMMA's detector branches. Given an input sequence of T clips ($T = 4$ in this figure), Center branch (TOP) detects action centers at I_t^T (i.e., the key frame). Trajectory branch (MIDDLE) predicts center offsets with respect to Center branch's prediction, and adjusts action centers for $I_t^1, I_t^2, ...I_t^{T-1}$ (i.e., non-key frames) accordingly. Finally, Box branch (BOTTOM) regresses action instances' height and width at action centers deduced by the other two branches.

heatmap $L_{x,y,c_i} = exp(-\frac{(x-x_{c_i})^2+(y-y_{c_i})^2}{2\sigma^2})$, where the salient region surrounds (x_{c_i}, y_{c_i}) and its dimension is determined by σ^2 derived from the groundtruth instance's size. The training objective for Center branch follows the focal loss as shown below:

$$l_{Center} = -\frac{1}{n}\sum_{x,y,c}\begin{cases}(1-\hat{L}_{xyc})^{\alpha}log(\hat{L}_{xyc}), & \text{if } L_{xyc}=1\\ (1-L_{xyc})^{\beta}(\hat{L}_{xyc})^{\alpha}log(1-\hat{L}_{xyc}), & \text{otherwise,}\end{cases} \tag{5.4}$$

where n is the number of groundtruth instances, while α and β are hyperparameters of the focal loss.

In the inference stage, the resulted heatmap is further filtered independently for each class to only keep local peaks that are greater than their 8-connected neighbors. Finally, the top N peaks across all classes are considered candidate action centers. In this study, we set α, β, and N to 2, 4, and 100 respectively.

5.5.2 Trajectory branch

Trajectory branch complements Center branch by modeling action instances' movement at frames $I_t^1, I_t^2, ..., I_t^{T-1}$ with respect to I_t^T. Similar to Center branch, Trajectory branch aggregates T clip-level features by concatenation across the channel dimension, followed by a standard 3×3 and 1×1 convolution interleaved with ReLU non-linearity. The output of the branch is movement map $\hat{m}^{I_t^T} \in \mathbb{R}^{\frac{H}{R} \times \frac{W}{R} \times 2T}$, where $2T$ denotes the center offsets (in X and Y directions) sequentially for $I_t^1, I_t^2, ..., I_t^T$ with respect to action centers at I_t^T as references.

For training, groundtruth action centers at $I_t^1, I_t^2, ..., I_t^T$ are first computed the same way as in Center branch. Then, the groundtruth movement ($m^{I_t^T}$) of any action instance with respect to I_t^T is simply the offset between its center at I_t^T and those at other frames. Finally, movement map $\hat{m}_i^{I_t^T}$ is optimized based on L1 loss as follows:

$$l_{Trajectory} = \frac{1}{n} \sum_{i=1}^{n} |\hat{m}_i^{I_t^T} - m_i^{I_t^T}|, \tag{5.5}$$

where i indicates the i^{th} out of n action instances.

During inference, Center Branch obtains action centers at the end of the input sequence as references, while Trajectory branch adjusts all action centers at the end of each clip according to the predicted offsets with respect to the reference centers. Note that the predicted center offset at I_t^T from itself is expected to be zero; as a result, we do not adjust action centers at I_t^T.

5.5.3 Box branch

Box branch serves to regress the spatial extent of action instances at $[I_t^1, I_t^2, ..., I_t^T]$, whose locations have been previously deduced by Center and Trajectory branch. Unlike the other two branches, incorporating temporal information from multiple frames intuitively does not benefit frame-wise prediction of class-agnostic bounding boxes. Hence, our Box branch regresses actions' width and height for each clip independently. It comprises a 3×3 and 1×1 convolutional layer in sequence (interleaved with ReLU) as the other branches, and generates spatial prediction map $\hat{s} \in \mathbb{R}^{\frac{H}{R} \times \frac{W}{R} \times 2}$, where 2 corresponds to the height and width prediction. As Box branch is shared by all T clip-level features, it outputs T spatial maps, each one being associated with the size prediction at $I_t^1, I_t^2, ..., I_t^T$. We optimize this branch by summing the L1 loss at all clips as follows:

$$l_{Box} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{T}|\hat{s}_i^j - s_i^j|, \tag{5.6}$$

where s_i^j corresponds to the groundtruth height and width of the i^{th} action instance (out of n instances) that belongs to the j^{th} clip.

5.5.4 AMMA - loss

The overall training objective of AMMA is shown in Equation 5.7, where hyperparameter a, b, and c are set to 1, 1, and 0.1 respectively in accordance with [83]:

$$l_{AMMA} = al_{Center} + bl_{Trajectory} + cl_{Box}. \tag{5.7}$$

5.6 Online detection and tube generation

5.6.1 Incremental detection via feature-caching-dequeueing

Our proposed action detector requires only RGB frames as input. As it generates motion representations on-the-fly, AMMA can be applied directly to real-time video streams. To efficiently and continuously handle incoming video frames, we employ a simple feature-caching mechanism that allows AMMA to focus on extracting relevant features only from the current clip while still being able to retrieve clip-level features from the past for spatiotemporal reasoning. Figure 5.3 illustrates such an online detection workflow.

In detail, given that T clips are needed for action inference, AMMA's backbone initially obtains T clip-level features from which action tubelets are produced at the detector branches. Meanwhile, the T clip-level features are also cached in AMMA's buffer. Once enough incoming frames are gathered as a valid new clip (i.e., reaching t frames), our detector only extracts the T^{th} clip-level features from this new clip. The past $T-1$ clip-level features can be efficiently retrieved from the buffer and combined with the current one, from which the detector branches predict new action tubelets. Clip-level feature-caching and dequeuing enable AMMA to incrementally infer action tubelets covering past and incoming new frames while processing only the newly arrived clip. During video streaming, AMMA's buffer will continue to be updated accordingly to keep the latest T features.

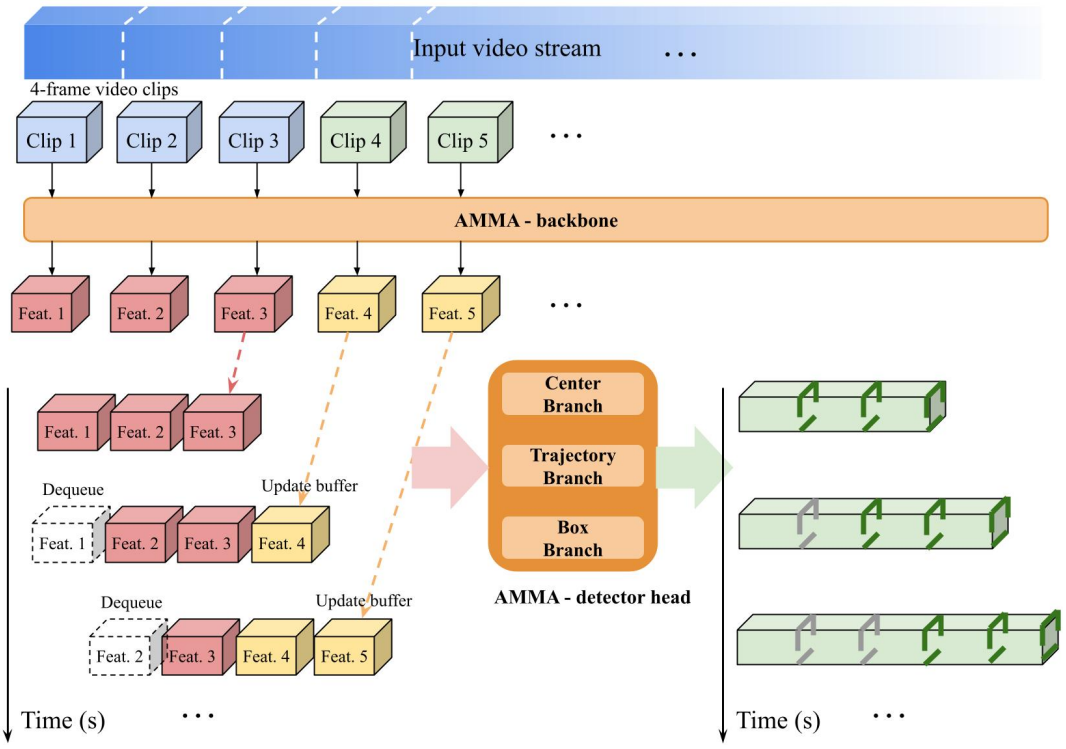

FIGURE 5.3: AMMA's incremental feature-caching-dequeueing mechanism for online action detection on video streams.

5.6.2 Linking coarse tubelets into action tubes

Given an incoming video stream, AMMA detects tubelets on top of the latest T clips. Notably, the lastly detected tubelets have a temporal overlap with the previous ones by $(T-1)$ clips (as illustrated in the bottom-right corner of Figure 5.3). When tubelet results within these temporal overlaps is consistent, AMMA can incrementally link local tubelets over time into action tubes, yielding long-range spatiotemporal action detection for trimmed or untrimmed video.

We adopt the same online tubelet linking algorithm as introduced in Chapter 4. To recap, given a video stream as input, AMMA detects N initial tubelets. Among these tubelets, the top ten tubelets with the highest confidence scores are kept as "active" action links for subsequent tubelet linking. As the video continues to be streamed, we incrementally extend active links with new tubelet candidates if their detections at corresponding temporal positions match. i.e., the average IoU exceeds threshold $\tau = 0.5$. It is noteworthy that each candidate tubelet can only be assigned to an active link. On the other hand, an active link stops extending and is terminated ("inactive") either when there no longer exists temporal overlap with the newly detected tubelets, or the video stops being streamed.

The final action tubes are constructed from all the inactive action links, where each tube's confidence score is calculated as the average score of all its enclosed tubelets. The temporal extent of any action tube is determined by the starting frame of the initialized tubelet and the end frame of the last tubelet. Lastly, we discard any final action tube having either a low confidence score or a short temporal duration. Note that AMMA shares the same coarse-detection scheme as that of TEDdet from Chapter 4. To acquire dense frame-wise detection, we apply coordinate-wise linear interpolation between bounding boxes located at two separate clips to infer detection for intermediate frames. This design form is reasonable as transitions of actions across consecutive frames are typically smooth and continuous.

5.7 Experimental validation

Following the same evaluation protocol presented in Chapter 4, we investigate various architectural configurations of AMMA on UCF-24 and JHMDB-21. For efficient exploration, the following studies are conducted based on ResNet18 unless specified otherwise. As a reminder, frame-mAP and video-mAP are used to evaluate the performance of our detector. The former metric validates the IoU between the detected and groundtruth boxes at each frame and is independent of the online linking strategy. For frame-mAP, the IoU threshold (τ) is fixed at 0.5 throughout all experiments. On the other hand, video-mAP inspects spatiotemporal overlaps between linked action tubes and groundtruth tubes at multiple IoU thresholds (τ = [0.2, 0.5, 0.75, 0.5:0.95:0.05]). Furthermore, to evaluate the efficiency of AMMA, we also report its model size (number of trainable parameters), MACs (number of multiply-accumulate operations), and speed (FPS: frame-per-second).

5.7.1 Implementation details

We implement AMMA in Pytorch [105]. Aiming to conduct highly accelerated and efficient detection, we adopt ResNet18 [19] as AMMA's main CNN backbone. All inputted RGB frames to our model are resized to 288×288. AMMA's backbone includes an encoder-decoder feature extractor followed by a bilinear upsampling layer, transforming video clips to clip-level representations of dimension $36 \times 36 \times 256$. Prior to AMMA's detector head, clip-level features are first fed to a 1×1 convolutional layer to reduce their

channel dimension by 4 in order to gain efficiency at Center and Trajectory branches (who work on channel-wise stacked features).

Within AMMA's backbone, spatial and temporal information are combined via uni-lateral fusion. In the case of ResNet18, we establish uni-lateral connections at the "stage" level. To investigate the influence of fusing micro-motion and RGB features at different scales, we vary the extent of fusion by incrementally adding a lateral connection at the output of each stage (up to five connections for ResNet18).

To verify our detection framework on ultra-lightweight architectures for resource-constrained devices, we also evaluate its integration with MobileNetV2 [108] and ShuffleNetV2 [109]. The weights of all 2D CNN backbones are initialized with COCO pretrain (except for ShuffleNetV2 using ImageNet pretrain).

During training, we apply common practices of data augmentation such as photometric transformation, scale jittering, random cropping/expansion and location jittering, etc. To train AMMA on T-clip sequences, each action tubelet is expected to last $T \times t$ frames. In practice, observing that consecutive video frames occasionally repeat (in JHMDB-21), we omit sampling the second-last frame of each clip to avoid accumulating zero-value micro-motion; hence the actual duration spans $T \times (t+1)$ frames. For any action video having a shorter duration, we pad the beginning of its T-clip sequence by the first frame of the video until the minimum length requirement is met, simulating an action without movement at the beginning. At AMMA's detector head, only the groundtruth associated with the final frame of a clip (i.e., I_t^T) is used to train Center branch. On the other hand, Trajectory branch and Box branch learn to regress movement and spatial extent of action instances over all clips, thus requiring groundtruth labels of $I_t^1, I_t^2, ..., I_t^T$.

We use the Adam optimizer to train our models. An initial learning rate of $5e^{-4}$, $2.5e^{-4}$, and $2.5e^{-4}$ is applied when employing ResNet18, MobileNetV2 and ShffleNetV2 as AMMA's backbone, respectively. For JHMDB-21, we train AMMA for 10 epochs while reducing the learning rate by a factor of 10 at the 6^{th} and 8^{th} epoch. Likewise, UCF-24 is trained for 10 epochs, but with the learning rate reduced by half at every epoch after the second one. In our experiments, all the training is conducted on an NVIDIA Titan V5 GPU while fixing the mini-batch size at 16.

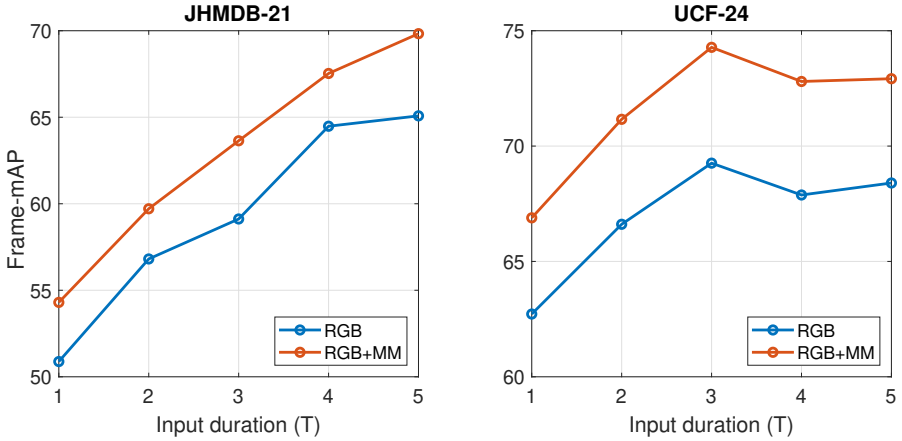

FIGURE 5.4: Frame-mAP performance under varied input duration (i.e., number of clips). Here, "MM" denotes micro-motion.

5.7.2 Effect of input duration

The core of AMMA lies in detecting action tubelets across a sequence of video clips. Intuitively, combining more clips as input encapsulates richer spatiotemporal context. However, longer sequences could potentially introduce irrelevant background cues, as well as raising difficulty to track tubelets' trajectories. To investigate how the input duration affects the proposed detector, we conduct experiments on both JHMDB-21 and UCF-24 by varying the number of input clips under two setups (with or without micro-motion fusion). To control the extent of motion fusion in these experiments, we uniformly attach one lateral connection at the end of the first three stages of ResNet18. The corresponding frame-mAP results are depicted in Figure 5.4.

From the above experiments, we observe that AMMA generally produces more accurate tubelets the longer video sequences it sees. This result matches our hypothesis that reasoning from longer video clips enriches spatiotemporal feature learning. In contrast, the configuration of $T = 1$ without micro-motion is essentially frame-wise detection, which is the least accurate due to a complete lack of temporal modeling (incorporating neither short-term dynamic nor long-term appearance evolution).

Notably, AMMA's accuracy continues to benefit on JHMDB-21 as T increases. We observe that longer input sequences improve accuracy mainly by reducing false-positive detection in videos where ambiguous visual cues are present. Figure 5.5 displays several examples where AMMA manages to detect correctly when enlarging its temporal receptive field across longer video sequences. We stop increasing the number of clips at 5. When $T = 5$, AMMA takes $5 \times (4 + 1) = 25$ consecutive frames as input, which are more

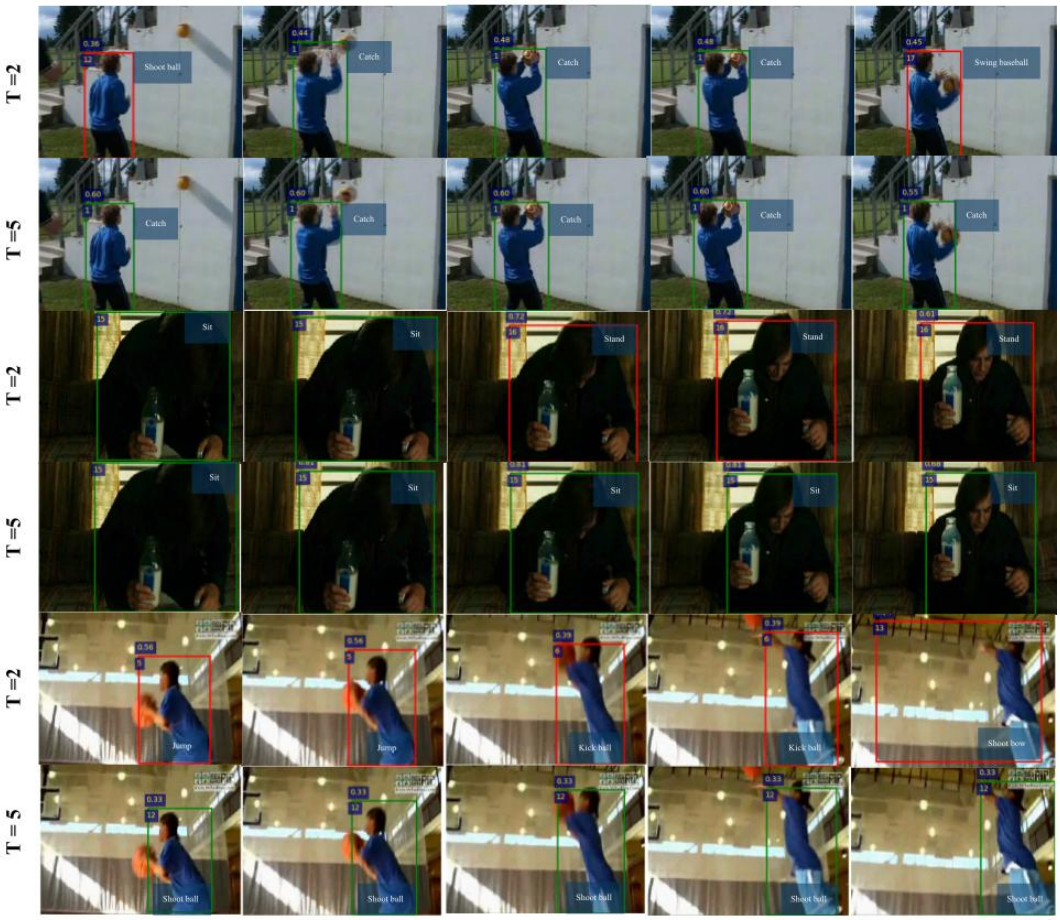

FIGURE 5.5: Examples of short-tubelet ($T = 2$) and long-tubelet ($T = 5$) detection on JHMDB-21. The groundtruth actions (from top to bottom) are Catch, Sit, and ShootBall. The green and red boxes correspond to correct and incorrect detection, respectively. Each colored box also displays the detected class and associated confidence score. Longer input sequences help to reduce false-positive detection which are prone to occur in the presence of ambiguous visual cues (e.g., confusion between Sit and Stand, Catch and ShootBall, etc.)

than half of the frames in most videos of this dataset. We adopt a similar setup when evaluating UCF-24. Interestingly, although input duration and accuracy still positively correlate, AMMA performs best when $T = 3$. We deduce that as UCF-24 consists of temporally untrimmed videos, AMMA is prone to produce more false-positive detection associated with time when assigning a unified action label to a longer sequence, i.e., making predictions on frames without an action. Alongside varied input duration, all architectures with micro-motion feature fusion consistently outperform those using only appearance cues, confirming the efficacy of introducing short-term dynamic motion to help differentiate actions.

TABLE 5.1: Performance summary of different forms of micro-motion fusion on JHMDB-21. Input duration is fixed to 5 clips, and three lateral connections are attached to the outputs of the first three stages in ResNet18.

	Frame-mAP	GMACs	# param. (M)
RGB only	65.08	3.51	15.07
RGB + MM_{Diff}	67.69	4.95	15.75
RGB + MM_{Conv_Diff}	69.74	5.21	15.75

5.7.3 Effect of micro-motion generation and fusion

The previous experiment demonstrates AMMA's extensible temporal modeling capacity with micro-motion feature fusion. We further investigate the influence of different forms of micro-motion generation and fusion.

Table 5.1 summarizes AMMA's detection accuracy and complexity on JHMDB-21 in accordance with varied input forms. Building upon the input-duration experiment, we adopt 5-clip input and three stages of lateral connection (when fusion is applied). To approximate our model's complexity under a streaming-video setup, we report the MACs of tubelet detection over T clips and then divide by T. In addition, to verify the necessity of our micro-motion sub-network (expressed as MM_{Conv_Diff}), we implement a simpler micro-motion variant (coined MM_{Diff}) by directly accumulating RGB difference maps.

Our experiment shows that with minor increase in the model size, fusing MM_{Conv_Diff} features largely enhances AMMA's accuracy from that of only using RGB frames by nearly 5 frame-mAP. Relative to the model size, which arises due to adding the micro-motion sub-network and duplicating three early stages of ResNet18, the elevation is more prominent in the required GMACs. We found that the additional operations associated with fusing MM_{Conv_Diff} all take place toward early layers of AMMA's backbone where target tensors still retain large spatial dimensions, resulting in a more noticeable raise in multiply-accumulate computation than in model size. On the other hand, fusing MM_{Conv_Diff} obtains higher accuracy than MM_{Diff} by more than 2 frame-mAP at a negligible increase in GMACs and model size. This suggests that the temporal evolution of general patterns better encodes motion dynamics than raw RGB differences, which are more likely to carry local noises. In Figure 5.6, we show some examples of our micro-motion representation which successfully captures motion boundaries near moving actors.

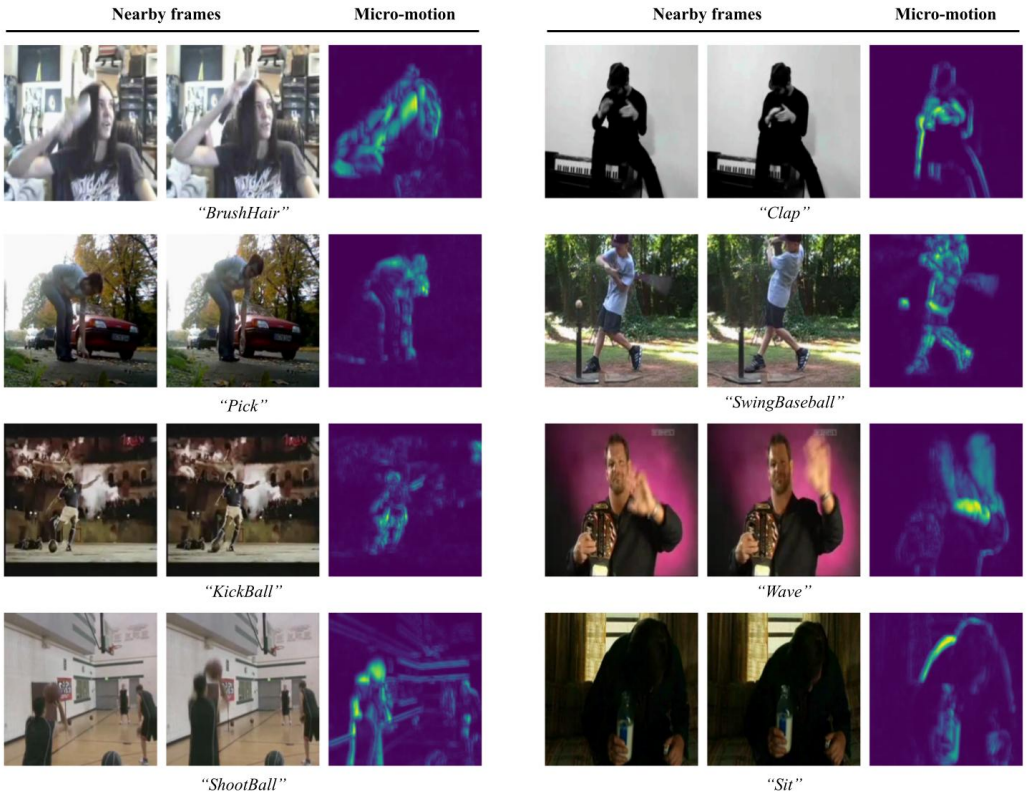

FIGURE 5.6: Visualization of micro-motion cues between pairs of action frames.

Next, we explore different extents of fusion between appearance and micro-motion information by incrementally raising the number of lateral connections. As shown in Table 5.2, the more stages of lateral fusion take place, the more accurate AMMA becomes. This correlation indicates that our accumulated micro-motion embeds rich temporal cues, from simple motion boundaries to abstract dynamic information. Multi-scale lateral fusion therefore ensures AMMA to simultaneously learn complementary spatiotemporal information throughout the backbone. In exchange for enhanced accuracy, more lateral fusion inherently raises the model size and computation associated with extracting micro-motion features at deeper layers. To conclude, AMMA's capacity to jointly model actions' visual and dynamic information can be improved when adopting deep fusion. In spite of that, with the global aim of keeping an efficient detection architecture, we continue leveraging 3 stages of lateral fusion throughout the rest of the experiments.

TABLE 5.2: Performance summary of varied extents of fusion between appearance and micro-motion features on JHMDB-21. Input duration is fixed to 5 clips.

	Frame-mAP	GMACs	# param. (M)
—	65.08	3.51	15.07
Stage1	66.58	3.76	15.08
Stage1,2	68.74	4.53	15.23
Stage1,2,3	69.74	5.21	15.75
Stage1,2,3,4	70.22	5.89	17.85
Stage1,2,3,4,5	72.48	6.57	26.24

5.7.4 From lightweight to ultra-lightweight

Ultimately aiming at deploying the detector onto resource-constrained devices, we examine AMMA's generalization ability on ultra-lightweight mobile architectures: MobileNetV2 and ShuffleNetV2. Both detection accuracy (e.g., frame-mAP and video-mAP) and model efficiency (e.g., inference speed, model complexity, and size) are assessed. In particular, speed is recorded based on the per-frame processing time of the entire action detection pipeline, i.e., the total runtime of generating action proposals for all videos divided by the total number of their frames.

Integration of micro-motion and lateral fusion in these mobile architectures closely follows our design with the ResNet18. With MobileNetV2 as the backbone, we append three lateral connections at the output of the 1^{st}, 3^{rd}, and 6^{th} bottleneck residual block (MobileNetV2 consists of 17 of these building blocks). For ShuffleNetV2, three lateral connections are established at the output of "Conv1", "Stage2", and "Stage3" (naming conventions of these layers/blocks follow those in [109]). The resulting models are represented by AMMA_{18} (ResNet18), AMMA_{M} (MobileNetV2), and AMMA_{S} (ShuffleNetV2) for simplicity. Note that we apply different clip-lengths on the two datasets following the best configuration found in Figure 5.4.

Results of the three AMMA variants are reported in Table 5.3. We observe that AMMA_{18} consistently obtains higher accuracy than the other two (especially reflected in video-mAP at high detection thresholds). This is expected as ResNet has higher capacity to extract richer visual context in general than the mobile architectures prioritizing efficiency. Indeed, both datasets consist of actions embedding prominent appearance cues such as ShootBow and PoleVault that could benefit from a more powerful feature extractor. In terms of efficiency, the average GMACs of AMMA_{M} and AMMA_{S} are approximately $1/4$ and $1/5$ of that of AMMA_{18}. Similarly, the model size of AMMA_{M} and AMMA_{S} are also significantly smaller.

TABLE 5.3: Performance summary of integrating different 2D CNN backbones.

	JHMDB-21 (T=5)							
	F-mAP	Video-mAP				GMACs	Param. (M)	FPS
		@0.2	0.5	0.75	0.5:0.95			
$AMMA_{18}$	69.7	73.7	72.7	60.1	50.3	5.2	15.8	80
$AMMA_{M}$	66.1	70.0	69.0	53.7	45.3	1.3	6.8	77
$AMMA_{S}$	67.7	72.3	70.9	47.9	43.0	1.0	6.0	75
	UCF-24 (T=3)							
	F-mAP	Video-mAP				GMACs	Param. (M)	FPS
		@0.2	0.5	0.75	0.5:0.95			
$AMMA_{18}$	74.6	81.1	53.5	24.6	26.3	5.2	15.8	115
$AMMA_{M}$	71.8	78.0	49.7	22.0	23.5	1.3	6.8	110
$AMMA_{S}$	71.3	78.7	47.4	20.9	22.5	1.0	6.0	100

Countering the above observations, the two ultra-lightweight variants have slightly slower runtime than $AMMA_{18}$ even though their computational cost is substantially lower. On the one hand, this phenomenon has been addressed in [110], which points out that the implementation of depth-wise separable convolution is not optimized in the cuDNN library (therefore, MobileNetV2 tends to be slower than ResNet18 in standard experimental setups). Moreover, computational complexity does not necessarily guarantee faster runtime as GMACs do not take into account factors such as memory access cost and platform characteristics [109]. Further, even though all three AMMAs are equipped with three lateral connections, their extent of spatial-temporal fusion still differs according to the architectural designs of their backbone CNNs. For example, ShuffleNetV2 has more convolutional layers in "Stage3" than those in ResNet18 to process motion cues. All of our models still exceed real-time performance by a large margin.

Interestingly, it can be observed that AMMA's runtime varies significantly between the two datasets. This is attributed to videos having drastically different lengths in these datasets. Videos in JHMDB-21 are generally short and thus benefit less from the feature-caching mechanism, as AMMA's buffer will be frequently cleared and await to be initialized by features from new videos. In addition, each initialization takes more time for JHMDB-21 than UCF-24, since JHMDB-21 is configured to take a longer input sequence for action tubelet inference (pending for 5 clip-level features rather than 3).

TABLE 5.4: Comparison with the state-of-the-art methods. Under column "Input", "+OF" indicates using optical flow as the additional input modality (alongside RGB input).

Method	Input	JHMDB-21					UCF-24				
		F-mAP	Video-mAP				F-mAP	Video-mAP			
			0.2	0.5	0.75	0.5:0.95		0.2	0.5	0.75	0.5:0.95
2D backbone											
Saha et al. [70]	+OF	−−	72.6	71.5	43.3	40.0	−−	66.7	35.9	7.9	14.4
Peng et al. (MR) [71]	+OF	58.5	74.3	73.1	−−	−−	−−	73.5	32.1	2.70	7.30
Saha et al. (AMTnet) [77]	+OF	−−	73.5	72.8	59.7	48.1	−−	78.5	49.7	22.2	24.0
Kalogeiton et al. (ACT) [75]	+OF	65.7	74.2	73.7	52.1	44.8	69.5	76.5	49.2	19.7	23.4
Singh et al. (ROAD) [73]	+OF	−−	73.8	72.0	44.5	41.6	−−	73.5	46.3	15.0	20.4
Yang et al. (STEP) [97]	+OF	−−	−−	−−	−−	−−	75	76.6	−−	−−	−−
Zhao et al. (Two-in-One) [84]	+OF	−−	−−	74.7	53.3	45.0	−−	78.5	50.3	22.2	24.5
Song et al. (TACnet) [112]	+OF	65.5	74.1	73.4	52.5	44.8	72.1	77.5	52.9	21.8	24.1
Zhang et al.[106]	+OF	37.8	−−	−−	−−	−−	67.7	74.8	46.6	16.7	21.9
Li et al. (MOC) [83]	+OF	68.0	76.2	75.4	68.5	54.0	76.9	81.3	54.4	29.5	28.4
AMMA$_{18}$	---	69.7	73.7	72.7	60.1	50.3	74.6	81.1	53.5	24.6	26.3
AMMA$_{M}$	---	66.1	70.0	69.0	53.7	45.3	71.8	78.0	49.7	22.0	23.5
AMMA$_{S}$	---	67.7	72.3	70.9	47.9	43.0	71.3	78.7	47.4	20.9	22.5
3D backbone											
Hou et al. (T-CNN) [79]	---	61.3	78.4	76.9	−−	−−	67.3	73.1	−−	−−	−−
Gu et al.[4]	---	73.2	−−	−−	−−	−−	77.0	−−	−−	−−	−−
Qiu et al.[113]	---	−−	77.3	74.2	−−	−−	−−	69.3	−−	−−	−−
Zhao et al. (TubeR) [89]	---	−−	−−	79.5	−−	58.0	−−	−−	52.0	−−	25.2

5.7.5 Global detection performance and comparison

In this section, we evaluate AMMA against several state-of-the-art methods on JHMDB-21 and UCF-24. The comparison follows the same principle as stated in previous chapters: since our tubelet detector not only seeks competitive accuracy, but also requires low complexity and real-time runtime for compact deployment, only state-of-the-arts with loosely comparable architectures as AMMA are listed in Table 5.4. Other top-performing approaches [90][111] which employ much heavier detection frameworks (e.g., two streams of 3D CNN) are excluded for fair comparison.

It can be observed from Table 5.4 that AMMA$_{18}$ achieves competitive accuracy on both datasets. Notably, our proposed model utilizes the most lightweight feature backbone than all other methods on the list. These include two-stream VGG16, two-stream DLA-34, C3D, I3D, and S3D, etc. Furthermore, leveraging only RGB frames as input, AMMA$_{18}$ still outperforms most of the other two-stream methods relying on fine-grained optical flow, (especially reflected in its frame-mAP and video-mAP at high detection thresholds). On the other hand, we notice that AMMA tends to keep a higher number of false-positive action tubes after the tubelet-linking process, decreasing its video-mAP at lower IoUs (more evident on JHMDB-21). This is likely caused by AMMA's coarse-detection scheme which assumes a unified action

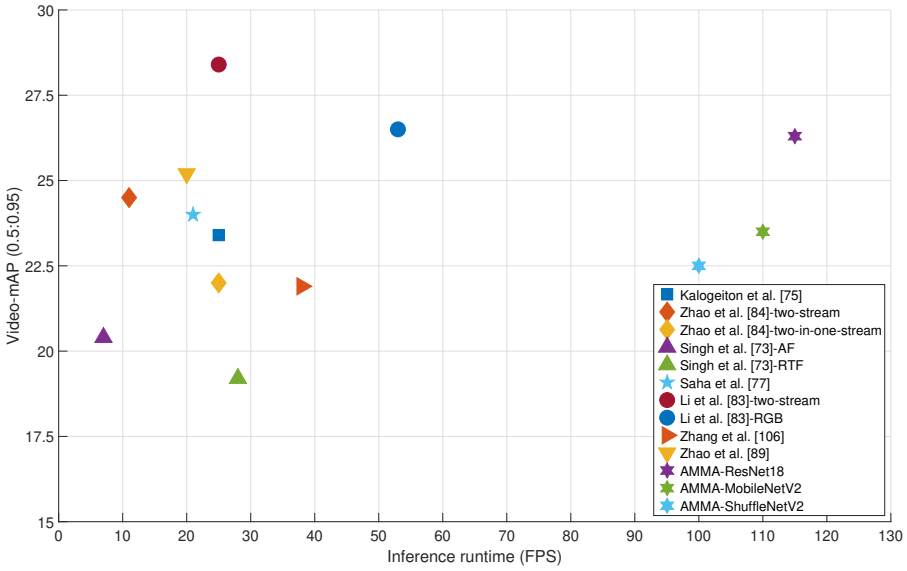

FIGURE 5.7: Comparisons of runtime-accuracy trade-off between AMMA and state-of-the-arts on UCF-24 (video-mAP). "AF" and "RTF" denote accurate flow and real-time flow, respectively. It is note-worthy that methods that depend on externally calculated optical flow typically omit this part of the computation in their runtime measurement.

class over an extended duration for any action instance. Consequently, unlike dense-tubelet detectors [75][97][83], it is more difficult to suppress false-positive detection simply by discarding short action tubes at the end. Finally, AMMA_{18} scores competitively against most 3D CNN-based methods even though ResNet18 has far less capacity to reason spatiotemporal information, indicating the effectiveness of fusing coarse-scale visuals and complementary dynamic cues. For AMMA_{M} and AMMA_{S}, due to their CNN backbones being less capable of abstracting visual patterns in exchange for substantially lower computational cost, there remains a perceivable margin from the accuracy of other top-performing detectors.

Beyond competitive accuracy, the evident strength of AMMA lies in its cost-effective architecture and workflow tailored for real-world scenarios and deployment. Specifically, the vast improvement in AMMA's processing efficiency is attributed to its coarse-detection paradigm as well as being free of optical flow extraction. The former not only bypasses redundancy associated with dense per-frame detection, but also facilitates capturing actions' prominent appearance variation over time. Adopting on-the-fly motion cues instead of pre-computed optical flow, AMMA supports detecting actions in an

online manner from video streams when exploiting feature-caching and interpolation from coarse-level detection. As shown in Figure 5.7, while retaining competitive video-mAP on UCF-24, our models considerably outperform other action detectors reporting real-time or near-real-time performance.

5.8 Summary and limitations

Summary. In this chapter, we propose a cost-effective 2D-based tubelet detection framework coined Accumulated Micro-Motion Action detector (AMMA). It adopts a coarse-level detection paradigm which sparsely extracts visual and complementary dynamic cues of actions spanning a longer sequence. Lifting reliance on optical flow, AMMA utilizes accumulated micro-motion to encode actions' short-term motion dynamics, which can be efficiently learned and generated on-the-fly from RGB frames. On top of AMMA's motion-aware CNN backbone, we adopt the state-of-the-art anchor-free detector, cooperatively modeling action instances' categories, locations, sizes, and trajectories as moving points in the time span. When compared to other methods, the proposed action tubelet detector achieves competitive accuracy on UCF-24 and JHMDB-21 benchmarks, while incurring substantially lower computation and exceeding real-time speed by multiple folds.

The main contributions of our work can be summarized as follows:

- We propose an integrated action tubelet detector which builds upon lightweight 2D CNN feature backbones (ResNet18, MobileNetV2, and ShuffleNetV2) and an anchor-free detector. It follows a coarse-to-fine detection paradigm, enabling learning rich spatiotemporal cues of actions over larger temporal fields.

- We design a novel micro-motion representation to efficiently encode and accumulate subtle motion dynamics of actions on-the-fly, replacing optical flow and permitting real-time detection over online video streams.

- Our detailed ablation studies on public JHMDB-21 and UCF-24 datasets demonstrate AMMA's competitive accuracy (both frame and video-mAP) at significantly lower computation and faster runtime.

Limitations. Producing action tubelets in a timely coarse manner, AMMA demonstrates being a more generic detector that performs better spatiotemporal localization than TEDdet in both frame and video-mAP. However, the

"global" mAP metric does not fully reflect the strengths/limits of our newly proposed detector. Does the addition of motion information help AMMA to better localize moving targets with weak visual cues? Or does distinct motion pattern facilitate more confident classification? From the perspective of real-world deployment, AMMA's transparency and its ability to scale under more complicated actions and dynamic scenarios require further investigation.

Looking ahead. In the next chapter, we continue to examine AMMA's performance at a deeper level. Mainly, the global accuracy evaluation metric, frame-mAP, is decomposed into mutually-exclusive sources of error categories to help understand what AMMA struggles to detect. Similar analysis on the computation and time cost of all variants of AMMA as well as TED-det, will also be conducted to identify potential computational bottlenecks for future design references.

Chapter 6

Diving more deeply into AMMA

6.1 Introduction

In Chapter 5, we propose a unified action tubelet detector coined AMMA. It utilizes approximated motion representations calculated on-the-fly from RGB frames, lateral feature fusion to combine spatial and temporal cues, cooperative detector branches, and a coarse-to-fine detection paradigm for accelerated detection. Our experimental results attest that AMMA, even when paired with lightweight CNN backbones (e.g., ResNet18, MobileNetV2, and ShuffleNetV2) that favor deployment on resource-constrained devices, retains highly competitive accuracy while largely excelling in detection speed ($\geq$ 75 FPS).

In this chapter, we further investigate AMMA in terms of its accuracy, computational complexity, and speed performance. To assess accuracy, the previous chapter mostly provides numeric metrics (e.g., frame-mAP or video-mAP) which partially disclose the underlying performance of the detector on a global scale. Alternatively, we take a step back and attempt to address the following question: **what contributes to AMMA's mAP score on designated datasets?** In other words, **what are the sources of errors that lead to mAP loss?**

Besides re-evaluating the accuracy, we also carry out complexity and runtime breakdown analysis to identify computational bottlenecks (if any) and comprehend the time-consumption of each working module. Detailed analysis and comparison are conducted among different AMMA variants, as well as TEDdet introduced in Chapter 4.

Note that ACDnet (presented in Chapter 3) has been excluded for comparison in this chapter. ACDnet is one of our earliest work whose detection

pipeline substantially differs from those of AMMA and TEDdet. These discrepancies include the line of adopted detectors (anchor-based vs. anchor-free), the way to obtain frame-wise detection (motion-guided feature propagation vs. coarse-to-fine detection accompanied by intra-frame interpolation), whether tube linking is supported (only by TEDdet and AMMA), implementation frameworks (MXNet vs. Pytorch), and an inconsistent evaluation protocol (UCF-24 was temporally trimmed prior to evaluation in Chapter 3). In spite of omitting this piece of comparison, our in-depth analysis on AMMA and TEDdet's performance shall suffice to make their advantages/limitations more transparent and palpable than simply reporting the benchmark accuracy. We believe our findings will provide meaningful and seminal insights for future designs of real-time spatiotemporal action detectors in the research community.

Outline. This chapter is organized as follows. We begin by analyzing the breakdown of errors contributing to AMMA's frame-mAP loss in Section 6.2. In this section, the global error distribution and class-wise AP are assessed quantitatively and qualitatively on both datasets. Second, we look into the computational cost of our detectors in Section 6.3, on a global scale as well as down to individual working modules. Next, Section 6.4 re-evaluates and compares the inference speed of our detection models. The relation of tubelet length and runtime is also uncovered in this section. Finally, we present an assessment summary with brief insights into plausible improvements in Section 6.5.

6.2 Accuracy and error breakdown analysis

6.2.1 Recap on AMMA and TEDdet

We first revisit the frame-mAP and video-mAP acquired by AMMA (Chapter 5) and TEDdet (Chapter 4). Throughout the chapter, variants of AMMA are represented by AMMA_{18}, AMMA_{M}, and AMMA_{S}, each employing ResNet18, MobileNetV2, and ShuffleNetV2 as the feature extractor backbone, respectively. Here, we only cover the best configuration of each method from the previous chapters unless specified otherwise.

In Table 6.1, one can see that AMMA_{18} consistently outperforms the other variants on both datasets by approximately 2-3 frame-mAP. This is expected as MobileNetV2 and ShuffleNetV2 are tailored to prioritize efficiency for low-end devices such as mobile platforms. Such superiority in accuracy is

TABLE 6.1: Accuracy recap on AMMA and TEDdet. The below results correspond to the best performing models found in Chapter 4 and 5.

	JHMDB-21					
	Input length (T)	Frame-mAP @0.5	Video-mAP @0.2	0.5	0.75	0.5:0.95
$AMMA_{18}$	5	69.7	73.7	72.7	60.1	50.3
$AMMA_{M}$	5	66.1	70.0	69.0	53.7	45.3
$AMMA_{S}$	5	67.7	72.3	70.9	47.9	43.0
TEDdet	5	64.7	67.9	67.4	53.7	44.7
	UCF-24					
	Input length (T)	Frame-mAP @0.5	Video-mAP @0.2	0.5	0.75	0.5:0.95
$AMMA_{18}$	3	74.6	81.1	53.5	24.6	26.3
$AMMA_{M}$	3	71.8	78.0	49.7	22.0	23.5
$AMMA_{S}$	3	71.3	78.7	47.4	20.9	22.5
TEDdet	5	70.8	74.6	50.4	21.8	25.0

also prominently shown in video-mAP at higher detection thresholds, especially reflected on JHMDB-21. We also observe that $AMMA_{M}$'s video-mAP significantly exceeds that of $AMMA_{S}$ at higher thresholds, which can be associated with more precise localization. Since the two models apply a slightly different training protocol ($AMMA_{S}$ is initialized with ImageNet instead of COCO pretrain weights), we believe that the discrepancy can be mitigated.

When compared to TEDdet, which also leverages ResNet18 as the feature extraction backbone, $AMMA_{18}$ demonstrates much higher capacity to distinguish among actions and generate better action proposals, mainly due to the addition of micro-motion feature fusion. The benefit of incorporating motion information can also be inferred from $AMMA_{M}$ and $AMMA_{S}$, which both obtain higher frame-mAP and comparable video-mAPs than TEDdet even though employing much more compact feature extractors.

6.2.2 Unveiling frame-mAP

Frame-mAP and video-mAP convey the global precision of our detectors with respect to state-of-the-art methods. Of the two metrics, video-mAP reflects the tubelet linking performance based on IoU matching, which can be considered a measure of consistency on detected tubelets over time. On the other hand, frame-mAP gives insights on how well our detection models infer action tubelets from video clips. To better understand predicted tubelets and their accuracy, we examine five mutually exclusive factors of possible

errors and analyze which percentage of the mAP is lost due to each of them. The five sources of errors are described as follows:

1. Localization error (E_L): a detection is in a frame containing the correct class, but its localization is incorrect (i.e., IoU < τ with the groundtruth box).

2. Classification error (E_C): a detection has IoU > τ with the groundtruth box of another action class.

3. Time error (E_T): a detection is in an untrimmed video for the correct class, but the groundtruth temporal extent of the action does not cover this frame.

4. Other error (E_O): a detection appears in a frame without the class and has IoU < τ with groundtruth boxes of any other class.

5. Missing detection error (E_M): there is no detection for a groundtruth box.

Among the different types of errors, E_M refers to false-negative detection, which is computed by measuring the percentage of missed detections, i.e., ratio of groundtruth boxes for which there are no correct detections. In contrast, E_L, E_C, E_T, and E_O all correspond to false positive detections. Throughout the following error breakdown analysis, we fix τ at 0.5 in accordance with the threshold used for computing frame-mAP throughout this manuscript.

6.2.3 Evaluation on JHMDB-21

Error breakdown analysis. Figure 6.1 presents the error distribution (%) of AMMA and TEDdet's frame-mAP on JHMDB-21 (only on split 1). At first glance, one can perceive that most of AMMAs' mis-detection comes from E_C while the rest of errors are small and scattered ($E_T = 0$ as JHMDB-21 is a temporally trimmed dataset). Specifically, $AMMA_{18}$ obtains 27.02% of E_C, which is very close to 26.76% of $AMMA_M$ and 27.11% of $AMMA_S$. These results imply that the three AMMA variants are almost equally competent for classification. Thanks to having a more powerful CNN backbone, $AMMA_{18}$'s higher frame-mAP lies in localizing actions more precisely (lower E_L), as well as reducing inconfident, noisy detection (lower E_O) and false-negative detection (lower E_M).

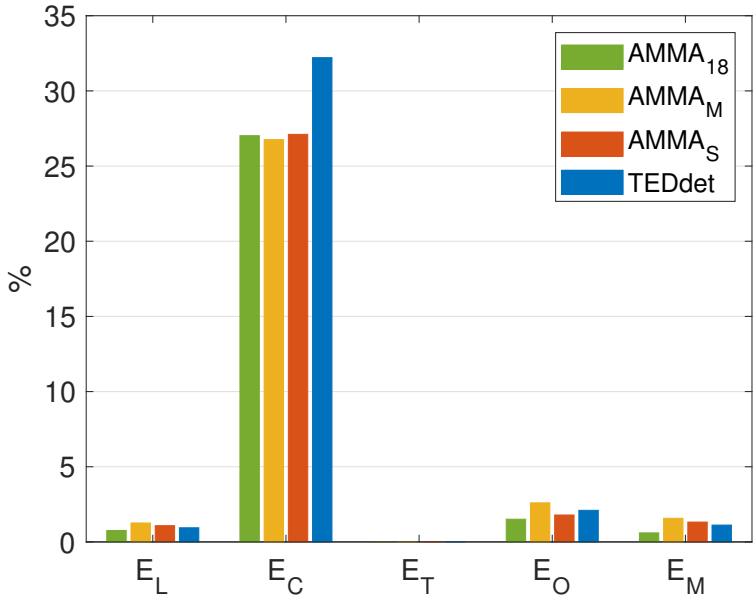

FIGURE 6.1: Error breakdown of AMMA and TEDdet's frame-mAP on JHMDB-21 (only on split 1).

Equipped with the same feature extractor, $AMMA_{18}$ significantly outperforms TEDdet in E_C by nearly 5%, which gives rise to the discrepancy between the two methods' frame-mAP as recapped in Table 6.1. Here, we remind our readers that when excluding micro-motion feature fusion, $AMMA_{18}$ with a sequence length $T = 5$ achieves similar frame-mAP as TEDdet (≈65.0 frame-mAP; see Figure 5.4 for reference). Inherently, integrating micro-motion features plays an essential role to induce learning action-specific dynamics and reduce false-positive classification. Our results are aligned with many previous studies which find optical flow cues particularly useful for JHMDB-21. In a similar spirit, even though TEDdet slightly outperforms both $AMMA_M$ and $AMMA_S$ throughout E_L, E_O (only $AMMA_M$) and E_M due to integrating a more powerful feature extractor, lacking motion-specific cues largely raises its E_C, resulting in lower frame-mAP overall.

Class-specific AP and error. In Table 6.2, we present the complete class-wise performance comparison between $AMMA_{18}$ and TEDdet to better visualize how micro-motion feature fusion influences individual action classes. The first two rows of the table consists of class-wise frame-AP of the two methods, followed by their respective E_C, which is the dominating source of error for JHMDB-21.

As shown in Table 6.2, $AMMA_{18}$ obtains better APs than TEDdet in 17 out of 21 classes. In particular, action classes such as "6-Jump", "12-Run",

TABLE 6.2: Class-wise frame-APs on JHMDB-21 (split 1). All 21 classes are presented in the following order: BrushHair (1-BH), Catch (2-CT), Clap (3-CP), ClimbStairs (4-CS), Golf (5-GF), Jump (6-JP), KickBall (7-KB), Pick (8-PK), Pour (9-PR), PullUp (10-PU), Push (11-PS), Run (12-RN), ShootBall (13-SB), ShootBow (14-SW), ShootGun (15-SG), Sit (16-SI), Stand (17-ST), SwingBaseball (18-SL), Throw (19-TW), Walk (20-WK), and Wave (21-WV).

Actions	1-BH	2-CT	3-CP	4-CS	5-GF	6-JP	7-KB
$AMMA_{18}$ (AP)	90.0	31.0	66.9	81.8	98.7	42.4	66.7
TEDdet (AP)	82.6	10.3	74.0	81.6	92.6	20.4	64.4
$AMMA_{18}$ (E_C)	9.6	67.8	32.2	16.9	1.3	35.5	30.8
TEDdet (E_C)	16.1	86.7	24.9	14.2	7.4	49.3	32.3

Actions	8-PK	9-PR	10-PU	11-PS	12-RN	13-SB	14-SW
$AMMA_{18}$ (AP)	87.9	94.7	98.1	91.0	41.6	52.8	94.1
TEDdet (AP)	84.4	92.6	97.2	94.5	16.7	53.4	81.5
$AMMA_{18}$ (E_C)	10.7	5.3	1.9	7.0	50.8	41.4	5.8
TEDdet (E_C)	15.3	7.5	2.8	5.3	71.2	41.0	18.5

Actions	15-SG	16-SI	17-ST	18-SL	19-TW	20-WK	21-WV
AMMA18 (AP)	66.3	74.9	82.5	76.9	28.0	79.0	27.3
TEDdet (AP)	58.9	71.1	70.6	71.1	23.4	59.7	35.4
AMMA18 (E_C)	33.2	20.3	16.5	23.0	69.8	20.1	67.7
TEDdet (E_C)	40.9	25.6	24.9	28.6	71.0	37.2	56.0

and "20-Walk", which do not depend on scenery cues improve the most from micro-motion (> 15 APs). From the table, it is also apparent that these classes gain accuracy mainly due to lower rates of false-positive classification (E_C). Similar outcomes (but less prominent) are also reflected in other classes such as "1-BrushHair", "2-Catch", "5-Golf", "14-ShootBow", and "19-Throw". Conversely, TEDdet achieves better detection for "3-Clap" and "21-Wave". These two action classes happen to share fairly similar action dynamics, e.g., movement around the hand/arm. We suspect that $AMMA_{18}$ is prone to confusion when seeing similar motion cues around the arms.

Difficult classes. There exist a few action categories that remain challenging for our methods. These actions include "2-Catch", "13-ShootBall", "12-Run", and "19-Throw", whose classification errors are all above 40%. To reason possible causes behind such high false-positive rates in classification, we qualitatively inspect the action data and deduce two potential limitations of our methods, both of which are illustrated in Figure 6.2. First, it is remarked that certain actions (e.g., "2-Catch" and "13-ShootBall") consist of a large number of redundant video frames which lack action-specific pattern (but still labeled as parts of the actions). AMMA's coarse detection scheme

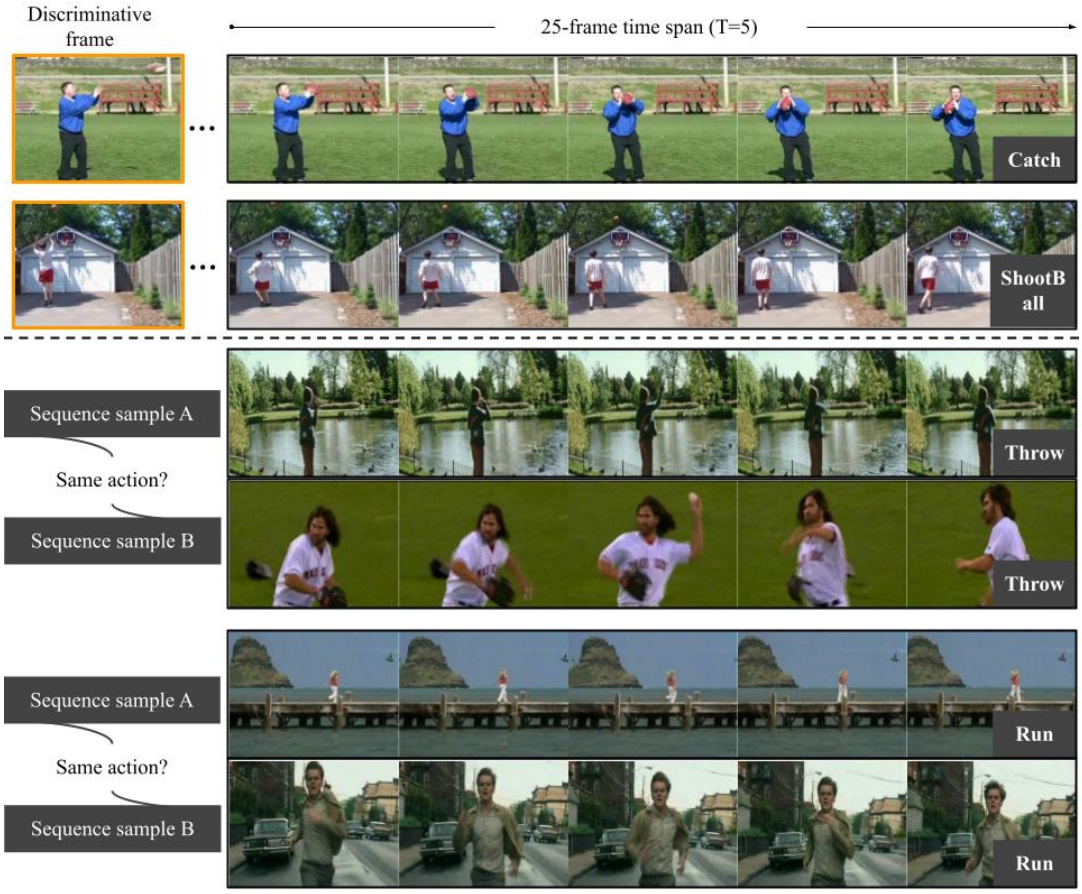

FIGURE 6.2: Examples of difficult action categories from JHMDB-21. (TOP) The selected action sequences for training/testing may not contain "discriminative" frames which prevents AMMA/TEDdet from learning/extracting relevant action cues. (BOTTOM) Strong inter-class variation also poses great challenges for learning generic action-specific pattern. When trained on a small dataset, our models are likely to overfit on non-essential spatiotemporal context.

has a temporal receptive field of 25 frames (when $T = 5$), which is occasionally insufficient to accommodate meaningful context/motion embedded in these actions (as shown in the top rows of the Figure 6.2). Second, we suspect that due to JHMDB-21's small data size, AMMA could not learn well-generalized pattern of specific groups of actions, especially those having drastic inter-class variations (examples are depicted in the bottom rows of the same figure). The problem is likely to be alleviated when more suitable pretrain weights specific to action-related tasks are applied.

6.2.4 Evaluation on UCF-24

We conduct a similar error breakdown analysis on UCF-24 as presented in Figure 6.3. Different from JHMDB-21, the highest frame-mAP loss for UCF-24 originates from time error E_T while the remaining loss is scattered across

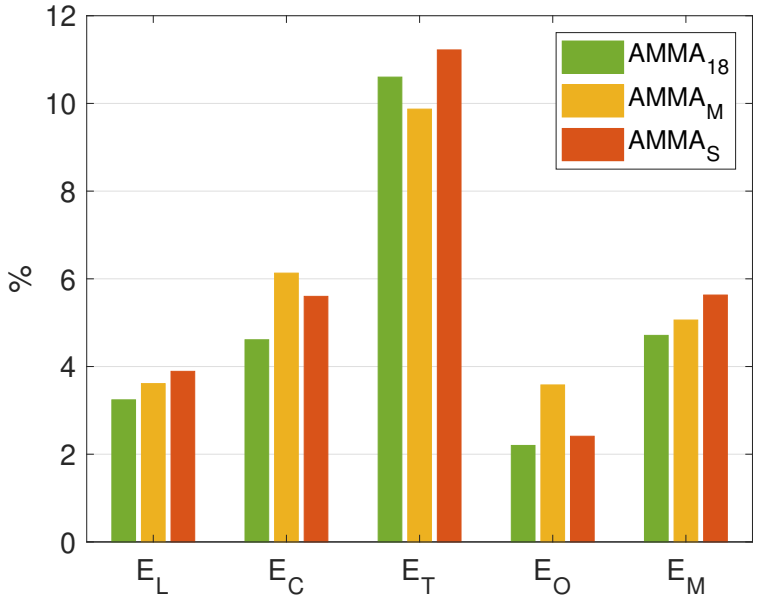

FIGURE 6.3: Error breakdown of AMMA's frame-mAP on UCF-24.

all other types of error.

Among different AMMA architectures, AMMA_{18}, AMMA_{M}, and AMMA_{S} obtain 10.60%, 9.87%, and 11.22% E_{T}, respectively. These results do not indicate clear correlation between a stronger feature backbone (i.e., ResNet18) and better temporal localization performance. On the other hand, AMMA_{18} acquires the highest frame-AP overall (as reported in Table 6.1) by reducing false-positive detection throughout all other error categories as well as false-negative rates.

Relation between input sequence length and accuracy. In our earlier study from Section 5.7, we discovered that even though AMMA's frame-mAP on JHMDB-21 continues to rise upon increasing sequence length (T), it appears to saturate and even slightly degrade beyond $T = 3$ on UCF-24. To uncover possible causes of this frame-mAP drop, we assess changes of error distribution for AMMA_{18} under the influence of varied sequence length ($T = \{3, 4, 5\}$). The results are shown in Figure 6.4 (green bars). From this experiment, E_{T} actually expresses little correlation with varied sequence length. Instead, error rates in E_L, E_C, and E_O arise consistently when T increases.

We refer to the study by Kalogeiton et al. [75] and come to understand that JHMDB-21 may be better at coping long-range temporal modeling than UCF-24 due to the different extent of motion embedded in the two datasets. Particularly, actors in UCF-24 undergo larger movement and location shift. Kalogeiton et al. quantify the extent of actors' movement by computing the

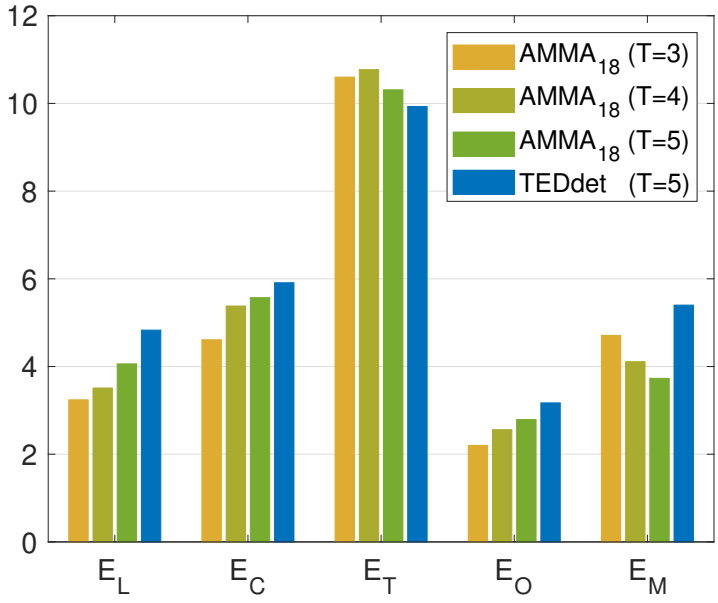

FIGURE 6.4: Error breakdown analysis on UCF-24 for AMMA_{18} ($T = \{3, 4, 5\}$) and TEDdet ($T = 5$).

mean action overlap over time. In detail, for each box in a groundtruth action tube, the overlap between this box and those n frames after is measured. The average action overlap among all groundtruth tubes and action classes is shown in Figure 6.5. One can observe that the overlap curve descends more quickly in UCF-24; the mean overlap is below 55% at $n = 16$ while that in JHMDB-21 still exceeds 66%. When an actor undergoes a significant location shift in the scene over time, action prediction (via 3×3 convolution) on top of their stacked features (i.e., the design of AMMA's Center branch) is prone to aggregate irrelevant background cues which inherently could lead to higher rate of E_C (and potentially E_O). In addition, large location shifts also raise the difficulty to track corresponding actors via Trajectory branch, which in turn negatively impacts E_L.

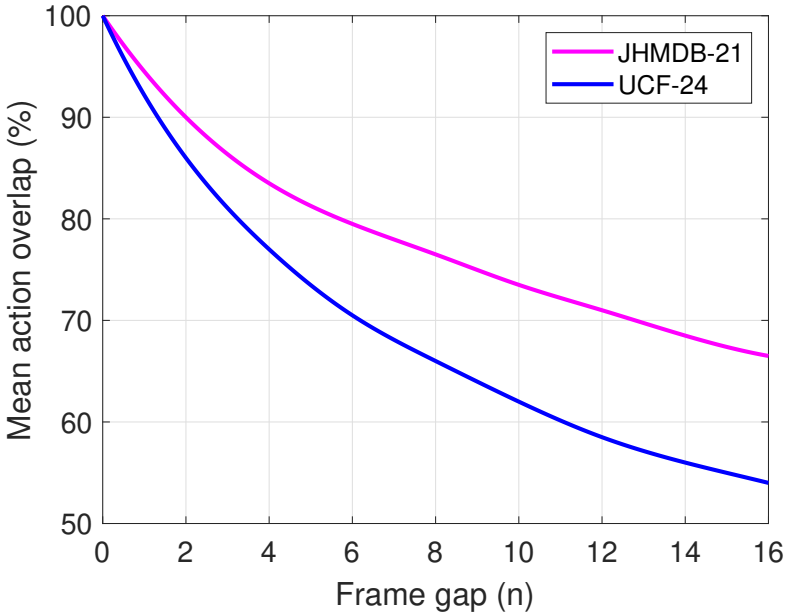

FIGURE 6.5: Mean action overlap between a box in a groundtruth tube and its box n frames later.

In Figure 6.4, we also compare AMMA$_{18}$ with TEDdet (blue bar) based on the same input sequence length ($T = 5$). We find that their different frame-mAP (73.5 vs. 70.8) are mainly tied with AMMA$_{18}$ managing to reduce localization and miss-detection error. For localization, even though AMMA and TEDdet both leverage stacking multi-frame features to model the trajectory of action instances, AMMA stacks features directly obtained from the backbone while TEDdet makes use of the displacement maps (i.e., via its Temporal Feature Difference module). The results in E_L show that modeling action movement on top of raw visual features is sufficient and arguably more general-purpose than the engineered displacement features adopted by TEDdet. Moreover, explicitly incorporating motion cues enables AMMA to better classify and recall action-specific pattern (improvement in E_C and E_M). On the downside, AMMA is more inclined to believe presence of actions outside the correct temporal extent (higher E_T).

It is note-worthy that in Figure 4.5, TEDdet did not seem to suffer from the aforementioned drawback of location shift ($T = 5$ being the best configuration on UCF-24). From the tracking perspective, TEDdet selects the middle frame of the input sequence as the reference one to model actors' movement, which is indeed less vulnerable than AMMA (selecting the final frame as reference) against location shift. From Center branch's point of view, the final feature used for classification by TEDdet is a "partial" collection of multiple frame features (results of TEDdet's TE module). Even though it contains less

information compared to directly stacking multi-frame features, we suspect that the partial-exchange mechanism has a repressive effect which prevents exceedingly aggregating irrelevant background cues.

Class-specific AP and error. Table 6.3 provides the class-wise frame-AP of $AMMA_{18}$ and TEDdet along with the distribution of errors (only on $AMMA_{18}$). For fair comparison, both presented models employ an input sequence length of 5. Additionally, the first two rows of the table report class-wise statistics of UCF-24 (testset). The average number of action instances per video is shown in the first row. The second row "Action span" reports the average duration of each action instance over its video (i.e., action-video ratio).

As shown in Table 6.3, $AMMA_{18}$ obtains higher APs than TEDdet in 17 out of 24 classes. From the per-class APs, time error, and action-video ratio, it is apparent that the most difficult classes are actions which span a shorter fraction of the video, highlighting the challenge of precise temporal localization in continuous action videos. These include "1-Basketball", "5-CricketBowling", "21-TennisSwing", and "23-VolleyballSpiking", each having a relatively large number of temporally untrimmed test videos. Note that the class "15-SalsaSpin" obtains high AP even though it appears to have a low action-video ratio. In fact, most test videos under this action category comprise a few actors lasting an extended duration, many of which are disconnected into separate instances due to disappearing from the scene for a few frames (e.g., blocked by other dancers). We also observe that the above action categories with high E_T tend to also express higher classification error (except for "23-VolleyballSpiking)". The high rate of E_T is likely an indicator that AMMA did not fully learn the most discriminative features embedded in these difficult classes, which in turn could lead to confusion among others, e.g., categorizing a running actor doing "PoleVault" as "CricketBowling", as both classes contain the running activity.

Further, actions such as "2-BasketBallDunk", "4-CliffDiving", and "12-LongJump" show higher rate of E_M, i.e., AMMA/TEDdet tend to miss detecting these actions more often. We qualitatively inspect their sequences and find that they contain a relatively large portion of non-discriminative frames with groundtruths. In other words, the class "2-BasketBallDunk" and "12-LongJump" expect correct action predictions during actors' run-up portion alone, which remain challenging under AMMA and TEDdet's detection framework.

Last but not least, localization error is more prominent for "2-BasketballDunk" and "22-TrampolineJump", whose video sequences often consist of more than

TABLE 6.3: Class-wise frame-APs and statistics of UCF-24. The first row (No. actions) refers to the average number of action instances per video. The second row (Action span) corresponds to the average action instance's duration relative to its full-video duration (action-video ratio). All 24 classes are presented in following order: Basketball (1-BB), BasketballDunk (2-BD), Biking (3-BK), CliffDiving (4-CD), CricketBowling (5-CB), Diving (6-DV), Fencing (7-FC), FloorGymnastics (8-FG), GolfSwing (9-GS), HorseRiding (10-HR), IceDancing (11-ID), LongJump (12-LJ), PoleVault (13-PV), RopeClimbing (14-RC), SalsaSpin (15-SS), SkateBoarding (16-SB), Skiing (17-SK), Skijet (18-SI), SoccerJuggling (19-SJ), Surfng (20-SF), TennisSwing (21-TS), TrampolineJumping (22-TJ), VolleyballSpiking (23-VS), and WalkingWithDog (24-WD).

Actions	1-BB	2-BD	3-BK	4-CD	5-CB	6-DV	7-FC	8-FG
No. actions	1.0	1.0	1.8	1.0	1.1	1.0	2.4	1.0
Action span	0.35	0.62	0.74	0.65	0.39	0.67	0.90	1.00
$AMMA_{18}$ (AP)	32.2	52.3	86.4	60.4	30.2	83.5	89.0	92.7
TEDdet (AP)	26.5	32.3	84.9	52.9	36.7	80.5	89.3	88.4
E_T	42.6	12.1	0.8	10.5	33.5	12.7	4.0	0.0
E_C	10.4	3.6	1.7	8.8	29.2	0.4	0.7	3.8
E_M	0.3	11.1	5.4	13.0	0.5	1.3	4.7	1.5
E_L	0.7	16.6	4.3	5.4	1.8	1.4	1.0	1.1

Actions	9-GS	10-HR	11-ID	12-LJ	13-PV	14-RC	15-SS	16-SB
No. actions	1	1.0	2.3	1.0	1.1	1.0	4.9	1.0
Action span	0.79	0.94	0.85	0.97	0.85	1.00	0.37	1.00
$AMMA_{18}$ (AP)	57.6	94.1	87.5	67.8	72.7	97.2	87.7	87.6
TEDdet (AP)	53.5	93.6	88.7	62.3	62.5	97.5	86.9	81.9
E_T	35.0	2.5	0.0	0.2	7.1	<0.1	<0.1	0.0
E_C	4.1	0.1	5.4	7.5	4.2	0.8	1.4	9.3
E_M	<0.1	1.6	0.9	15.0	6.5	0.8	2.6	0.4
E_L	0.7	1.5	5.4	6.9	5.6	0.6	7.4	1.0

Actions	17-SK	18-SI	19-SJ	20-SF	21-TS	22-TJ	23-VS	24-WD
No. actions	1.0	1.0	1.1	1.6	1.3	2.5	1.1	1.1
Action span	1.00	0.96	0.94	0.57	0.36	0.68	0.34	0.83
$AMMA_{18}$ (AP)	77.4	86.6	95.2	91.1	28.8	77.2	42.5	87.3
TEDdet (AP)	80.0	87.6	94.4	88.0	31.8	74.8	38.6	84.7
E_T	<0.1	0.1	0.2	2.5	32.2	3.1	41.6	6.9
E_C	11.2	0.9	2.3	0.7	21.5	1.6	1.4	2.7
E_M	2.5	5.3	0.2	2.6	1.5	4.7	6.4	0.8
E_L	5.6	5.6	0.8	2.6	1.4	11.9	6.9	1.2

one actors (regardless if they are labeled as groundtruth actions). For instance, "2-BasketBallDunk" videos are made up of highlights from multi-player games in full court, raising the difficulty to distinguish among players. On the contrary, "8-FloorGymnastics", "10-HorseRiding", "19-SoccerJuggling", and "14-RopeClimbing" are the easiest classes to detect as their sequences contain mostly one actor at a time with salient appearance features.

6.3 Computational complexity

Besides accuracy, a model's computational complexity is another important aspect to consider for real-world tasks and often approximated by the number of multiply-accumulate operations (MACs). The size of a model (measured by the number of trainable parameters) also indirectly expresses its degree of compactness.

In Table 6.4, we present the GMACs needed for AMMA and TEDdet to produce coarse tubelets over T frames. The total GMACs is reported along with the breakdown cost of individual sub-modules, comprising feature extraction from RGB frames, micro-motion (MM) generation & temporal feature extraction (for AMMA), and cooperative detection from three detector branches. The relative cost (%) of each sub-module is shown in parenthesis. Note that in practice, both AMMA and TEDdet exploit the feature-caching and dequeueing mechanism which reuses the previously acquired $T-1$ features for tubelet inference. As a result, their streaming GMACs can be approximated by dividing the total computation by T. Under this streaming-video setup, total GMACs reflect the cost to initialize AMMA/TEDdet upon encountering new videos.

On can clearly observe that most computation to produce coarse tubelets comes from extracting appearance cues from RGB frames ($\geq$ 65% of the total computation for all the methods). In AMMA, motion cue extraction is conducted over duplicated early layers of the same CNN backbone used for RGB stream; thus, the cost of micro-motion feature extraction is much smaller and correlates with the designated 2D backbone. Overall, effective motion-appearance feature fusion significantly improves AMMA_{18}'s ability to reason action-specific pattern (see Table 6.1) than that of TEDdet at the cost of raising computation by approximately 1.5 times. Meanwhile, ultra-lightweight backbone architectures combined with such spatiotemporal modeling scheme empower AMMA_{M} and AMMA_{S} to obtain highly competitive accuracy with only 1/4 and 1/5 the computational complexity of AMMA_{18}, respectively.

TABLE 6.4: Measure of AMMA and TEDdet's computational complexity (GMACs) and model size (number of trainable parameters). "MM" stands for micro-motion in the table.

Modules (T=5)	$AMMA_{18}$	$AMMA_M$	$AMMA_S$	TEDdet
RGB feat. extraction	16.9 (65%)	4.8 (71%)	3.4 (67%)	16.9 (97%)
MM generation	0.3 (1%)	0.3 (4%)	0.3 (5%)	0.0 (0%)
MM feat. extraction	8.3 (32%)	1.1 (16%)	0.8 (16%)	0.0 (0%)
Detector branches	0.6 (2%)	0.6 (9%)	0.6 (12%)	0.5 (3%)
Total GMACs	26.1	6.7	5.0	17.4
Streaming GMACs	5.2	1.3	1.0	3.5
Param. (M)	∼16	∼7	∼6	∼15

Compared to feature extraction, micro-motion generation and cooperative detection over the three detector branches incur significantly lower GMACs. This is more spectacularly reflected in $AMMA_{18}$ and TEDdet leveraging the slightly heavier ResNet18 feature backbone, whereas these sub-modules only take up to 3% of the total computation. On top of these observations, we can anticipate more rooms to improve action modeling based on more sophisticated motion representations or cooperative detection scheme at the expense of additional cost from these lightweight sub-modules.

Finally, the size of each method (i.e., number of trainable parameters) is reported in the last row of Table 6.4. To put these values in perspective, the standard SSD which is widely used in the domain of spatiotemporal action detection [73] [75] [77][84], has around 27M trainable parameters (54M when adopting the two-stream CNN framework). Similarly, the two-stream SSD architecture incurs approximately 32 GMACs (with input image of size 300×300), which is nearly 6 and 32 times more computationally expensive than our heaviest and lightest architectures. Methods leveraging 3D CNN [4][90] such as I3D or S3D as the feature backbone, are estimated to exceed 45 and 32 GMACs, respectively [83].

6.4 Runtime

In this section, we formally compare the runtime performance of AMMA and TEDdet. Following the hardware configuration presented in Chapter 4 and 5, we carry out runtime estimation on a desktop computer with a single NVIDIA Titan V5 GPU. Feature-caching and dequeuing are applied, and the testing mini-batch size is fixed to 1 to simulate real-world action detection on streaming videos. Runtime is first measured and analyzed based on

JHMDB-21 as all the top-performing AMMA and TEDdet models adopt the same input configuration on this dataset (i.e., input length $T = 5$). The complete pipeline to generate action tube proposals can be broken down into the followings: data loading, tubelet inference, tubelet linking, and intra-frame detection interpolation (the last two steps are grouped together as tube generation). Their results are presented in Table 6.5. Each row in the table (middle section) comprises the per-frame time cost for a given task (millisecond/f, or ms/f), as well as the time consumption relative to the entire detection pipeline (%).

From Table 6.5, we perceive that CNN model inference to acquire coarse tubelets takes the least amount of time out of the total runtime (<20%). In the case for AMMA, data loading ends up being the most time-consuming task, loading 4 frames each time to acquire the RGB frame and generate complementary micro-motion. It is worth the reminder that due to JHMDB-21 composing of short video clips, the streaming mechanism more frequently clears the feature buffer and loads 20 new frames upon processing any new video clip, thus taking relatively longer in data loading than TEDdet (which only requires sparse RGB frames).

Overall, all three AMMA architectures obtain fairly close speed performance (AMMA_{18} being slighter faster than the others). Their model inference time (tubelet detection) exhibit noticeable differences, where AMMA_{M} and AMMA_{S} actually being slower despite incurring less computational cost (as reported in Table 6.4). We have reported this finding and provided our insights in Section 5.7.

Compared to TEDdet, AMMA_{18} obtains similar tubelet inference time even though it is equipped with additional motion-related sub-networks (as opposed to TEDdet fully operating on a single stream). This is attributable to two factors. First, although incurring minimal extra cost in theory, the Temporal Feature Exchange module in TEDdet involves a series of reshape and transpose operations to carry out feature exchange across the time axis (plus additional operations to reverse these actions). As TEDdet includes multiple modules as such, the overall speed is slightly compromised. Second, feature-caching is executed in a slightly different manner between the two detectors. During inference, TEDdet retrieves $T - 1$ cached features extracted at the output of ResNet18, followed by temporal modeling (feature exchange) in the subsequent deconvolution block and then the detector branches. On the other hand, AMMA retrieves cached features directly from the output of the

TABLE 6.5: Measure of AMMA and TEDdet's runtime (millisecond and FPS).

Modules (T=5)	$AMMA_{18}$	$AMMA_{M}$	$AMMA_{S}$	TEDdet
Data loading (ms)	7.6 (60%)	8.0 (61%)	7.7 (57%)	3.8 (42%)
Tubelet detection (ms)	1.4 (11%)	1.9 (15%)	2.2 (17%)	1.6 (18%)
Tube generation (ms)	3.7 (29%)	3.1 (24%)	3.5 (26%)	3.7 (40%)
Total runtime (ms)	12.7	13.0	13.4	9.1
Overall speed (FPS)	80	77	75	110

deconvolution block, leaving only cooperative detection for tubelet prediction.

Influence of sequence length on runtime. One may have noticed the conspicuous speed difference of $AMMA_{18}$ between JHMDB-21 and UCF-24 (80 FPS vs. 115 FPS) in Table 5.3. Such discrepancy is mainly associated with the choice of input sequence length. Here, we demonstrate the influence of sequence length ($T = \{2, 3, 4, 5\}$) on $AMMA_{18}$'s runtime based on UCF-24, as presented in Figure 6.6. The runtime of each detection phase is separately presented while the final FPS is plotted in red. As T increases, tubelet inference time remains unaffected as our detector exploits feature-caching and retrieval. We also observe a consistent yet minor increase in runtime associated with data loading, which can be explained by the initialization state of AMMA. On the other hand, the runtime associated with tubelet linking prominently arises along sequence length. As tubelet linking depends on calculating the mean IoUs of detection across $T - 1$ overlapping frames, determining whether two tubelets match inherently becomes more computationally demanding when longer tubelets are considered. These results and those on actions' location shift in Figure 6.5 pin-point the importance of a carefully chosen sequence length for balancing AMMA's accuracy and speed performance.

6.5 Summary, limitations, and looking ahead

This chapter extends from Chapter 5 and elaborates various performances of AMMA at a deeper level. It also formally compares many aspects of the detector with our previously proposed TEDdet from Chapter 4.

Specifically, thorough error breakdown analysis (five mutually-exclusive error categories) is carried out to evaluate possible sources of frame-mAP loss. Compared to TEDdet on JHMDB-21, we find that with the addition of motion cue reasoning, AMMA consistently obtains better frame-mAP by greatly reducing false-positive classification. On the other hand, qualitative

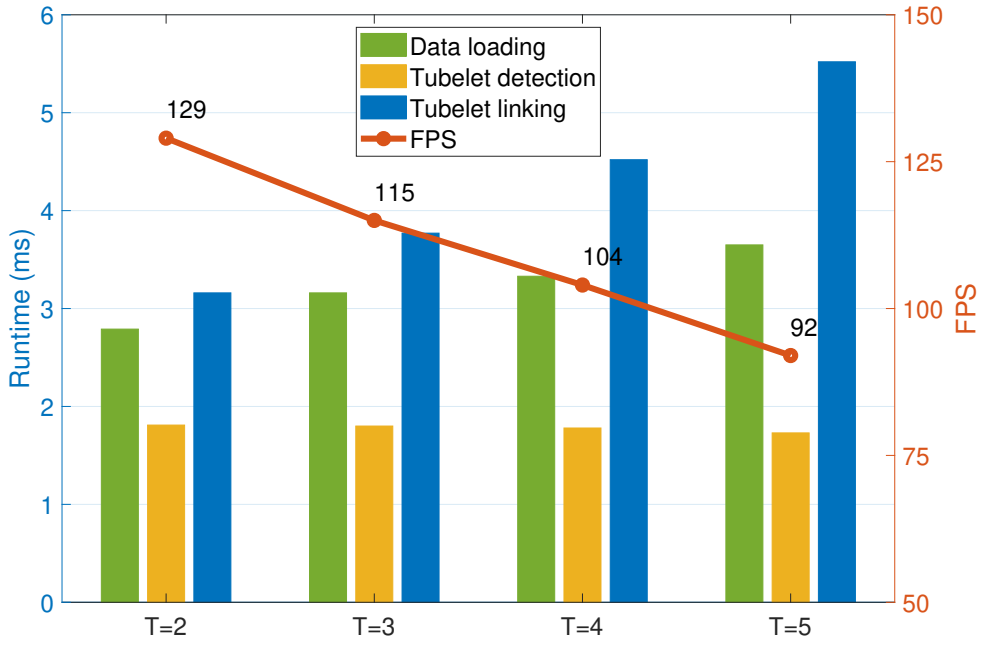

FIGURE 6.6: AMMA$_{18}$'s runtime breakdown over varied sequence length (T) evaluated on UCF-24. The bar graph (referencing Y-axis on the left) captures the decomposition of detection runtime. The red plot (referencing Y-axis on the right) describes the overall FPS of each configured T.

inspection on action sequences of lower AP shows that lacking long-temporal range modeling potentially prevents our models from learning discriminative, defining pattern specific to these actions. In the case of the temporally-untrimmed dataset UCF-24, our methods tend to incur higher time error on classes where action instances take place relatively short in their videos, indicating the need for an explicit temporal localization module. Detailed breakdown analysis and comparison on AMMA (and TEDde)'s model complexity and runtime have also been covered, providing a holistic view on our design choices and potential bottlenecks.

In conclusion, the ultra-fast and lightweight nature of our detectors opens up manifold potential applications in budget online action detection. Even though AMMA demonstrates competitive accuracy at significantly reduced computation and in real-time, there remain rooms for improvement. Mainly, long-temporal range modeling and temporal boundary modeling are anticipated to help reduce false-positive detection in the form of classification and time error, respectively. In the next chapter, we will refresh on our findings throughout this thesis as well as discussing pending works in the future.

Chapter 7

Conclusion and Future perspectives

In this thesis, we have presented our three-year research work addressing the problem of spatiotemporal action detection in videos. Unlike many concurrent research, we imposed strict constraints mandating an online, real-time and lightweight detection paradigm for feasible edge-device deployment. These constraints closely reflect the design specifications in realistic application scenarios such as unmanned surveillance and human-computer interaction systems. Our main contributions are proposals of three highly efficient action detectors leveraging compact 2D CNN architectures and raw video frames (RGB modality) as input, which have been detailed in Chapter 3, 4, 5, and 6.

In this final chapter, we first summarize our contributions in a chapter-wise order. We then point out some limitations of our research outcomes, and propose pertinent research directions in terms of both short-term and long-term future works.

7.1 Summary of contributions

7.1.1 Detection acceleration and spatiotemporal modeling via flow-guided features

In Chapter 3, we first attempted to enhance the inference speed of the single-frame action detector on videos by exploiting temporal coherence between consecutive frames. Our proposed ACDnet sparsely extracts visual features from a video's key frames while densely performing feature approximation at other timestamps (by spatially guiding appearance features from their precedent key frames using reduced flow fields). Such a detection strategy largely mitigates redundant feature extraction on neighboring frames

sharing visually similar context. Further, ACDnet recursively accumulates key frame features from the past (leveraging the same warping schema to align past and current feature maps), allowing it to adaptively aggregate past-current observations and model temporal variations of actions in an implicit manner. The proposed feature-approximation and memory aggregation modules are generic; they can be inserted in off-the-shelf object detectors with minor adjustments (we experimented with SSD and R-FCN).

Feature approximation helps to accelerate video action detection in exchange for a minor drop in accuracy, while pairing it with memory aggregation enables ACDnet to supersede the baseline detector in both accuracy and speed (75 FPS when integrated with SSD). In particular, accumulating past observations allows ACDnet to understand degenerated frames with limited visual cues.

7.1.2 Spatiotemporal Modeling via feature-channel exchange and feature-map displacement

In lieu of the previous single-frame detection strategy, we further explored extracting discriminating video representations in Chapter 3 by concurrently processing a series of video frames and predicting the underlying action tubelets. The proposed action tubelet detector (coined TEDdet) makes use of its Temporal Feature Exchange (TE) and Temporal Feature Difference (TD) modules for spatiotemporal modeling. Given the CNN features of successive video frames, TE performs partial and weighted channel-wise exchange among neighboring features to gather action-specific context. Meanwhile, TD generates motion maps based on the feature-level displacement to track actors' locations over time. We jointly optimize the complementary TE and TD modules with three detector branches derived from CenterNet, which is anchor-free thus is not bounded by heuristic anchor configurations.

TEDdet adopts a coarse-detection schema (i.e., its input frames are separated by the designated temporal stride) to efficiently cover prominent appearance variations over time. Employing an incremental detection pipeline and tube construction algorithm, it generates action tubes in real-time (110FPS) at competitive accuracy when equipped with a 2D ResNet18 backbone. Our ablation studies indicate that TE-induced feature-channel exchange is capable of modeling action-specific pattern that is time-sensitive (such as distinguishing between Sit and Stand). Meanwhile, TD-induced tracking addresses actors' location shift throughout the video sequence, leading to more

precise localization of individual action instances. It was also found that enlarging the temporal stride of an input sequence during training permits learning higher variations of visual pattern and improves the detection robustness at test time. Such an attribute not only reduces training intensity on long video sequences, but also lifts off reliance on densely annotated bounding boxes.

7.1.3 Leveraging accumulated motion boundaries

In Chapter 5, we first retraced how optical flow benefited action recognition from recent studies. Based on the insight that motion boundaries of moving actors/objects play a critical role to enhance recognition accuracy, we devised a compact convolutional flow-like estimator to explicitly encode actions' short-temporal dynamics on-the-fly from raw video frames (we referred to this motion representation as micro-motion). Multiple micro-motions from different timestamps are accumulated to capture continuous dynamics. Generation of such motion cues can be further integrated into the backbone of our previous tubelet detector (as described in Chapter 4). In practice, each pair of RGB and complementary micro-motion features are extracted from a short clip via two-CNN pathways (and combined by lateral-fusion). The entire tubelet detector (termed AMMA) models a long action sequence on top of a series of connected video clips.

We conducted multiple ablation studies by varying the input sequence length, motion compositions, extent of lateral-fusion, and a variety of ultra-compact 2D CNN backbones (e.g., ResNet18, MobileNetV2, and ShuffleNetV2). AMMA demonstrates highly competitive frame-mAP and video-mAP while substantially reducing the computation (lightest variant incurs 1.0 GMACs) and runtime cost (maximally at 80 FPS). In Chapter 6, we further carried out comprehensive breakdown analysis to uncover different categories of errors for AMMA. By comparing three AMMA variants with TEDdet, we showed that incorporating micro-motion cues with appearance information consistently produces more accurately classified tubelets than inferring only from RGB features.

7.2 Limitations in our models

The latest action detector (AMMA) from Chapter 5 has successfully addressed a number of limitations of our earlier works. To recapitulate, AMMA takes

a series of non-overlapped video clips as input, surmounting ambiguous action estimation based upon the glance over only one frame. Building upon an anchor-free detection framework, AMMA does not suffer from the heuristic anchor configuration, complicated training-sample selection nor 3D-NMS to filter redundant detection (the above improve upon Chapter 3). Moreover, the integration of a convolutional motion estimator and the two-pathway CNN architecture facilitate simultaneous mining of action-specific appearances, subtle dynamics, and their interactions. Last but not least, AMMA's detection paradigm is generic and has been validated on three ultra-lightweight CNN backbones, demonstrating a better mix of efficiency and accuracy than state-of-the-art models (the above improve upon Chapter 4). We refer our readers to the introduction and conclusion sections of Chapter 3-6 regarding our research advancement in detail.

Despite having gradually overcome a myriad of design challenges under a tight hardware constraint, we argue that AMMA still exhibits perceivable limitations when encountering the following scenarios.

7.2.1 Drastic inter-class variations

Firstly, it was observed that AMMA's classification performance was compromised on groups of actions encompassing drastic inter-class variations. For instance, the action Run or Throw can each be perceived very distinctively when captured by different camera shot angles (examples are illustrated at the bottom of Figure 6.2). When video samples are insufficient and fail to convey a wide range of possible inter-class visual variations for particular actions, it is reasonable to suspect that our model is prone to overfitting on scenic cues that do not generalize well to those actions' defining traits. The inclusion of appearance-invariant motion representation moderately mitigates this issue but only to an extent.

7.2.2 Pruning low-confidence predictions

Second, we observed that AMMA's video-mAP is less impressive at lower detection thresholds (refer to the JHMDB-21's comparison in Table 5.4) due to retaining a higher number of false-positive action tubes. On the one hand, this is attributed to AMMA coarsely detecting tubelets from an extended video sequence and then interpolating the inter-frame detection (presuming the consistency of any action instance within this duration). It thus becomes more difficult to simply prune false-positive detection by a bounding-box

association strategy during the linking phase. On the other hand, the manifestation of low-confidence detection may suggest that the spatiotemporal features learned and extracted by our models are not discriminative enough to produce quality tubelets in the first place.

7.2.3 Temporal localization

Third, it was apparent that the most challenging classes in the untrimmed UCF-24 dataset for AMMA are those spanning a shorter fraction of their videos (see Table 6.3). When the appearance and motion cues of a continuous action gradually change over time, the frames near the boundaries (between action and no-action) may only exhibit subtle differences. In such a case, it is extremely difficult to precisely identify actions' temporal boundaries, leaving a large room for improvement of temporal localization.

7.2.4 Long-temporal relations

Notably, AMMA lacks the capacity to extract very long-range supportive context as it operates on connected video clips. Consequently, it is likely to miss capturing relevant visual/motion cues pertinent to specific actions (some examples are depicted at the top of Figure 6.2). In addition, this limitation also makes it hard to identify actions that are somewhat similar throughout most parts of the video, e.g., both the action LongJump and PoleVault involve actors running at the beginning and are only different at the last part of their videos. Naively taking in more clips is sub-optimal as such a strategy is prone to introducing more redundancy and irrelevant context.

7.3 Future works

In this section, we propose a number of future works based on the aforementioned experimental results, observed limitations, and the latest advancements in action understanding. Our future works can be loosely divided into short-term tasks and long-term research directions as detailed below.

7.3.1 Short-term future tasks

Actions can be semantically similar but visually very different. Even when initialized with ImageNet pretraining, detection models are inclined to overfit on non-generalized scenic cues (especially when the target dataset is small).

A better model initialization that can guide CNN to learn generalized video representations is inherently critical. Meanwhile, recent action recognition methods have demonstrated success when first pretraining the CNN backbones on top of the large-scale Kinetics dataset [38] and then fine-tuning their models on smaller action datasets. The same practice can be transferred to our action detection problem and possibly mitigate limitation 7.2.1. Even though the volume of Kinetics may be training-intensive, the research community in action understanding is active and has made various off-the-shelf pretrained models available [114][115].

It is worth mentioning that off-the-shelf Kinetics-pretrained models are typically based on 3D CNN, which can not be directly applied in our proposed 2D-based architecture. To leverage 3D-pretrained weights, one possible way is to revert the inflated 3D architecture [38] back to 2D. We hypothesize that by properly collapsing the weights of a 3D convolutional kernels along the temporal axis (e.g., averaging or weighted-sum), we can obtain better generalized visual representations to initialize our 2D CNN backbone. At the same time, it would be interesting to investigate decomposed variants of 3D architectures such as P3D [45], R(2+1)D [46] or S3D [47] which decouple 3D convolution into 2D-spatial and 1D-temporal convolutions for efficiency gain.

After the write-up of this manuscript, we will also evaluate the proposed methods against more public benchmarks, especially on AVA [4]. This is a tremendously challenging spatiotemporal action dataset comprising realistically complex scenes of multiple co-occurring action instances and subtly different action categories (such as touching vs. holding an object). Moreover, actors can be associated with multiple labels corresponding to the their poses, interactions with surrounding objects, and with other persons. We believe the performance on AVA will grant us a more transparent understanding of our models' strengths and weakness when coping with realistic scenarios.

Furthermore, we aim to look into the topic of network compression and deploy our latest models (i.e., variants of AMMA integrated with ResNet18, MobileNetV2 and ShuffleNetV2) on embedded vision systems, such as those powered by NVIDIA Jetson or Xavier series. Not only will we formally assess the feasibility of the proposed detectors localizing actions from streaming videos, but also examine computational bottlenecks, time and power consumption closer to the real-world sensing configuration.

7.3.2 Long-term future research directions

As opposed to short-term tasks which may be oriented toward algorithmic implementation and optimization, we propose three long-term research directions with the aim of systematically enhancing our current action detection framework.

The first research direction is to improve temporal localization for untrimmed action videos (this limitation was described in Section 7.2.3). A potential solution is to additionally devise class-specific actionness classifiers (i.e., N binary classifiers where N is the number of action classes) to post-process linked action tubes, as inspired by Su et al. [111]. In such a setup, each class-specific training sample of a linked action tube may include bounding boxes from frames that do not contain the action. These frames can thus serve as negative samples, allowing its class-specific actionness classifier to learn and identify the subtle yet defining attributes of that particular action. At test time, the actionness classifier can be simply applied after action tubes are constructed, refining temporal localization by filtering out bounding boxes that take place in low-actionness frames. Similarly, Song et al. [112]. proposed a transition-aware classifier to jointly predict the action category and transitional state (as shifting between no-action and action is gradual and challenging to distinguish). We believe that these two works, along with literature in temporal action detection, will offer us solid insights to proceed improving temporal localization.

The second research direction is to explore finer-grained motion representations for precisely encoding dynamics of actions. This is driven by the limitation (observed in Section 7.2.2) that our action tube detector is manifested with low-confidence, false-positive detection, an indication of insufficient spatiotemporal modeling to distinguish visually similar actions. Here, we identify a number of research works to explore, all of which focusing on producing trainable motion representations in the latent space [30][28][52][116][117]. Specifically, the above methods encode actions' motion information by operating on successive CNN feature maps of the video sequence. As feature maps have smaller spatial resolutions, these methods tend to be much more efficient and computationally lightweight. Moreover, we believe that motion can be further employed to adaptively sample salient video frames pertinent to actions while suppressing the irrelevant background distraction [118]. At last, it would be interesting to also seek different motion-appearance fusion schemes, such as bilateral fusion [119], more sophisticated message-passing

modules [111], or a heterogeneous two-stream CNN architecture when extracting spatial-temporal context.

Finally, we aim to incorporate long-range temporal modeling in our methods for understanding complex actions/activities in streaming videos. Recent works in [92] and [120] have demonstrated caching distant observations from the past in a feature bank, which can then be retrieved for augmenting current observations and detection. Particularly, it would be encouraging to progressively delve into the topic of the attention mechanism [44] and its derivations in the image [121][122] and video space [89][88]. The attention mechanism not only facilitates long-temporal range modeling but also redundant-feature filtering, both being highly crucial for effectively modeling video sequences. In search of an architecture that is both efficient and accurate, a factorized transformer [43] that retains all spatiotemporal feature structure while maintaining high efficiency is particularly promising to explore. While some of the aforementioned methods may have overlooked efficiency and real-time performance (not to mention their compatibility with computation-constrained devices), we will continue to seek for ultra-compact alternatives that accomplish an optimal trade-off among accuracy, speed, and efficiency.

Bibliography

[1] Thomas Marcoux et al. "Analyzing cyber influence campaigns on YouTube using YouTubeTracker". In: *Big Data and Social Media Analytics*. Springer, 2021, pp. 101–111.

[2] Yann LeCun et al. "Object recognition with gradient-based learning". In: *Shape, contour and grouping in computer vision*. Springer, 1999, pp. 319–345.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012).

[4] Chunhui Gu et al. "Ava: A video dataset of spatio-temporally localized atomic visual actions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6047–6056.

[5] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. "UCF101: A dataset of 101 human actions classes from videos in the wild". In: *arXiv preprint arXiv:1212.0402* (2012).

[6] Hueihan Jhuang et al. "Towards understanding action recognition". In: *IEEE ICCV*. 2013, pp. 3192–3199.

[7] Matthew S Hutchinson and Vijay N Gadepally. "Video action understanding: A tutorial". In: *IEEE Access* (2021).

[8] Adnan ÖZSOY. "A Comprehensive Performance Comparison of Dedicated and Embedded GPU Systems". In: *Dicle Üniversitesi Mühendislik Fakültesi Mühendislik Dergisi* 11.3 (2020), pp. 1011–1020.

[9] Ivan Laptev. "On space-time interest points". In: *International journal of computer vision* 64.2 (2005), pp. 107–123.

[10] Marti A. Hearst et al. "Support vector machines". In: *IEEE Intelligent Systems and their applications* 13.4 (1998), pp. 18–28.

[11] Alexander Klaser, Marcin Marszałek, and Cordelia Schmid. "A spatio-temporal descriptor based on 3d-gradients". In: *BMVC 2008-19th British Machine Vision Conference*. British Machine Vision Association. 2008, pp. 275–1.

[12] Paul Scovanner, Saad Ali, and Mubarak Shah. "A 3-dimensional sift descriptor and its application to action recognition". In: *Proceedings of the 15th ACM international conference on Multimedia*. 2007, pp. 357–360.

[13] Geert Willems, Tinne Tuytelaars, and Luc Van Gool. "An efficient dense and scale-invariant spatio-temporal interest point detector". In: *European conference on computer vision*. Springer. 2008, pp. 650–663.

[14] Ivan Laptev et al. "Learning realistic human actions from movies". In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2008, pp. 1–8.

[15] Heng Wang and Cordelia Schmid. "Action recognition with improved trajectories". In: *Proceedings of the IEEE international conference on computer vision*. 2013, pp. 3551–3558.

[16] Gabriella Csurka et al. "Visual categorization with bags of keypoints". In: *Workshop on statistical learning in computer vision, ECCV*. Vol. 1. 1-22. Prague. 2004, pp. 1–2.

[17] Florent Perronnin and Christopher Dance. "Fisher kernels on visual vocabularies for image categorization". In: *2007 IEEE conference on computer vision and pattern recognition*. IEEE. 2007, pp. 1–8.

[18] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[19] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[20] Andrej Karpathy et al. "Large-scale video classification with convolutional neural networks". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2014, pp. 1725–1732.

[21] Karen Simonyan and Andrew Zisserman. "Two-stream convolutional networks for action recognition in videos". In: *Advances in neural information processing systems* 27 (2014).

[22] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. "Convolutional two-stream network fusion for video action recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 1933–1941.

[23] Eunbyung Park et al. "Combining multiple sources of knowledge in deep cnns for action recognition". In: *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2016, pp. 1–8.

[24] Limin Wang et al. "Temporal segment networks: Towards good practices for deep action recognition". In: *European conference on computer vision*. Springer. 2016, pp. 20–36.

[25] Jeffrey Donahue et al. "Long-term recurrent convolutional networks for visual recognition and description". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 2625–2634.

[26] Joe Yue-Hei Ng et al. "Beyond short snippets: Deep networks for video classification". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 4694–4702.

[27] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[28] Shuyang Sun et al. "Optical flow guided feature: A fast and robust motion representation for video action recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1390–1399.

[29] Bowen Zhang et al. "Real-time action recognition with deeply transferred motion vector cnns". In: *IEEE Transactions on Image Processing* 27.5 (2018), pp. 2326–2339.

[30] Hakan Bilen et al. "Dynamic image networks for action recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3034–3042.

[31] Can Zhang et al. "Pan: Towards fast action recognition via learning persistence of appearance". In: *arXiv preprint arXiv:2008.03462* (2020).

[32] Alexey Dosovitskiy et al. "Flownet: Learning optical flow with convolutional networks". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2758–2766.

[33] Eddy Ilg et al. "Flownet 2.0: Evolution of optical flow estimation with deep networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2462–2470.

[34] Joe Yue-Hei Ng et al. "Actionflownet: Learning motion representation for action recognition". In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2018, pp. 1616–1624.

[35] Yi Zhu et al. "Hidden two-stream convolutional networks for action recognition". In: *Asian conference on computer vision*. Springer. 2018, pp. 363–378.

[36] Shuiwang Ji et al. "3D convolutional neural networks for human action recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 35.1 (2012), pp. 221–231.

[37] Du Tran et al. "Learning spatiotemporal features with 3d convolutional networks". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4489–4497.

[38] Joao Carreira and Andrew Zisserman. "Quo vadis, action recognition? a new model and the kinetics dataset". In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6299–6308.

[39] Christoph Feichtenhofer et al. "Slowfast networks for video recognition". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 6202–6211.

[40] Xiaolong Wang et al. "Non-local neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7794–7803.

[41] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. "Is space-time attention all you need for video understanding". In: *arXiv preprint arXiv:2102.05095* 2.3 (2021), p. 4.

[42] Daniel Neimark et al. "Video transformer network". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 3163–3172.

[43] Raivo Koot and Haiping Lu. "VideoLightFormer: Lightweight Action Recognition using Transformers". In: *arXiv preprint arXiv:2107.00451* (2021).

[44] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[45] Zhaofan Qiu, Ting Yao, and Tao Mei. "Learning spatio-temporal representation with pseudo-3d residual networks". In: *proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 5533–5541.

[46] Du Tran et al. "A closer look at spatiotemporal convolutions for action recognition". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2018, pp. 6450–6459.

[47] Saining Xie et al. "Rethinking spatiotemporal feature learning for video understanding". In: *arXiv preprint arXiv:1712.04851* 1.2 (2017), p. 5.

[48] Mohammadreza Zolfaghari, Kamaljeet Singh, and Thomas Brox. "Eco: Efficient convolutional network for online video understanding". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 695–712.

[49] Chenxu Luo and Alan L Yuille. "Grouped spatial-temporal aggregation for efficient action recognition". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 5512–5521.

[50] Christoph Feichtenhofer. "X3d: Expanding architectures for efficient video recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 203–213.

[51] Ji Lin, Chuang Gan, and Song Han. "Tsm: Temporal shift module for efficient video understanding". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7083–7093.

[52] Myunggi Lee et al. "Motion feature network: Fixed motion filter for action recognition". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 387–403.

[53] Boyuan Jiang et al. "Stm: Spatiotemporal and motion encoding for action recognition". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 2000–2009.

[54] Yan Li et al. "Tea: Temporal excitation and aggregation for action recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 909–918.

[55] Limin Wang et al. "Tdn: Temporal difference networks for efficient action recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 1895–1904.

[56] Jasper RR Uijlings et al. "Selective search for object recognition". In: *International journal of computer vision* 104.2 (2013), pp. 154–171.

[57] C Lawrence Zitnick and Piotr Dollár. "Edge boxes: Locating object proposals from edges". In: *European conference on computer vision*. Springer. 2014, pp. 391–405.

[58] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems* 28 (2015).

[59] Jifeng Dai et al. "R-fcn: Object detection via region-based fully convolutional networks". In: *Advances in neural information processing systems* 29 (2016).

[60] Tsung-Yi Lin et al. "Feature pyramid networks for object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

[61] Joseph Redmon and Ali Farhadi. "YOLO9000: better, faster, stronger". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7263–7271.

[62] Wei Liu et al. "Ssd: Single shot multibox detector". In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[63] Tsung-Yi Lin et al. "Focal loss for dense object detection". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[64] Hei Law and Jia Deng. "Cornernet: Detecting objects as paired keypoints". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 734–750.

[65] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. "Objects as points". In: *arXiv preprint arXiv:1904.07850* (2019).

[66] Zhi Tian et al. "Fcos: Fully convolutional one-stage object detection". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 9627–9636.

[67] Georgia Gkioxari and Jitendra Malik. "Finding action tubes". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 759–768.

[68] G David Forney. "The viterbi algorithm". In: *Proceedings of the IEEE* 61.3 (1973), pp. 268–278.

[69] Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. "Learning to track for spatio-temporal action localization". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 3164–3172.

[70] Suman Saha et al. "Deep learning for detecting multiple space-time action tubes in videos". In: *arXiv preprint arXiv:1608.01529* (2016).

[71] Xiaojiang Peng and Cordelia Schmid. "Multi-region two-stream R-CNN for action detection". In: *European conference on computer vision*. Springer. 2016, pp. 744–759.

[72] Zhenheng Yang, Jiyang Gao, and Ram Nevatia. "Spatio-temporal action detection with cascade proposal and location anticipation". In: *arXiv preprint arXiv:1708.00042* (2017).

[73] Gurkirt Singh et al. "Online real-time multiple spatiotemporal action localisation and prediction". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 3637–3646.

[74] Harkirat Singh Behl et al. "Incremental tube construction for human action detection". In: *arXiv preprint arXiv:1704.01358* (2017).

[75] Vicky Kalogeiton et al. "Action tubelet detector for spatio-temporal action localization". In: *IEEE ICCV*. 2017, pp. 4405–4413.

[76] Suman Saha, Gurkirt Singh, and Fabio Cuzzolin. "Amtnet: Action-micro-tube regression by end-to-end trainable deep architecture". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 4414–4423.

[77] Suman Saha, Gurkirt Singh, and Fabio Cuzzolin. "Two-stream amtnet for action detection". In: *arXiv preprint arXiv:2004.01494* (2020).

[78] Pavel Tokmakov, Karteek Alahari, and Cordelia Schmid. "Learning video object segmentation with visual memory". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 4481–4490.

[79] Rui Hou, Chen Chen, and Mubarak Shah. "Tube convolutional neural network (T-CNN) for action detection in videos". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5822–5831.

[80] Dong Li et al. "Recurrent tubelet proposal and recognition networks for action detection". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 303–318.

[81] Wei Li et al. "Deformable tube network for action detection in videos". In: *arXiv preprint arXiv:1907.01847* (2019).

[82] Jiawei He et al. "Generic tubelet proposals for action localization". In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2018, pp. 343–351.

[83] Yixuan Li et al. "Actions as moving points". In: *European Conference on Computer Vision*. Springer. 2020, pp. 68–84.

[84] Jiaojiao Zhao and Cees GM Snoek. "Dance with flow: Two-in-one stream action detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9935–9944.

[85] Okan Köpüklü, Xiangyu Wei, and Gerhard Rigoll. "You only watch once: A unified cnn architecture for real-time spatiotemporal action localization". In: *arXiv preprint arXiv:1911.06644* (2019).

[86] Yuxi Li et al. "Finding action tubes with a sparse-to-dense framework". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 11466–11473.

[87] Yuxi Li et al. "Cfad: Coarse-to-fine action detector for spatiotemporal action localization". In: *European Conference on Computer Vision*. Springer. 2020, pp. 510–527.

[88] Rohit Girdhar et al. "Video action transformer network". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 244–253.

[89] Jiaojiao Zhao et al. "Tuber: Tube-transformer for action detection". In: *arXiv preprint arXiv:2104.00969* (2021).

[90] Chen Sun et al. "Actor-centric relation network". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 318–334.

[91] Yubo Zhang et al. "A structured model for action detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9975–9984.

[92] Chao-Yuan Wu et al. "Long-term feature banks for detailed video understanding". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 284–293.

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

[94] Philippe Weinzaepfel, Xavier Martin, and Cordelia Schmid. "Human action localization with sparse spatial supervision". In: *arXiv preprint arXiv:1605.05197* (2016).

[95] Hildegard Kuehne et al. "HMDB: a large video database for human motion recognition". In: *2011 International conference on computer vision*. IEEE. 2011, pp. 2556–2563.

[96] Mark Everingham et al. "The PASCAL visual object classes challenge 2007 (VOC2007) results". In: (2008).

[97] Xitong Yang et al. "Step: Spatio-temporal progressive learning for video action detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 264–272.

[98] Yu Liu, Fan Yang, and Dominique Ginhac. "ACDnet: An Action Detection network for real-time edge computing based on flow-guided feature approximation and memory aggregation". In: *Pattern Recognition Letters* 145 (2021), pp. 118–126.

[99] Xizhou Zhu et al. "Deep feature flow for video recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2349–2358.

[100] Thomas Brox et al. "High accuracy optical flow estimation based on a theory for warping". In: *European conference on computer vision*. Springer. 2004, pp. 25–36.

[101] Congrui Hetang et al. "Impression network for video object detection". In: *arXiv preprint arXiv:1712.05896* (2017).

[102] Tianqi Chen et al. "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems". In: *arXiv preprint arXiv:1512.01274* (2015).

[103] Alaaeldin Ali and Graham W Taylor. "Real-time end-to-end action detection with two-stream networks". In: *2018 15th Conference on Computer and Robot Vision (CRV)*. IEEE. 2018, pp. 31–38.

[104] Till Kroeger et al. "Fast optical flow using dense inverse search". In: *European Conference on Computer Vision*. Springer. 2016, pp. 471–488.

[105] Adam Paszke et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019).

[106] Dejun Zhang et al. "Learning motion representation for real-time spatio-temporal action localization". In: *Pattern Recognition* 103 (2020), p. 107312.

[107] Laura Sevilla-Lara et al. "On the integration of optical flow and action recognition". In: *German conference on pattern recognition*. Springer. 2018, pp. 281–297.

[108] Mark Sandler et al. "Mobilenetv2: Inverted residuals and linear bottlenecks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520.

[109] Ningning Ma et al. "Shufflenet v2: Practical guidelines for efficient cnn architecture design". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 116–131.

[110] Marin Orsic et al. "In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 12607–12616.

[111] Rui Su et al. "Improving action localization by progressive cross-stream cooperation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 12016–12025.

[112] Lin Song et al. "Tacnet: Transition-aware context network for spatio-temporal action detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 11987–11995.

[113] Zhaofan Qiu et al. "Learning spatio-temporal representation with local and global diffusion". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 12056–12065.

[114] Okan Köpüklü et al. "Resource efficient 3d convolutional neural networks". In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE. 2019, pp. 1910–1919.

[115] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. "Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?" In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2018, pp. 6546–6555.

[116] AJ Piergiovanni and Michael S Ryoo. "Representation flow for action recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9945–9953.

[117] Heeseung Kwon et al. "Motionsqueeze: Neural motion feature learning for video understanding". In: *European Conference on Computer Vision*. Springer. 2020, pp. 345–362.

[118] Yuan Zhi et al. "Mgsampler: An explainable sampling strategy for video action recognition". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 1513–1522.

[119] Guoxi Huang and Adrian G Bors. "Video classification with finecoarse networks". In: *arXiv e-prints* (2021), arXiv–2103.

[120] Sara Beery et al. "Context r-cnn: Long term temporal context for per-camera object detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 13075–13085.

[121] Nicolas Carion et al. "End-to-end object detection with transformers". In: *European conference on computer vision*. Springer. 2020, pp. 213–229.

[122] Alexey Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).