Deep Learning for Action Understanding in Video

Zheng Shou

Submitted in partial fulfillment of the

requirements for the degree

of Doctor of Philosophy

in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2019

ABSTRACT

Deep Learning for Action Understanding in Video

Zheng Shou

Action understanding is key to automatically analyzing video content and thus is important for many real-world applications such as autonomous driving car, robot-assisted care, *etc*. Therefore, in the computer vision field, action understanding has been one of the fundamental research topics. Most conventional methods for action understanding are based on hand-crafted features. Like the recent advances seen in image classification, object detection, image captioning, *etc*, deep learning has become a popular approach for action understanding in video. However, there remain several important research challenges in developing deep learning based methods for understanding actions. This thesis focuses on the development of effective deep learning methods for solving three major challenges.

Action detection at fine granularities in time: Previous work in deep learning based action understanding mainly focuses on exploring various backbone networks that are designed for the video-level action classification task. These did not explore the fine-grained temporal characteristics and thus failed to produce temporally precise estimation of action boundaries. In order to understand actions more comprehensively, it is important to detect actions at finer granularities in time. In Part I, we study both segment-level action detection and frame-level action detec-

tion. Segment-level action detection is usually formulated as the temporal action localization task, which requires not only recognizing action categories for the whole video but also localizing the start time and end time of each action instance. To this end, we propose an effective multi-stage framework called Segment-CNN consisting of three segment-based 3D ConvNets: (1) a proposal network identifies candidate segments that may contain actions; (2) a classification network learns one-vs-all action classification model to serve as initialization for the localization network; and (3) a localization network fine-tunes the learned classification network to localize each action instance. In another approach, frame-level action detection is effectively formulated as the per-frame action labeling task. We combine two reverse operations (*i.e.* convolution and deconvolution) into a joint Convolutional-De-Convolutional (CDC) filter, which simultaneously conducts downsampling in space and upsampling in time to jointly model both high-level semantics and temporal dynamics. We design a novel CDC network to predict actions at frame-level and the frame-level predictions can be further used to detect precise segment boundary for the temporal action localization task. Our method not only improves the state-of-the-art mean Average Precision (mAP) result on THUMOS'14 from 41.3% to 44.4% for the per-frame labeling task, but also improves mAP for the temporal action localization task from 19.0% to 23.3% on THUMOS'14 and from 16.4% to 23.8% on ActivityNet v1.3.

Action detection in the constrained scenarios: The usual training process of deep learning models consists of supervision and data, which are not always available in reality. In Part II, we consider the scenarios of incomplete supervision and incomplete data. For incomplete supervision, we focus on the weakly-supervised temporal action localization task and propose AutoLoc which is the first framework that can directly predict the temporal boundary of each action instance with only the video-level annotations available during training. To enable the training of such a boundary prediction model, we design a novel Outer-Inner-Contrastive (OIC) loss to help discover the segment-level supervision and we prove that the OIC loss is differentiable to the underlying boundary prediction model. Our method significantly improves mAP on THUMOS14 from 13.7% to 21.2% and mAP on ActivityNet from 7.4% to 27.3%. For the scenario of incomplete data, we formulate a novel task called Online Detection of Action Start (ODAS) in streaming videos to enable detecting the action start time on the fly when a live video action is just starting. ODAS is important in many applications such as early alert generation to allow timely security or emergency response. Specifically, we propose three novel methods to address the challenges in training ODAS models: (1) hard negative samples generation based on Generative Adversarial Network (GAN) to distinguish ambiguous background, (2) explicitly modeling the temporal consistency between data around action start and data succeeding action start, and (3) adaptive sampling strategy to handle the scarcity of training data.

Action understanding in the compressed domain: The mainstream action understanding methods including the aforementioned techniques developed by us require first decoding the compressed video into RGB image frames. This may result in significant cost in terms of storage and computation. Recently, researchers started to investigate how to directly perform action understanding in the compressed domain in order to achieve high efficiency while maintaining the state-of-the-art action detection accuracy. The key research challenge is developing effective backbone networks that can directly take data in the compressed domain as input. Our baseline is to take models developed for action understanding in the decoded domain and adapt them to attack the same tasks in the compressed domain. In Part III, we address two important issues in developing the backbone networks that exclusively operate in the compressed domain. First, compressed videos may be produced by different encoders or encoding parameters, but it is impractical to train a different compressed-domain action understanding model for each different format. We experimentally analyze the effect of video encoder variation and develop a simple yet effective training data preparation method to alleviate the sensitivity to encoder variation. Second, motion cues have been shown to be important for action understanding, but the motion vectors in compressed video are often very noisy and not discriminative enough for directly performing accurate action understanding. We develop a novel and highly efficient framework called DMC-Net that can learn to predict discriminative motion cues based on noisy motion vectors and residual errors in the compressed video streams. On three action recognition benchmarks, namely HMDB-51, UCF101 and a subset of Kinetics, we demonstrate that our DMC-Net can significantly shorten the performance gap between state-of-the-art compressed video based methods with and without optical flow, while being two orders of magnitude faster than the methods that use optical flow.

By addressing the three major challenges mentioned above, we are able to develop more robust models for video action understanding and improve performance in various dimensions, such as (1) temporal precision, (2) required levels of supervision, (3) live video analysis ability, and finally (4) efficiency in processing compressed video. Our research has contributed significantly to advancing the state of the art of video action understanding and expanding the foundation for comprehensive semantic understanding of video content.

Table of Contents

List of Figures

List of Tables

Glossary

mAP - mean Average Precision

CNN - Convolutional Neural Networks

RNN - Recurrent Neural Networks

LSTM - Long Short-Term Memory

STIP - Space-Time Interest Points

iDTF - improved Dense Trajectory Feature

CDC - Convolutional-De-Convolutional

HOG - Histogram of Gradient

HOF - Histogram of Flow

MBH - Motion Boundary Histograms

FV - Fisher Vector

TAL - Temporal Action Localization

S-CNN - Segment-CNN

CAS - Class Activation Sequence

OIC - Outer-Inner-Contrastive

AS - Action Start

AE - Action End

ODAS - Online Detection of Action Start

GOP - Group Of Pictures

DMC - Discriminative Motion Cues

I - I-frame

MV - Motion Vector

R - Residual

DMC-Net - Discriminative Motion Cues Networks

EPE - End-Point Error

Acknowledgments

First of all, I would like to express my deep gratitude to my adviser, Professor Shih-Fu Chang, for his fabulous guidance and continuous support. Five years ago, I was an newbie to research. It was Prof. Chang who taught me every details about research from professional presentation to critical thinking, instructed me how to understand the key in challenging questions and explain complex things concisely and clearly, enlightened me by his deep insights and wide knowledge, and shaped my research attitude by urging me to do rigorous and solid work of profound impact. I feel extremely fortunate to work with him and I will keep learning from his wisdom in the future.

I would like to thank Prof. John Wright, Prof. John Paisley, Prof. Mingoo Seok and Prof. Carl Vondrick for serving as my dissertation committee members and attending my PhD thesis defense. Thanks go to Elsa, Karin and other EE staffs who helped me in this five years.

I want to thank Horace Meng and Tao Chen who warmly hosted my first internship at California. My second internship is at Microsoft Research in Redmond with Lei Zhang, who is a wonderful mentor and has been helping me a lot. Thank you Lei. I appreciate the close and fruitful collaboration with Zhicheng Yan, Yannis Kalantidis, Laura Sevilla-Lara and Marcus Rohrbach during my third internship at Facebook Research. Thanks also go to other mentors and colleagues who have helped me and discussed with me during these internships.

I would like to thank Mitsubishi Electric and Wei Family Private Foundation for supporting my PhD study. I enjoy the collaboration with researchers in Mitsubishi including Kazuyuki Miyazawa, Hassan Mansour, Anthony Vetro and Dong Tian. I also want to thank my other co-authors: Dongang Wang, Jonathan Chan, Alireza Zareian, Hang Gao, Junting Pan, Xavi Gir-i-Nieto, Hanwang Zhang, Tianwei Lin, Xu Zhao, Xudong Lin, Hongyi Liu, Du Tran, Jamie Ray, Manohar Paluri.

Thanks to other researchers who helped me in my research and career including Ju Sun, Wei Liu, Matt Feiszli, Dong Liu, Guangnan Ye, Liangliang Cao, etc. There are many other friends that I want to thank, including DVMM labmates, friends I met during undergraduate at Wuhan and graduate at Columbia, and friends I met during internships. I wish I could have listed them all.

"Beauty is a characteristic that provides a perceptual experience of pleasure or satisfaction". This was the first sentence in my personal statement for PhD application. During the past five years, I have been very lucky to immensely enjoy the beauty in computer vision and deep learning thanks to the help from many beautiful people. I hope in the future I can make full use of what I have learned during my PhD to make this beautiful world even more beautiful.

Finally, I want to thank my girlfriend, Xinyu Hu, whose accompanying made my last two years at Columbia colorful and enjoyable. Thanks also go to my family members worldwide. Notably, my special appreciation is dedicated to my parents, who inspire and care me so much and wish me all the best with their unconditional and endless love all the time. I sincerely wish them good health and happiness forever. This thesis would not have been possible without their love and support.

This work is dedicated to my dear parents.

Chapter 1

Introduction

1.1 Motivation

In the current era of data explosion, large amounts of videos have been generated by various video capturing devices and tremendous users on the Internet and social media. Therefore, there is an urgent need to develop intelligent system that can automatically analyze video contents. To this end, one key and unique technical challenge is understanding actions contained in video. For example, YouTube needs to understand which sports actions contained in each Olympics video to provide more accurate recommendations; autonomous driving car requires understanding of actions performed by the surrounding cars and pedestrians.

Over the years, researchers in computer vision have developed a lot of hand-crafted features for action understanding such as Space-Time Interest Points (STIP) [1], improved Dense Trajectory Feature (iDTF) [2], etc. Recently, deep learning has made a breakthrough in image understanding, achieving state-of-the-art performances in image classification, object detection, image segmentation, etc. However, since actions in video characterize movements and usually span over a sequence of frames, it remains unclear how to utilize deep neural networks to understand actions. Therefore, we motivate this thesis by first identifying important research issues to be addressed for comprehensively understanding actions in video.

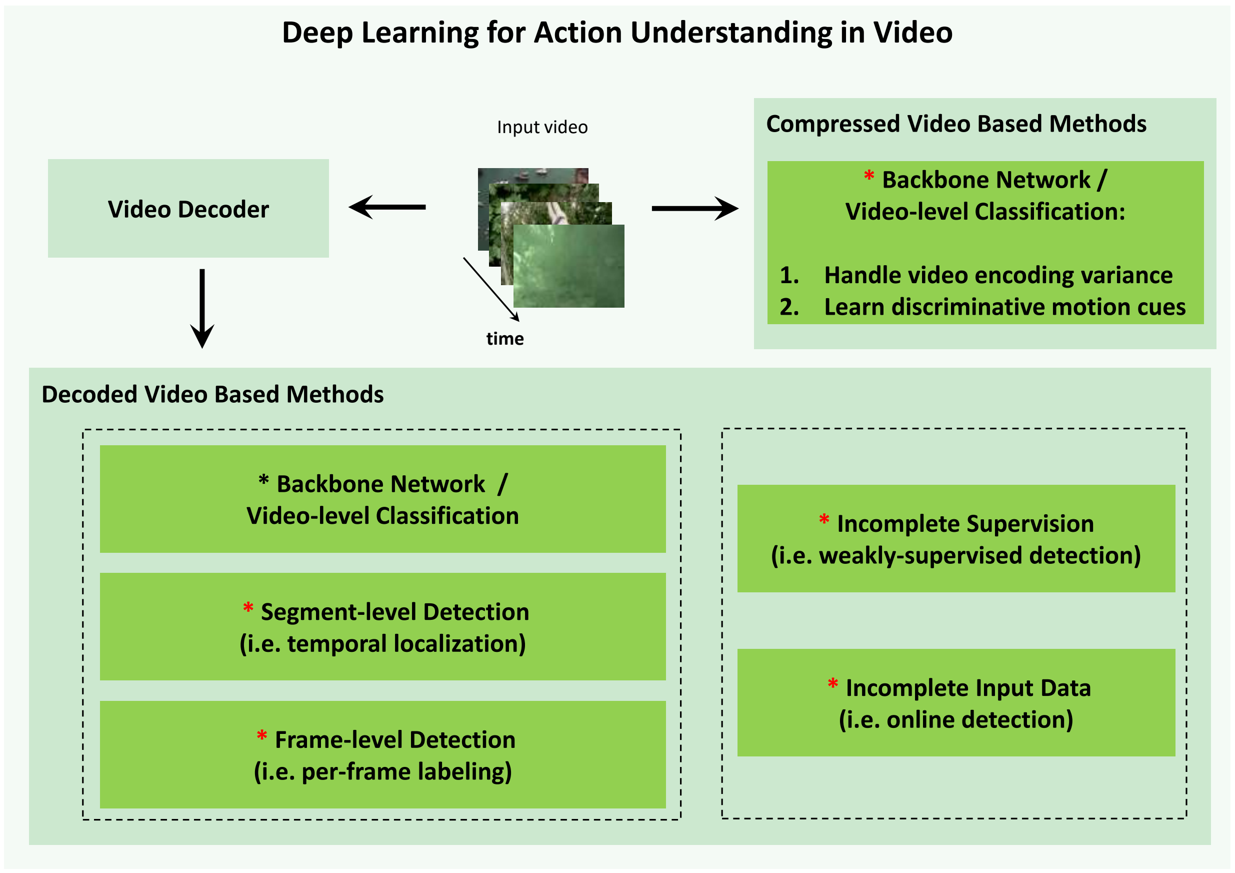

Deep Learning for Action Understanding in Video
Input video
time
Video Decoder
Compressed Video Based Methods
* Backbone Network / Video-level Classification:
1. Handle video encoding variance
2. Learn discriminative motion cues
Decoded Video Based Methods
* Backbone Network / Video-level Classification
* Segment-level Detection (i.e. temporal localization)
* Frame-level Detection (i.e. per-frame labeling)
* Incomplete Supervision (i.e. weakly-supervised detection)
* Incomplete Input Data (i.e. online detection)

Figure 1.1: The overview of research issues in developing deep learning methods for action understanding in video. * indicates issues addressed in previous work and * indicates where contributions are made in this thesis.

As shown in Figure 1.1, each input video is effectively a collection of consecutive image frames. In real applications, video usually is compressed for the sake of accelerating transmission and saving storage. To analyze the compressed video, there are two directions as discussed in the following Section 1.1.1 and Section 1.1.2 respectively.

1.1.1 Decoded Video Based Methods

Prevalent action understanding methods first decode the compressed video into RGB image frames. Based on the decoded frames, researchers have developed quite a few successful **backbone networks** specifically for analyzing video. To compare various backbone networks, evaluation is usually based on the **video-level classification** task: given one input video, we need to detect which action class contained in this whole video.

To comprehensively understand actions, it is important to go beyond video-level classification and detect actions at finer granularities including (1) **segment-level detection** which aims at localizing temporal boundaries (start time and end time) of each action segment and (2) **frame-level detection** which targets classifying every frame into background or specific action classes. With the aforementioned backbone networks on hand, the unresolved technical challenge here is designing new detection frameworks and architectures to handle the unique training supervision and expected outputs in the segment-level detection and frame-level detection.

The input of deep learning models consists of supervision and data, which are not always completely available in some realistic scenarios. (1) Incomplete supervision: annotating the start time and end time of each action instance is quite expensive and thus it is worthwhile researching **weakly-supervised detection** methods that during training we only have videos with the video-level action labels, but during testing we need to predict both the action class and the temporal boundary of each action instance. (2) Incomplete data: many real applications requires continuously monitoring the live video stream without seeing the future frames and therefore **online detection of action start**: given a streaming video, we need to immediately process each newly arrived frame to detect the occurrence and class of each newly started action in real time.

1.1.2 Compressed Video Based Methods

There is an emerging attention in directly analyzing video in the compressed domain without video decoding because bypassing RGB frame extraction and optical flow computation can save considerable running time during inference. When switching from the decoded domain to the compressed domain, the main technical challenge for action understanding lies in the design of backbone network. Once we have the backbone network ready in hand, it is not difficult to adapt methods developed based on decoded video to address various action understanding tasks in the compressed domain. Regarding the backbone network for compressed video input, Wu *et al.* [3] recently proposed a multiple-stream framework which processes i-frame image, motion vector and residual respectively with individual networks and finally conducts late fusion. However, two important research issues still remain unresolved: videos can be compressed by different video codecs with different encoding configurations and thus we need to study how such **video encoding variations** would affect the performance of action understanding; motion vectors are usually noisy and have low-resolution and therefore it is important to explore how to obtain more **discriminative motion cues** from compressed video in order to perform more accurate action understanding.

1.2 Our Techniques and Contributions

1.2.1 Action Detection at Fine Granularities in Time

In Part I, we improve deep learning based action detection frameworks at two different granularities. In Chapter 3, we propose an effective multi-stage framework called Segment-CNN to detect actions at segment-level (i.e. temporal action localization) in untrimmed, long videos. We exploit the effectiveness of deep networks in temporal action localization via three segment-based 3D ConvNets: (1) a proposal network identifies candidate segments in a long video that may contain actions; (2) a classification network learns one-vs-all action classification model to serve as initialization for the localization network; and (3) a localization network fine-tunes the learned classification network to localize each action instance. We propose a novel loss function for the localization network to explicitly consider temporal overlap and achieve high temporal localization accuracy. In the end, only the proposal network and the localization network are used during prediction. Our approach achieves significantly superior performances compared with other state-of-the-art systems: mAP increases from 1.7% to 7.4% on MEXaction2 and increases from 15.0% to 19.0% on THUMOS'14. Our ablation studies confirm that the proposal network improves the efficiency by eliminating unlikely candidate segments and the localization network is key to temporal localization accuracy boosting. In Chapter 4, we propose a novel Convolutional-De-Convolutional (CDC) filter to simultaneously perform spatial downsampling (for spatio-temporal semantic abstraction) and temporal upsampling (for precise temporal localization). Further, we design a CDC network to detect actions at frame-level and we further leverage the per-frame detection results to achieve precise segment-level detection. Our method not only improves mAP on THUMOS'14 from 41.3% to 44.4% for the per-frame labeling task, but also improves mAP for the temporal action localization task from 19.0% to 23.3% on THUMOS'14 and from 16.4% to 23.8% on ActivityNet v1.3.

1.2.2 Action Detection in the Constrained Scenarios

In Part II, we address action understanding in the constrained scenarios with incomplete supervision or incomplete input data. In Chapter 5, we develop a novel weakly-supervised temporal action localization framework called AutoLoc to directly predict temporal boundary in a single-shot fashion. In order to train the boundary prediction model with only the video-level labels, we propose a novel Outer-Inner-Contrastive (OIC) loss to automatically discover the segment-level supervision. With the direct boundary prediction technique and the discovered segment-level supervision, our method significantly improves mAP on THUMOS14 from 13.7% to 21.2% and mAP on ActivityNet from 7.4% to 27.3%. In Chapter 6, we formulate a novel Online Detection of Action Start (ODAS) task in a practical setting involving streaming, untrimmed videos. We propose three training methods to specifically improve the capability of ODAS models in detecting action timely and accurately: (1) hard negative samples generation based on Generative Adversarial Network (GAN) to distinguish ambiguous background, (2) explicitly modeling the temporal consistency between data around action start and data succeeding action start, and (3) adaptive sampling strategy to handle the scarcity of training data. Our methods can be applied to any existing video backbone network. During evaluations, our proposed methods significantly improve the state-of-the-art methods including shot detection methods, per-frame labeling methods and segment-level detection methods.

1.2.3 Action Understanding in the Compressed Domain

In Part III, we address important, unresolved research issues in designing the backbone networks for action understanding in the compressed domain. In Chapter 7, we study the effects of video encoding variations and experimentally benchmark several training data preparation methods. We find that training a generic model using videos transcoded from a certain format such as small macroblock size or large Group Of Pictures (GOP) size, can achieve performance comparable to training individual models that each model corresponds to one specific format, indicating that such a generic model is robust enough to handle the video encoding variations during testing. In Chapter 8, in order to obtain motion representations that are semantically discriminative and thus are suitable for understanding actions, we introduce a Discriminative Motion Cues Network (DMC-Net) for generating discriminative motion cues directly from data in the compressed video. We devise a highly efficient architecture for the generator network in DMC-Net. Since optical flow is a more accurate motion representation, we propose to train the generator to approximate flow using a reconstruction loss and a generative adversarial loss, jointly with the downstream action classification task. On three action recognition benchmarks, namely HMDB-51, UCF101 and a subset of Kinetics, we demonstrate that our DMC-Net can significantly shorten the performance gap between state-of-the-art compressed video based methods with and without optical flow, while being two orders of magnitude faster than the methods that use optical flow.

1.3 Thesis Overview

We provide a brief overview of this thesis. In Chapter 2, we present a brief literature survey on deep learning and then we review state-of-the-art work on video backbone network. The rest of this thesis is divided into 3 main parts: Part I focuses on improving deep learning based action detection frameworks at two different granularities, including Chapter 3 for segment-level detection and Chapter 4 for frame-level detection which can be further used to detect precise segment boundary for segment-level detection; Part II is dedicated to detecting actions in the constrained scenarios, containing Chapter 5 on weakly-supervised detection for incomplete supervision and Chapter 6 on online detection of action start for incomplete input data. Part III addresses unique challenges when directly performing action understanding in the compressed domain, containing Chapter 7 for studying the effects of video encoding variations and Chapter 8 for generating discriminative motion cues. These three parts are strongly correlated: all three parts are dedicated to understanding actions in video; Part I and Part II correspond to the settings that model inputs are respectively complete and incomplete; the backbone network studied in Part III can be incorporated with frameworks and ideas proposed in Part I and Part II to address specific action detection tasks in the compressed domain. Finally, Part IV (containing only Chapter 9) concludes the whole thesis by summarizing our contributions and discussing open issues.

Chapter 2

Background

2.1 Introduction

Recent years have witnessed the success of deep learning in computer vision, natural language processing, speech recognition, *etc*. In the rest of this chapter, Section 2.2 reviews popular deep network architectures with an emphasis on their applications in the video domain. Section 2.3 focuses on the video-level action classification task, which is typically used for studying and evaluating video backbone networks. In Section 2.3, we briefly review both the conventional models such as hand-crafted features and also the deep learning based models. Detailed surveys can be found in [4; 5; 6; 7]. These video-level action classification models can be used for feature extraction in the frameworks developed for various action detection tasks and scenarios as presented in Part I and Part II.

2.2 Deep Learning

2.2.1 Convolutional Neural Networks

In computer vision, Convolutional Neural Networks (CNN) have made a series of breakthroughs for image classification [8], object detection [9], image segmentation [10], *etc*. Back then in 2012, Alex *et al.* [8] first demonstrated the effectiveness of CNN on the challenging ImageNet dataset [11]. Then He *et al.* [12] proposed a residual learning framework called ResNet to ease the training of very deep networks. Further, Huang *et al.* [13] introduced Dense Convolutional Network (DenseNet) which connects each layer to every other layer in a feed-forward fashion.

In the video domain, Karpathy *et al.* [14] adopted 2D frame-level CNNs to address large-scale video classification tasks. Tran *et al.* [15; 16] proposed to learn spatio-temporal features in video with 3D Convolutional Networks.

2.2.2 Recurrent Neural Networks

In computer vision, Recurrent Neural Networks (RNN) have been adapted to address image captioning [17], video captioning [18], video action detection [19], *etc*. Karpathy *et al.* [17] used Bidirectional Recurrent Neural Networks to model sentences in generating image descriptions. Alahi *et al.* [20] utilized Long Short-Term Memory (LSTM) network to predict human trajectory in crowded spaces.

In the video domain, Donahue *et al.* [18] adopted LSTM network to perform video-level action classification and video captioning. Ma *et al.* [19] used LSTM to model activity progression in activity detection and early detection.

2.2.3 Adversarial Learning

Inspired by the seminal work of Goodfellow *et al.* [21] that demonstrated the power of adversarial training for image synthesis, a large number of works have experimented with incorporating adversarial loss for generative tasks like image-to-image translations [22; 23], super-resolution [24] or inpainting [25].

In the video domain, adversarial losses have been mostly used for frame prediction [26; 27; 28]. Compared to the conventional generative methods based on L2 reconstruction loss, which tends to produce blurry results, adversarial loss based methods can generate sharp and more realistic results.

2.3 Video-level Action Classification

2.3.1 Conventional Methods

There are several hand-crafted features that were developed for analyzing appearance and motion in video and are still used in some real applications nowadays. Laptev proposed the Space-Time Interest Points (STIP) which provide compact and abstract representations of spatio-temporal patterns in video [1]. Wang *et al.* [29] proposed extracting Histogram of Gradient (HOG), Histogram of Flow (HOF), Motion Boundary Histograms (MBH) features along dense trajectories, resulting in the Dense Trajectory Feature (DTF). Later, Wang *et al.* [30] proposed the improved Dense Trajectory Feature (iDTF) [2] to take camera motion into consideration and further incorporated iDTF with Fisher Vector (FV) encoding [30; 31; 32].

2.3.2 Deep Learning Methods

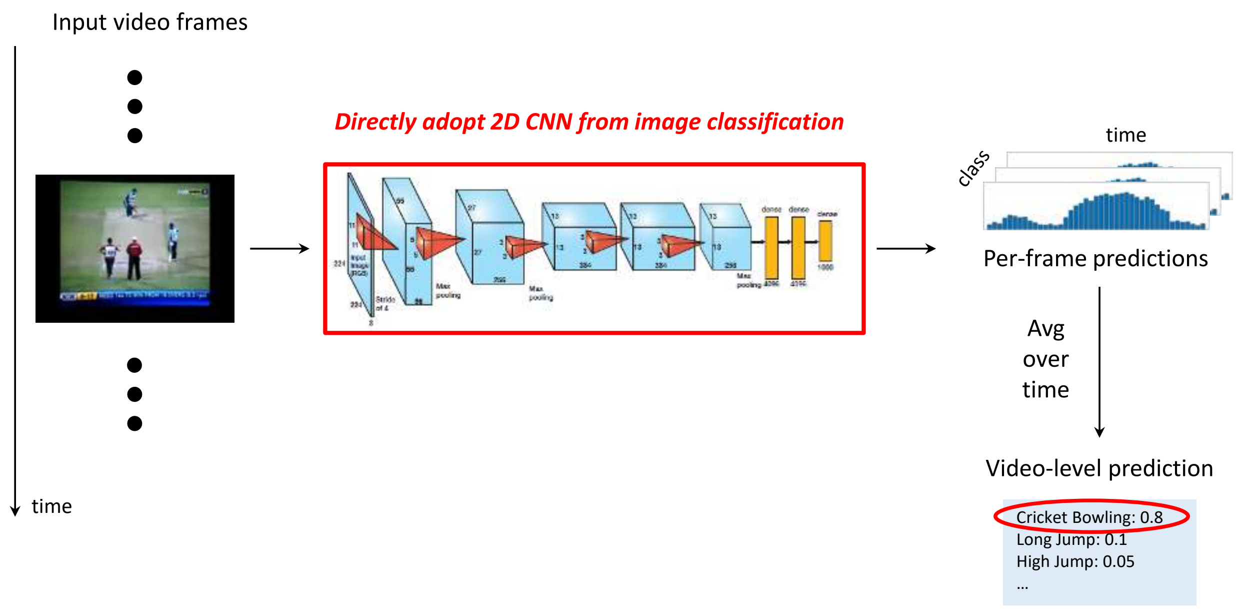


Figure 2.1: Illustration of directly adopting 2D CNN from image classification to process each video frame individually. The video-level prediction is obtained by averaging the predictions of all frames.

The mainstream deep learning methods for video-level action classification are based on the decoded video frames. In addition to the methods mentioned in the above Section 2.2, quite a few other deep learning based backbone models have been proposed to specifically address unique challenges in analyzing video. As shown in Figure 2.1, a simple yet popular approach is directly adopting 2D convolution developed for image classification to process each video frame and finally averaging predictions over all frames to obtain the video-level prediction. In order to comprehensively compare various forms of convolutions for action understanding, Tran *et al.* [33] experimented 2D convolution, 3D convolution and their combinations on the standard action classification benchmarks and found that the form of 2D spatial convolution followed by 1D temporal convolution, named as (2+1)D Convolution, yields significantly better accuracy than other forms. As shown in Figure 2.2, (2+1)D Convolution has less parameters to be learned and thus is easier to be optimized in practice compared to 3D convolution [33]. To model appearance and motion jointly, Simonyan and Zisserman [34] designed a two-stream framework to process appearance and motion separately with two individual networks as illustrated in Figure 2.3.

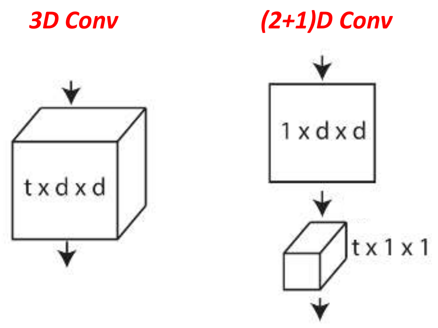


Figure 2.2: Comparisons between 3D Convolution and (2+1)D Convolution. 3D Convolution operates in space and time simultaneously. (2+1)D Convolution consists of a 2D spatial convolution followed by 1D temporal convolution.

Recently, researchers started to explore deep learning methods in the compressed domain. Wu *et al.* [3] proposed to directly process i-frame RGB image, motion vector and residual respec-

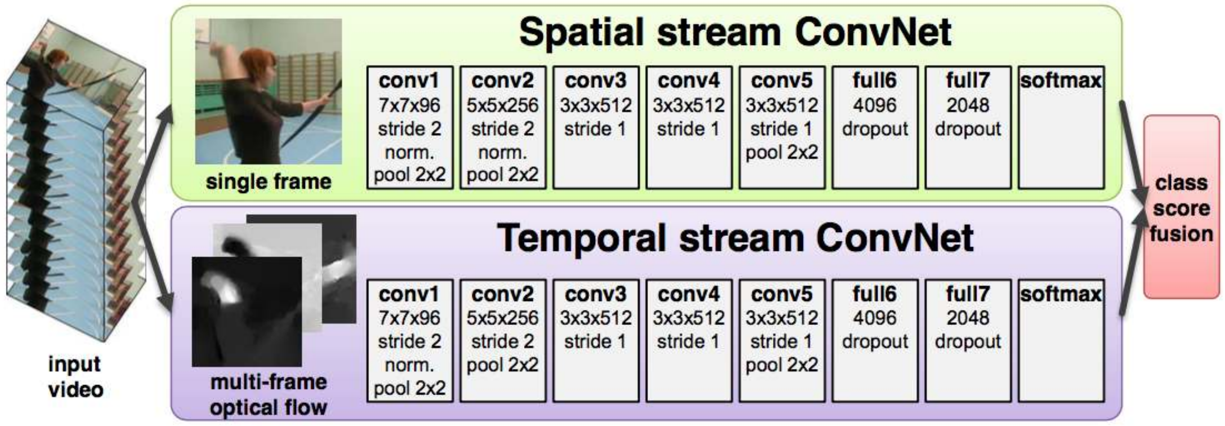


Figure 2.3: Illustration of two-stream network for video action classification. The spatial stream is for modeling appearance based on RGB data and the temporal stream is for modeling motion based on optical flow data. The predictions from these two streams are fused at the end.

tively with individual networks and finally perform late fusion. Although promising results can be achieved, there are important research issues that still remain unresolved for video-level classification in the compressed domain and therefore we explore these issues in Part III of this thesis.

Part I

Action Detection at Fine Granularities in Time

Chapter 3

Segment-level Action Detection

3.1 Introduction

Besides detecting action in manually trimmed short video via video-level classification, researchers start to develop techniques for detecting actions in untrimmed long videos in the wild. This trend motivates us to study another challenging task - **Temporal Action Localization (TAL)**: given a long, untrimmed video, "when does a specific action start and end?" This problem is important because real applications usually involve untrimmed videos, which can be highly unconstrained in space and time, and one video can contain multiple action instances plus background scenes or other activities. Localizing actions in long videos, such as those in surveillance, can save tremendous time and computational costs.

Most state-of-the-art methods rely on manually selected features, and their performances still require much improvement. For example, top performing approaches in THUMOS Challenge 2014 [35; 36; 37; 38] and 2015 [39; 40] both used improved Dense Trajectory (iDT) with Fisher Vector (FV) [30; 41]. There have been some recent attempts at incorporating iDT features with appearance features automatically extracted by frame-level deep networks [35; 36; 37]. Nevertheless, such 2D ConvNets do not capture motion information, which is important for modeling actions and determining their temporal boundaries.

As an analogy in still images, object detection recently achieved large improvements by using deep networks. Inspired by Region-based Convolutional Neural Networks (R-CNN) [42] and its upgraded versions [43; 9; 44], we develop Segment-CNN, which is an effective deep network framework for temporal action localization as outlined in Figure 3.1. We adopt 3D ConvNets [45; 46], which recently has been shown to be promising for capturing motion characteristics in videos, and add a new multi-stage framework. First, multi-scale segments are generated as candidates for three deep networks. The **proposal** network classifies each segment as either action or background in order to eliminate background segment estimated to be unlikely to contain actions of interest. The **classification** network trains typical one-vs-all classification model for all action categories plus the background.

However, the classification network aims at finding key evidences to distinguish different categories, rather than localizing precise action presences in time. Sometimes, the scores from the classification network can be high even when the segment has only a very small overlap with the ground truth instance. This can be detrimental because subsequent post-processing steps, such as

Non-Maximum Suppression (NMS), might remove segment of small score but large overlap with ground truth. To explicitly take temporal overlap into consideration, we introduce the **localization** network based on the same architecture, but this network uses a novel loss function, which rewards segments with higher temporal overlap with the ground truths, and thus can generate confidence scores more suitable for post-processing. Note that the classification network cannot be replaced by the localization network. We will show later that using the trained classification network (without considering temporal overlap) to initialize the localization network (take into account temporal overlap) is important, and achieves better temporal localization accuracies.

In the rest of this chapter, we review the related work in Section 3.2, describe the network architecture and the training and testing pipelines of the proposed Segment-CNN framework in Section 3.3, present the experimental results in Section 3.4, and finally draw our summary in Section 3.5.

3.2 Related Work

3.2.1 Temporal Action Localization

This topic has been studied in two directions. When training data only have video-level category labels but no temporal annotations, researchers formulated this as weakly supervised problems or multiple instance learning problems to learn the key evidences in untrimmed videos and temporally localize actions by selecting key instances [47; 48]. Sun *et al.* [49] transferred knowledge from web images to address temporal localization in untrimmed web videos.

Another line of work focuses on learning from data when the temporal boundaries have been annotated for action instances in untrimmed videos, such as THUMOS. Most of these works pose this as a classification problem and adopt a temporal sliding window approach, where each window is considered as an action candidate subject to classification [41]. Recently, two directions lead the state-of-the-art:

(1) Wang *et al.* [29] proposed extracting HOG, HOF, MBH features along dense trajectories, and later on they took camera motion into consideration [30]. Further improvement can be achieved by stacking features with multiple time skips [50]. Richard and Gall [51] proposed using statistical length and language modeling to represent temporal and contextual structures. Heilbron *et al.* [52] introduced a sparse learning framework for generating segment proposals of high recall.

(2) Inspired by the success of CNNs in recent works [8; 53], Karpathy *et al.* [14] evaluated frame-level CNNs on large-scale video classification tasks. Simonyan and Zisserman [34] designed two-stream CNNs to learn from still image and motion flow respectively. In [54], a latent concept descriptor of convolutional feature map was proposed, and great results were achieved on event detection with VLAD encoding. To learn spatio-temporal features together, the architecture of 3D ConvNets was explored in [45; 46], achieving competitive results. Oneata *et al.* [55] proposed approximately normalized Fisher Vectors to reduce the high dimensionality of FV. Stoian *et al.* [56] introduced a two-level cascade to allow fast search for action instances. Instead of precision, these methods focus on improving the efficiency of conventional methods. To

specifically address the temporal precision of action detection, Gaidon *et al.* [57; 58] modeled the structure of action sequence with atomic action units (actoms). The explicit modeling of action units allows for matching more complete action unit sequences, rather than just partial content. However, this requires manual annotations for actoms, which can be subjective and burdensome. Our Segment-CNN here aims to solve the same problem of precise temporal localization, but without requiring the difficult task of manual annotation for atomic action units. Meanwhile, RNN has been widely used to model temporal state transitions over frames: Escorcia *et al.* [59] built a temporal action proposal system based on Long-Short Term Memory (LSTM); Yeung *et al.* [60] used REINFORCE to learn decision policies for a RNN-based agent; Yeung *et al.* [61] introduced MultiTHUMOS dataset of multi-label annotations for every frame in THUMOS videos and defined a LSTM network to model multiple input and output connections; Yuan *et al.* [62] proposed a pyramid of score distribution feature at the center of each sliding window to capture the motion information over multiple resolutions, and utilized RNN to improve inter-frame consistency. In addition, Lea *et al.* [63] used temporal 1D convolution to capture scene changes when actions were being performed. Although RNN and temporal 1D convolution can model temporal dependencies among frames and make frame-level predictions, they are usually placed on top of deep ConvNets, which take a single frame as input, rather than directly modeling spatio-temporal characteristics in raw videos.

3.2.2 Spatio-temporal Action Localization

Spatio-temporal action localization requires localizing action in space and time simultaneously. This problem is different from temporal localization because spatio-temporal localization requires exhaustive annotations for objects of interest on every frame as training data. This makes it overwhelmingly time-consuming particularly for long, untrimmed videos compared with the task of simply labeling the start time and end time of an action depicted in the video, which is sufficient to satisfy many applications.

3.2.3 Object Detection

Inspired by the success of deep learning approaches in object detection, we also review R-CNN and its variations. R-CNN consists of selective search, CNN feature extraction, SVM classification, and bounding box regression [42]. Fast R-CNN reshapes R-CNN into a single-stage using multi-task loss, and also has a RoI pooling layer to share the computation of one image in ConvNets [43]. Our work differs from R-CNN in the following aspects: (1) Temporal annotations in training videos can be diverse: some are cleanly trimmed action instances cut out from long videos, such as UCF101 [64], and some are untrimmed long videos but with temporal boundaries annotated for action instances, such as THUMOS [38; 40]. We provide a paradigm that can handle such diverse annotations. (2) As proven in Faster R-CNN [9] which proposes region proposal network, and DeepBox [44] which detects objectness to re-rank the results of R-CNN, using deep networks for learning objectness is effective and efficient. Therefore, we directly use deep network to classify background and action to obtain candidate segments. (3) We remove the regression stage because learning regression for time shift and duration of video segment does not work well in our experiments, probably because actions can be quite diverse, and therefore do not contain consistent patterns for predicting start/end time. To achieve precise localization, we design the localization network using a new loss function to explicitly consider temporal overlap. This can decrease the score for the segment that has small overlap with the ground truth, and increase the segment of larger overlap. This also benefits post-processing steps, such as NMS, to keep segment with higher temporal localization accuracy.

3.3 Detailed Descriptions of Segment-CNN

3.3.1 Problem Setup

Problem definition. We denote a video as $X = \{x_t\}_{t=1}^{T}$ where x_t is the t-th frame in X, and T is the total number of frames in X. Each video X is associated with a set of temporal action annotations $\Psi = \left\{\left(\psi_m, \psi_m^{'}, k_m\right)\right\}_{m=1}^{M}$, where M is the total number of action instances in X, and k_m, ψ_m, $\psi_m^{'}$ are, respectively, action category of the instance m and its starting time and ending time (measured by frame ID). $k_m \in \{1, \ldots, K\}$, where K is the number of categories. During training, we have a set $\mathcal{T}$ of trimmed videos and a set $\mathcal{U}$ of untrimmed videos. Each trimmed video $X \in \mathcal{T}$ has $\psi_m = 1$, $\psi_m^{'} = T$, and $M = 1$.

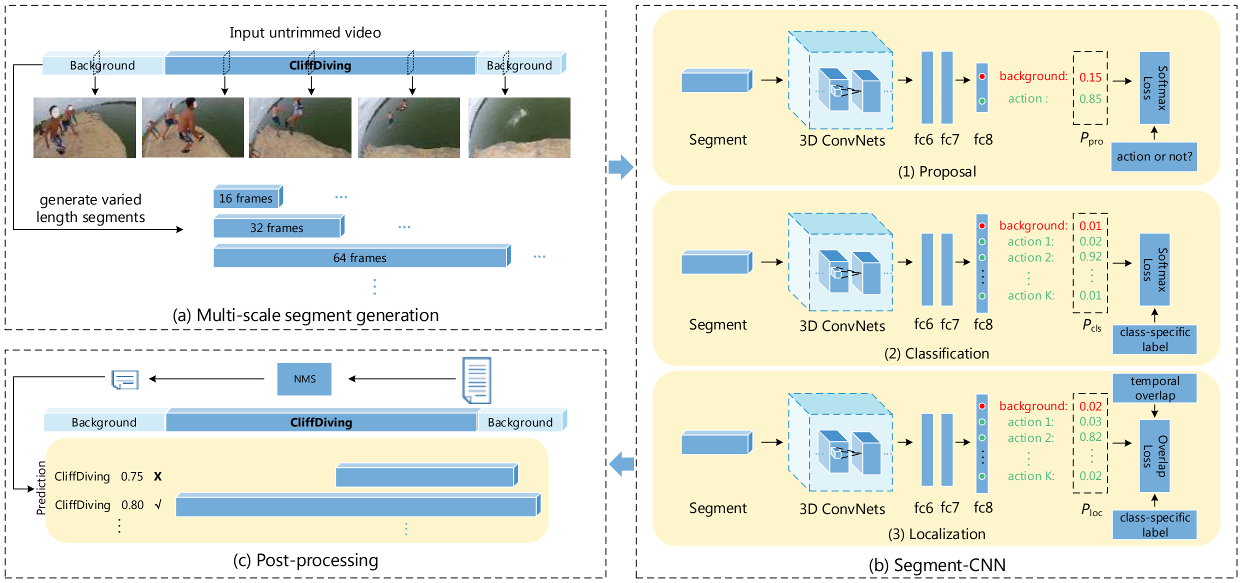


Figure 3.1: Overview of our framework. (a) Multi-scale segment generation: given an untrimmed video, we generate segments of varied lengths via sliding window; (b) Segment-CNN: the **proposal** network identifies candidate segments, the **classification** network trains an action recognition model to serve as initialization for the localization network, and the **localization** network localizes action instances in time and outputs confidence scores; (c) Post-processing: using the prediction scores from the localization network, we further remove redundancy by NMS to obtain the final results. During training, the classification network is first learned and then used as initialization for the localization network. During prediction, only the proposal and localization networks are used.

Multi-scale segment generation. First, each frame is resized to 171 (width) $\times$ 128 (height) pixels. For untrimmed video $X \in \mathcal{U}$, we conduct temporal sliding windows of varied lengths as 16, 32, 64, 128, 256, 512 frames with 75% overlap. For each window, we construct segment s by uniformly sampling 16 frames. Consequently, for each untrimmed video X, we generate a set of candidates $\Phi = \left\{ \left(s_h, \phi_h, \phi_h^{'} \right) \right\}_{h=1}^{H}$ as input for the proposal network, where H is the total number of sliding windows for X, and ϕ_m and $\phi_m^{'}$ are respectively starting time and ending time of the h-th segment s_h. For trimmed video $X \in \mathcal{T}$, we directly sample a segment s of 16 frames from X in uniform.

Network architecture. 3D ConvNets conducts 3D convolution/pooling which operates in spatial and temporal dimensions simultaneously, and therefore can capture both appearance and motion for action. Given the competitive performances on video classification tasks, our deep networks use 3D ConvNets as the basic architecture in all stages and follow the network architecture of [46]. All 3D pooling layers use max pooling and have kernel size of 2×2 in spatial with stride 2, while vary in temporal. All 3D convolutional filters have kernel size 3 and stride 1 in all three dimensions. Using the notations $\texttt{conv}$(number of filters) for the 3D convolutional layer, $\texttt{pool}$(temporal kernel size, temporal stride) for the 3D pooling layer, and $\texttt{fc}$(number of filters) for the fully connected layer, the layout of these three types of layers in our architecture is as follows: $\texttt{conv1a}(64)$ - $\texttt{pool1}(1,1)$ - $\texttt{conv2a}(128)$ - $\texttt{pool2}(2,2)$ - $\texttt{conv3a}(256)$ - $\texttt{conv3b}(256)$ - $\texttt{pool3}(2,2)$ - $\texttt{conv4a}(512)$ - $\texttt{conv4b}(512)$ - $\texttt{pool4}(2,2)$ - $\texttt{conv5a}(512)$ - $\texttt{conv5b}(512)$ - $\texttt{pool5}(2,2)$ - $\texttt{fc6}(4096)$ - $\texttt{fc7}(4096)$ - $\texttt{fc8}(K+1)$. Each input for this deep network is a segment s of dimension $171 \times 128 \times 16$. C3D is training this network on Sports-1M train split [46], and we use C3D as the initialization for our proposal and classification networks.

3.3.2 Training Procedure

The proposal network: We train a CNN network Θ_{pro} as the background segment filter. Basically, $\texttt{fc8}$ has two nodes that correspondingly represent the background (rarely contains action of interest) and being-action (has significant portion belongs to the actions of interest).

We use the following strategy to construct training data $\mathcal{S}_{\text{pro}} = \{(s_n, k_n)\}_{n=1}^{N}$, where label $k_n \in \{0, 1\}$. For each segment of the trimmed video $X \in \mathcal{T}$, we set its label as positive. For candidate segments from an untrimmed video $X \in \mathcal{U}$ with temporal annotation Ψ, we assign a label for each segment by evaluating its Intersection-over-Union (IoU) with each ground truth instance in Ψ : (1) if the highest IoU is larger than 0.7, we assign a positive label; (2) if the highest IoU is smaller than 0.3, we set it as the background. On the perspective of ground truth, if there is no segment that overlaps with a ground truth instance with IoU larger than 0.7, then we assign a positive label segment s if s has the largest IoU with this ground truth and its IoU is higher than 0.5. At last, we obtain $\mathcal{S}_{\text{pro}} = \{(s_n, k_n)\}_{n=1}^{N_{\text{pro}}}$ which consists of all $N_{\mathcal{T}} + N_{\mathcal{U}}$ positive segments and $N_b \approx N_{\mathcal{T}} + N_{\mathcal{U}}$ randomly sampled background segments, where $N_{\text{pro}} = N_{\mathcal{T}} + N_{\mathcal{U}} + N_b$.

In all experiments, we use a learning rate of 0.0001, with the exception of 0.01 for `fc8`, momentum of 0.9, weight decay factor of 0.0005, and drop the learning rate by a factor of 10 for every 10K iterations. The number of total iterations depends on the scale of dataset and will be clarified in Section 3.4.

Note that, compared with the multi-class classification network, this proposal network is simpler because the output layer only consists of two nodes (action or background).

The classification network: After substantial background segments are removed by the proposal network, we train a classification model Θ_{cls} for K action categories as well as background.

Preparing the training data $\mathcal{S}_{\text{cls}}$ follows a similar strategy for the proposal network. Except when assigning label for positive segment, the classification network explicitly indicates action category $k_m \in \{1, \ldots, K\}$. Moreover, in order to balance the number of training data for each class, we reduce the number of background instances to $N_b \approx \frac{N_{\mathcal{T}}+N_{\mathcal{U}}}{K}$.

As for parameters in SGD, the learning rate is 0.0001, with the exception of 0.01 for `fc8`, momentum is 0.9, weight decay factor is 0.0005, and the learning rate is divided by a factor of 2 for every 10K iterations, because the convergence shall be slower when the number of classes increases.

The localization network: As illustrated in Figure 3.2, it is important to push up the prediction score of the segment with larger overlap with the ground truth instance and decrease the scores of the segment with smaller overlap, to make sure that the subsequent post-processing steps can choose segments with higher overlap over those with small overlap. Therefore, we propose this localization network Θ_{loc} with a new loss function, which takes IoU with ground truth instance

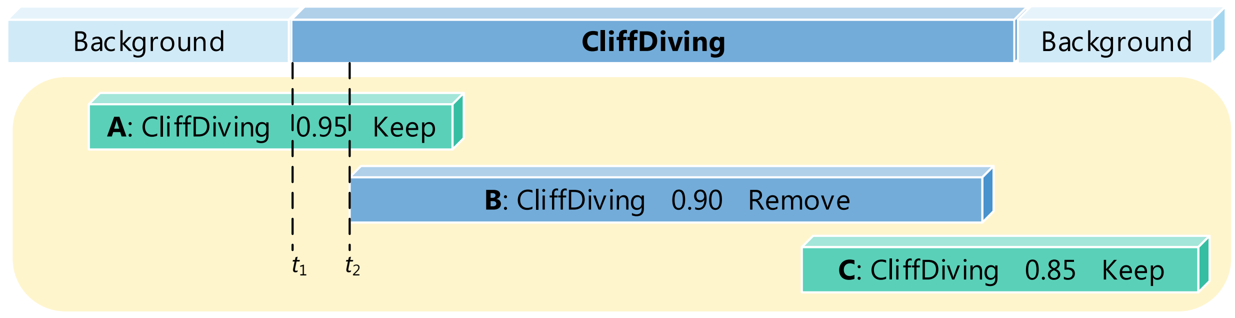


Figure 3.2: Typical case of bad localizations. Assume that the system outputs three predictions: A, B, C. Probably due to that there are some evidences during $[t_1, t_2]$, and A has the highest prediction score. Therefore, the NMS will keep A, remove B, and then keep C. However, actually we hope to remove A and C in NMS, and keep B because B has the largest IoU with the ground truth instance.

into consideration.

Training data $\mathcal{S}_{\text{loc}}$ for the localization network are augmented from $\mathcal{S}_{\text{cls}}$ by associating each segment s with the measurement of overlap, v. In specific, we set $v = 1$ for s from trimmed video. If s comes from untrimmed video and has positive label k, we set v equal to the overlap (measured by IoU) of segment s with the associated ground truth instance. If s is a background segment, as we can see later, its overlap measurement v will not affect our new loss function and gradient computation in back-propagation, and thus we simply set its v as 1.

During each mini-batch, we have N training samples $\{(s_n, k_n, v_n)\}_{n=1}^{N}$. For the n-th segment, the output vector of `fc8` is O_n and the prediction score vector after the softmax layer is P_n. Note that for the i-th class, $P_n^{(i)} = \frac{e^{O_n^{(i)}}}{\sum_{j=1}^{N} e^{O_n^{(j)}}}$. The new loss function is formed by combining $\mathcal{L}_{\text{softmax}}$ and $\mathcal{L}_{\text{overlap}}$:

$$\mathcal{L} = \mathcal{L}_{\text{softmax}} + \lambda \cdot \mathcal{L}_{\text{overlap}}, \tag{3.1}$$

where λ balances the contribution from each part, and through empirical validation, we find that $\lambda = 1$ works well in practice. $\mathcal{L}_{\text{softmax}}$ is the conventional softmax loss and is defined as

$$\mathcal{L}_{\text{softmax}} = \frac{1}{N} \sum_{n} \left(-\log \left(P_n^{(k_n)} \right) \right), \tag{3.2}$$

which is effective for training deep networks for classification. $\mathcal{L}_{\text{overlap}}$ is designed to jointly reduce the classification error and adjust the intensity of confidence score according to the extent of overlap:

$$\mathcal{L}_{\text{overlap}} = \frac{1}{N} \sum_{n} \left(\frac{1}{2} \cdot \left(\frac{\left(P_n^{(k_n)} \right)^2}{(v_n)^{\alpha}} - 1 \right) \cdot [k_n > 0] \right). \tag{3.3}$$

Here, $[k_n > 0]$ is equal to 1 when the true class label k_n is positive, and it is equal to 0 when $k_n = 0$, which means the s_n is a background training sample. $\mathcal{L}_{\text{overlap}}$ is intended to boost the detection scores (P) of segments that have high overlaps (v) with ground truth instances, and reduce the scores of those with small overlaps. The hyper-parameter α controls the adjustment range for the intensity of the confidence score. The sensitivity of α is explored in Section 3.4. In addition, the total gradient w.r.t output of the i-th node in `fc8` is as follows:

$$\frac{\partial \mathcal{L}}{\partial O_n^{(i)}} = \frac{\partial \mathcal{L}_{\text{softmax}}}{\partial O_n^{(i)}} + \lambda \cdot \frac{\partial \mathcal{L}_{\text{overlap}}}{\partial O_n^{(i)}}, \tag{3.4}$$

in which

$$\frac{\partial \mathcal{L}_{\text{softmax}}}{\partial O_n^{(i)}} = \begin{cases} \frac{1}{N} \cdot \left(P_n^{(k_n)} - 1\right) & \text{if } i = k_n \\ \frac{1}{N} \cdot P_n^{(i)} & \text{if } i \neq k_n \end{cases} \tag{3.5}$$

and

$$\frac{\partial \mathcal{L}_{\text{overlap}}}{\partial O_n^{(i)}} = \begin{cases} \frac{1}{N} \cdot \left(\frac{\left(P_n^{(k_n)}\right)^2}{(v_n)^\alpha} \cdot \left(1 - P_n^{(k_n)}\right) \right) \cdot [k_n > 0] \\ \qquad \text{if } i = k_n \\ \frac{1}{N} \cdot \left(\frac{\left(P_n^{(k_n)}\right)^2}{(v_n)^\alpha} \cdot \left(-P_n^{(i)}\right) \right) \cdot [k_n > 0] \\ \qquad \text{if } i \neq k_n \end{cases} . \tag{3.6}$$

Given a training sample (s_n, k_n, v_n), Figure 3.3 shows how $\mathcal{L}_{\text{overlap}}$ influences the original softmax loss. It also provides more concrete insights about the design of this loss function. (1) If the segment belongs to the background, $\mathcal{L}_{\text{overlap}} = 0$ and $\mathcal{L} = \mathcal{L}_{\text{softmax}}$. (2) If the segment is positive, $\mathcal{L}$ reachs the minimum at $P_n^{(k_n)} = \sqrt{(v_n)^\alpha}$, and therefore penalizes two cases: either $P_n^{(k_n)}$ is too small due to misclassification, or $P_n^{(k_n)}$ explodes and exceeds the learning target $\sqrt{(v_n)^\alpha}$

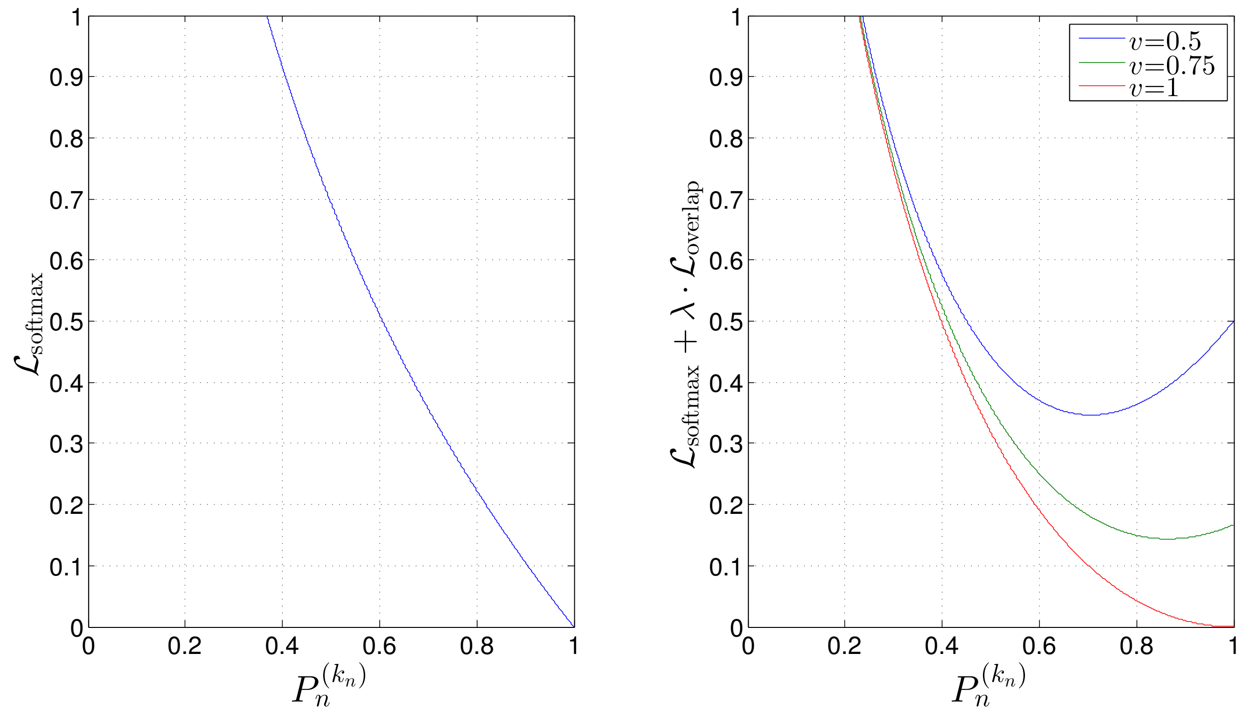


Figure 3.3: An illustration of how $\mathcal{L}_{\text{overlap}}$ works compared with $\mathcal{L}_{\text{softmax}}$ for each positive segment. Here we use $\alpha = 1$, $\lambda = 1$, and vary overlap v in $\mathcal{L}_{\text{overlap}}$. The x-axis is the prediction score at the node that corresponds to true label, and the y-axis is the loss.

which is proportional to overlap v_n. Also note that $\mathcal{L}$ is designed to increase as v_n decreases, considering that the training segment with smaller overlap with ground truth instance is less reliable because it may include considerable noise. (3) In particular, if this positive segment has overlap $v_n = 1$, the loss function becomes similar to the softmax loss, and $\mathcal{L}$ gradually decreases from $+\infty$ to 1 as $P_n^{(k_n)}$ goes from 0 to 1.

During optimization, Θ_{loc} is fine-tuned on Θ_{cls}. Because doing classification is also one objective of the localization network, and a trained classification network can be good initialization. We use the same learning rate, momentum, and weight decay factor as for the classification network.

Other parameters depending on the dataset are indicated in Section 3.4.

3.3.3 Prediction and Post-processing

During prediction, we slide varied length temporal window to generate a set of segments and input them into Θ_{pro} to obtain proposal scores P_{pro}. In this thesis, we keep segments with $P_{\text{pro}} \geq 0.7$. Then we evaluate the retained segments by Θ_{loc} to obtain action category predictions and confidence scores P_{loc}. During post-processing, we remove all segments predicted as the background and refine P_{loc} by multiplying with class-specific frequency of occurrence for each window length in the training data to leverage window length distribution patterns. Finally, because redundant detections are not allowed in evaluation, we conduct NMS based on P_{loc} to remove redundant detections, and set the overlap threshold in NMS to a little bit smaller than the overlap threshold θ in evaluation ($\theta - 0.1$ in this thesis).

3.4 Experiments

3.4.1 Datasets and Setup

MEXaction2 [65]. This dataset contains two action classes: "BullChargeCape" and "HorseRiding". This dataset consists of three subsets: INA videos, YouTube clips, and UCF101 Horse Riding clips. YouTube clips and UCF101 Horse Riding clips are trimmed, whereas INA videos are untrimmed and are approximately 77 hours in total. With regard to action instances with temporal

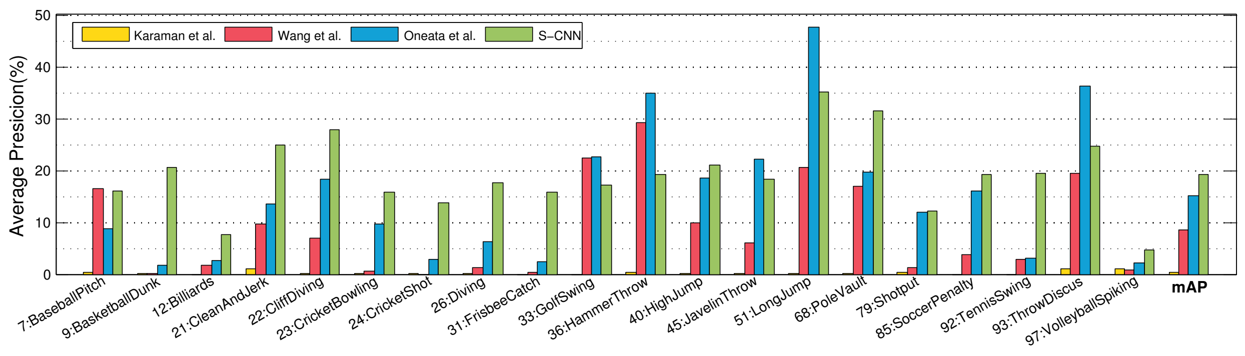


Figure 3.4: Histogram of average precision (%) for each class on THUMOS 2014 when the overlap threshold is set to 0.5 during evaluation.

annotation, they are divided into train set (1336 instances), validation set (310 instances), and test set (329 instances).

THUMOS 2014 [38]. The temporal action detection task in THUMOS Challenge 2014 is dedicated to localizing action instances in long untrimmed videos. The detection task involves 20 categories as indicated in Figure 3.4. The trimmed videos used for training are 2755 videos of these 20 actions in UCF101. The validation set contains 1010 untrimmed videos with temporal annotations of 3007 instances in total. The test set contains 3358 action instances from 1574 untrimmed videos, whereas only 213 of them contain action instances of interest. We exclude the remaining 1361 background videos in the test set.

3.4.2 Comparison with state-of-the-art systems

Evaluation metrics. We follow the conventional metrics used in THUMOS Challenge to regard temporal action localization as a retrieval problem, and evaluate average precision (AP). A prediction is marked as correct only when it has the correct category prediction, and has IoU with ground truth instance larger than the overlap threshold (measured by IoU). Note that redundant detections are not allowed.

Results on MEXaction2. We build our system based on Caffe [66] and C3D [46]. We use the train set in MEXaction2 for training. The number of training iterations is 30K for the proposal network, 20K for the classification network, and 20K in the localization network with $\alpha = 0.25$.

We denote our Segment-CNN using the above settings as S-CNN and compare with typical dense trajectory features (DTF) with bag-of-visual-words representation. The results of DTF is provided by [65] [1], which trains three SVM models with different set of negative samples and averages AP overall. According to Table 3.1, our Segment-CNN achieves tremendous performance gain for "BullChargeCape" action and competitive performance for "HorseRiding" action. Figure 3.5 displays our prediction results for "BullChargeCape" and "HorseRiding", respectively.

Results on THUMOS 2014 The instances in train set and validation set are used for training.

[1]Note that the results reported in [65] use different evaluation metrics. To make them comparable, we re-evaluate their prediction results according to standard criteria mentioned in Section 3.4.2.

AP(%)	BullChargeCape	HorseRiding	mAP
DTF	0.3	3.1	1.7
S-CNN	11.6	3.1	7.4

Table 3.1: Average precision on MEXaction2. The overlap threshold is set to 0.5 during evaluation.

The number of training iterations is 30K for all three networks. We again set $\alpha = 0.25$ for the localization network. We denote our Segment-CNN using the above settings as S-CNN.

θ	0.1	0.2	0.3	0.4	0.5
Karaman et al. [37]	1.5	0.9	0.5	0.3	0.2
Wang et al. [36]	19.2	17.8	14.6	12.1	8.5
Oneata et al. [35]	39.8	36.2	28.8	21.8	15.0
S-CNN	47.7	43.5	36.3	28.7	19.0

Table 3.2: Mean average precision on THUMOS 2014 as the overlap IoU threshold θ used in evaluation varies.

As for comparisons, beyond DTF, several baseline systems incorporate frame-level deep networks and even utilize lots of other features: (1) Karaman et al. [37] used FV encoding of iDT with weighted saliency based pooling, and conducted late fusion with frame-level CNN features. (2) Wang et al. [36] built a system on iDT with FV representation and frame-level CNN features, and performed post-processing to refine the detection results. (3) Oneata et al. [35] conducted localization using FV encoding of iDT on temporal sliding windows, and performed post-processing following [41]. Finally, they conducted weighted fusion for the localization scores of temporal windows and video-level scores generated by classifiers trained on iDT features, image features, and audio features. The results are listed in Table 3.2. AP for each class can be found in Figure 3.4. Our Segment-CNN significantly outperforms other systems for 14 of 20 actions, and the average performance improves from 15.0% to 19.0%. We also show two prediction results for the THUMOS 2014 test set in Figure 3.6.

Efficiency analysis. Our approach is very efficient when compared with all other systems, which typically fuse different features, and therefore can become quite cumbersome. Most segments generated from sliding windows are removed by the first proposal network, and thus the operations in classification and localization are greatly reduced. For each batch, the speed is around 1 second, and the number of segments can be processed during each batch depends on the GPU memory (approximately 25 for GeForce GTX 980 of 4G memory). The storage requirement is also extremely small because our method does not need to cache intermediate high dimensional features, such as FV to train SVM. All required by Segment-CNN is three deep network models, which occupy less than 1 GB in total.

3.4.3 Impact of Individual Networks

To study the effects of each network individually, we compare four Segment-CNNs using different settings: (1) **S-CNN**: keep all three networks and settings in Section 3.4.2, and Θ_{loc} is fine-tuned on Θ_{cls}; (2) **S-CNN (w/o proposal)**: remove the proposal network completely, and directly use Θ_{loc} to do predictions on sliding windows; (3) **S-CNN (w/o classification)**: remove the classification network completely and thus do not have Θ_{cls} to serve as initialization for training Θ_{loc}; (4) **S-CNN (w/o localization)**: remove the localization network completely and instead use classification

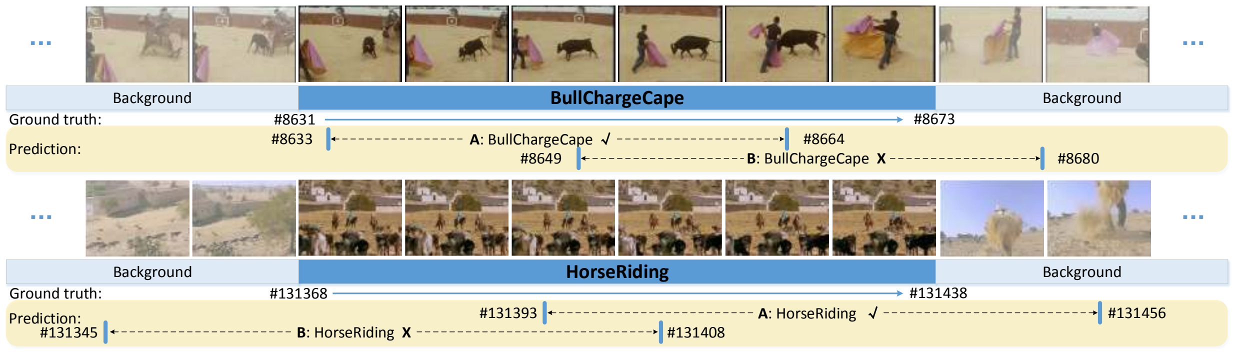


Figure 3.5: Prediction results for two action instances from MEXaction2 when the overlap threshold is set to 0.5 during evaluation. For each ground truth instance, we show two prediction results: A has the highest confidence score among the predictions associated with this ground truth, and B is an incorrect prediction. BullChargeCape: A is correct, but B is incorrect because each ground truth only allows one detection. HorseRiding: A is correct, but B is incorrect because each ground truth only allows one detection. The numbers shown with # are frame IDs.

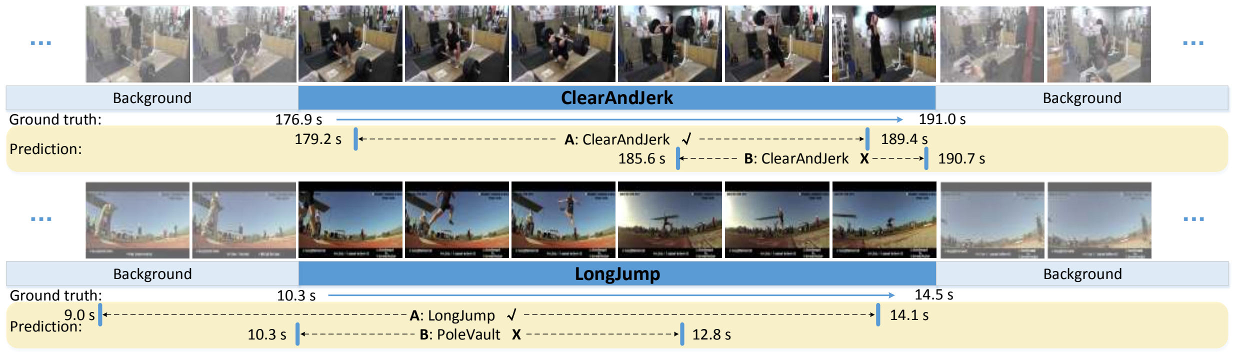


Figure 3.6: Prediction results for two action instances from THUMOS 2014 test set when the overlap threshold is set to 0.5 during evaluation. For each ground truth instance, we show two prediction results: A has the highest confidence score among the predictions associated with this ground truth, and B is an incorrect prediction. ClearAndJerk: A is correct, but B is incorrect because its overlap IoU with ground truth is less than threshold 0.5. LongJump: A is correct, but B is incorrect because it has the wrong action category prediction - PoleVault.

model Θ_{cls} to produce predictions.

The proposal network. We compare S-CNN (w/o proposal) and S-CNN, which includes the

networks	S-CNN (w/o proposal)	S-CNN
mAP(%)	17.1	19.0

Table 3.3: mAP comparisons on THUMOS 2014 between removing the proposal network and keeping the proposal network. The overlap threshold is set to 0.5 during evaluation.

proposal network as described above (two nodes in `fc8`). Because of the smaller network architecture than S-CNN (w/o proposal), S-CNN can reduce the number of operations conducted on background segments, and therefore accelerate speed. In addition, the results listed in Table 3.3 demonstrate that keeping the proposal network can also improve precision because it is designed for filtering out background segments that lack action of interests.

The classification network. Although Θ_{cls} is not used during prediction, the classification network is still important because fine-tuning on Θ_{cls} results in better performance. During evaluation here, we perform top-κ selection on the final prediction results to select κ segments with maximum confidence scores. As shown in Figure 3.7, S-CNN fine-tuned on Θ_{cls} outperforms S-CNN (w/o classification) consistently when κ varies, and consequently the classification network is necessary during training.

The localization network. Figure 3.7 also proves the effectiveness of the localization network. By adding the localization network, S-CNN can significantly improve performances compared with the baseline S-CNN (w/o localization), which only contains the proposal and classification networks. This is because the new loss function introduced in the localization network refines the

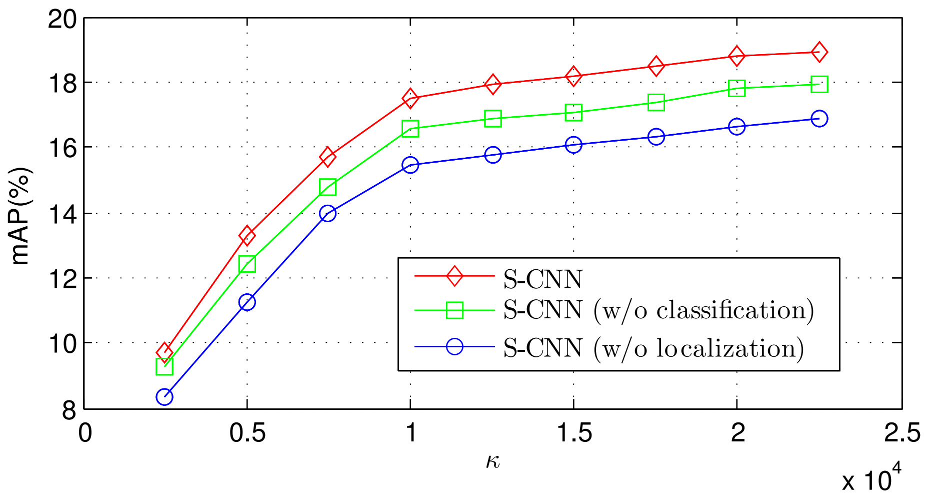


Figure 3.7: Effects of the classification and localization networks. y-axis is mAP(%) on THUMOS 2014, and x-axis varies the depth κ in top-κ selection. The overlap threshold is set to 0.5 during evaluation.

scores in favoring segments of higher overlap with the ground truths, and therefore higher temporal localization accuracy can be achieved.

In addition, we vary α in the overlap loss term $\mathcal{L}_{\text{overlap}}$ of the loss function to evaluate its sensitivity. We find that our approach has stable performances over a range of α value (e.g., from 0.25 to 1.0).

3.5 Summary

In this chapter, we address temporal action localization in untrimmed long videos. This is important because videos in real applications are usually unconstrained and contain multiple action instances plus video content of background scenes or other activities. To address this challenging issue, we have exploited the effectiveness of deep networks in temporal action localization via three segment-based 3D ConvNets: a proposal network, a classification network and a localization network. Through the above ablation study for each network, we have validated the following hypotheses: (1) instead of directly performing classification based on exhaustive scanning, the proposal network can filter out unlikely segments and improve the final detection accuracy; (2) it is beneficial to first train a classification network to provide good initialization for training the localization network; (3) the new loss function used in the localization network is key to precisely localizing action instances in time. Finally, when the overlap threshold used in evaluation is set to 0.5, our approach consisting of all three networks improves mAP on MEXaction2 from 1.7% to 7.4% and mAP on THUMOS 2014 from 15.0% to 19.0%.

Chapter 4

Precise Segment Boundary Detection and Frame-level Action Detection

4.1 Introduction

The **Temporal Action Localization (TAL)** task involves two components: (1) determining whether a video contains specific actions (such as diving, jump, *etc.*) and (2) identifying temporal boundaries (start time and end time) of each action instance. In Chapter 3, we have proposed an end-to-end deep learning framework called Segment-CNN (S-CNN) [67] based on 3D ConvNets [46] demonstrated superior performances both in efficiency and accuracy on standard benchmarks such as THUMOS'14 [38]. S-CNN consists of a proposal network for generating candidate video segments and a localization network for predicting segment-level scores of action classes. Although the localization network can be optimized to select segments with high overlaps with ground truth action instances, the detected action boundaries are still retained and thus are restricted to the pre-determined boundaries of a fixed set of proposal segments.

As illustrated in Figure 4.1, our goal is to refine temporal boundaries from proposal segments to precisely localize boundaries of action instances. This motivates us to move beyond existing practices based on segment-level predictions, and explicitly focus on the frame-level detection task of conducting fine-grained, dense predictions in time. To achieve this goal, some existing techniques can be adapted: (1) Single-frame classifiers operate on each frame individually; (2) Recurrent Neural Networks (RNN) further take into account temporal dependencies across frames. But both of them fail to explicitly model the spatio-temporal information in raw videos.

3D CNN [46; 67] has been shown that it can learn spatio-temporal abstraction of high-level semantics directly from raw videos but loses granularity in time, which is important for precise localization, as mentioned above. For example, layers from `conv1a` to `conv5b` in the well-known C3D architecture [46] reduce the temporal length of an input video by a factor of 8. In pixel-level semantic segmentation, de-convolution proves to be an effective upsampling method in both image [68; 69] and video [70] for producing output of the same resolution as the input. In our temporal localization problem, the temporal length of the output should be the same as the input video, but the spatial size should be reduced to 1x1. Therefore, we not only need to upsample in time but also need to downsample in space. To this end, we propose a novel **Convolutional-De-Convolutional** (CDC) filter, which performs convolution in space (for semantic abstraction) and

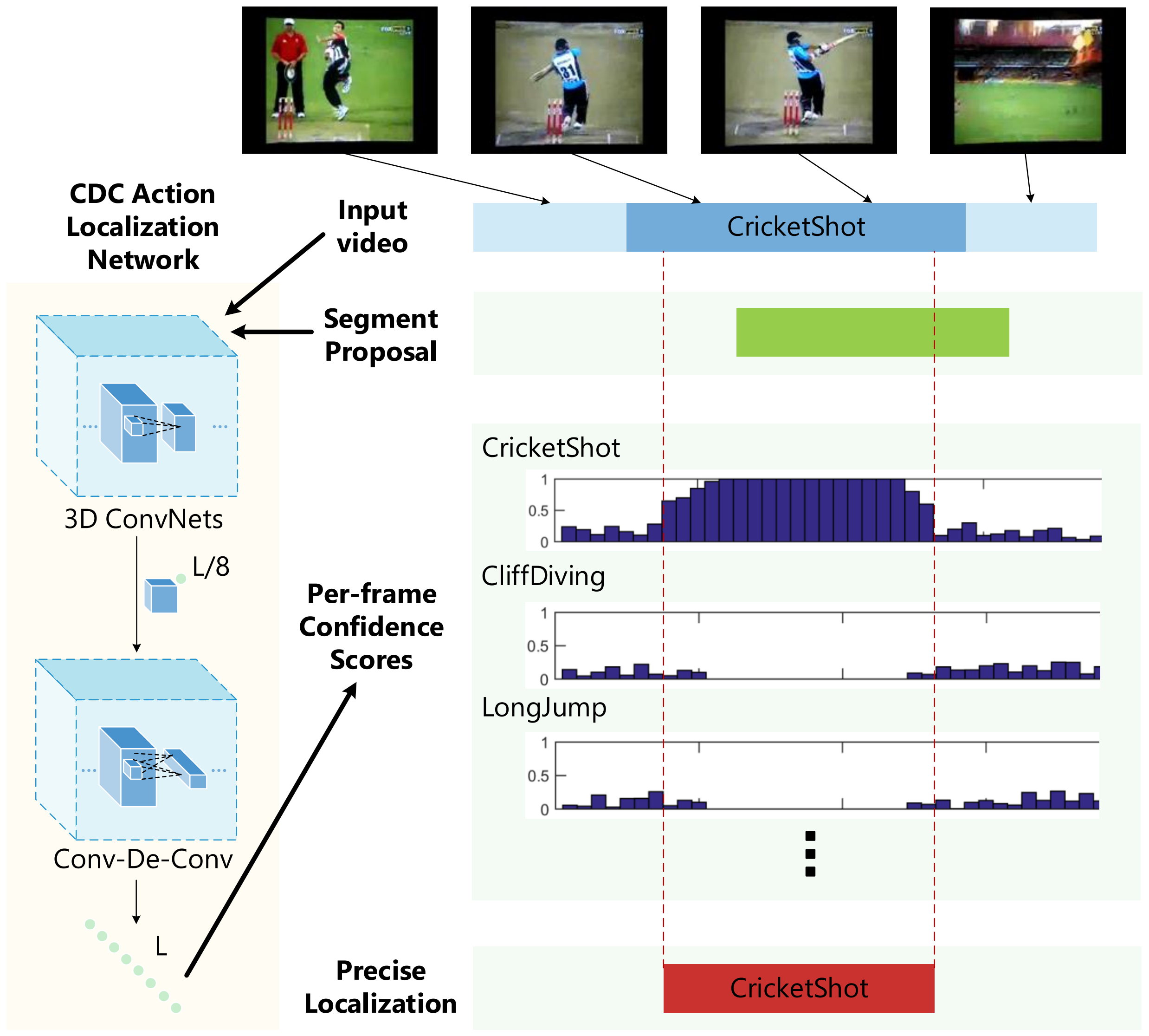


Figure 4.1: Our framework of precise segment boundary detection for temporal action localization. Given an input raw video, it is fed into our CDC localization network, which consists of 3D ConvNets for semantic abstraction and a novel CDC network for dense score prediction at the frame-level. Such fine-granular score sequences are combined with segment proposals to detect action instances with precise boundaries.

de-convolution in time (for frame-level resolution) simultaneously. It is unique in jointly modeling the spatio-temporal interactions between summarizing high-level semantics in space and inferring fine-grained action dynamics in time. On top of 3D ConvNets, we stack multiple CDC layers to form our CDC network, which can achieve the aforementioned goal of temporal upsampling and spatial downsampling, and thereby can determine action categories and can refine boundaries of proposal segments to precisely localize action instances.

In the rest of this chapter, we review the related work about de-convolution and semantic segmentation in Section 4.2, describe the proposed Convolutional-De-Convolutional networks in Section 4.3, present the experimental results in Section 4.4, and finally draw our summary in Section 4.5.

4.2 Related Work

Zeiler *et al.* [71] originally proposed de-convolutional networks for image decomposition, and later Zeiler and Fergus [72] re-purposed de-convolutional filter to map CNN activations back to the input to visualize where the activations come from. Long *et al.* [68; 69] showed that deep learning based approaches can significantly boost performance in image semantic segmentation. They proposed Fully Convolutional Networks (FCN) to output feature maps of reduced dimensions, and then employed de-convolution for upsampling to make dense, pixel-level predictions. The fully convolutional architecture and learnable upsampling method are efficient and effective, and thus inspired many extensions [73; 74; 75; 76; 77; 78; 79; 10; 80]. Note that this de-convolutional filter is effectively the transpose of convolution and thus is different from the conventional de-convoltion in signal processing [81].

Recently, Tran *et al.* [70] extended de-convolution from 2D to 3D and achieved competitive results on various voxel-level prediction tasks such as video semantic segmentation. This shows that de-convolution is also effective in the video domain and has the potential to be adapted for making dense predictions in time for our temporal action localization task. However, unlike the problem of semantic segmentation, we need to upsample in time but maintain downsampling in space. Instead of stacking a convolutional layer and a de-convolutional layer to conduct upsampling and downsampling separately, our proposed CDC filter learns a joint model to perform these two operations simultaneously, and proves to be more powerful and easier to train.

4.3 Convolutional-De-Convolutional networks

4.3.1 The need of downsampling and upsampling

C3D architecture, consisting of 3D ConvNets followed by three Fully Connected (FC) layers, has achieved promising results in video analysis tasks such as recognition [46] and localization [67]. Further, Tran *et al.* [70] experimentally demonstrated the 3D ConvNets, *ie.* from `conv1a` to `conv5b`, to be effective in summarizing spatio-temporal patterns from raw videos into high-level semantics.

Therefore, we build our CDC network upon C3D. We adopt from `conv1a` to `conv5b` as the first part of our CDC network. For the rest of layers in C3D, we keep `pool5` to perform max pooling in height and width by a factor of 2 but retain the temporal length. Following conventional settings

[46; 67; 70], we set the height and width of the CDC network input to 112x112. Given an input video segment of temporal length L, the output data shape of `pool5` is (512, $L/8$, 4, 4) [1]. Now in order to predict the action class scores at the original temporal resolution (frame-level), we need to upsample in time (from $L/8$ back to L), and downsample in space (from 4x4 to 1x1). To this end, we propose the CDC filter and design a CDC network to adapt the FC layers from C3D to perform the required upsample and downsample operations. Details are described in Sections 4.3.2 and 4.3.3.

4.3.2 CDC filter

In this section, we walk through a concrete example of adapting `FC6` layer in C3D to perform spatial downsampling by a factor of 4x4 and temporal upsampling by a factor of 2. For the sake of clarity, we focus on how a filter operates within one input channel and one output channel.

As explained in [68; 69], the FC layer is a special case of a convolutional layer (when the input data and the kernel have the same size and there is no striding and no padding). So we can transform `FC6` into `conv6`, which is shown in Figure 4.2 (a). Previously, a filter in `FC6` takes a 4x4 feature map from `pool5` as input and outputs a single value. Now, a filter in `conv6` can slide on $L/8$ feature maps of size 4x4 stacked in time and respectively output $L/8$ values in time. The kernel size of `conv6` is 4x4=16.

Although `conv6` performs spatial downsampling, the temporal length remains unchanged. To

[1]We denote the shape of data in the networks using the form of (number of channels, temporal length, height, width) and the size of feature map, kernel, stride, zero padding using (temporal length, height, width).

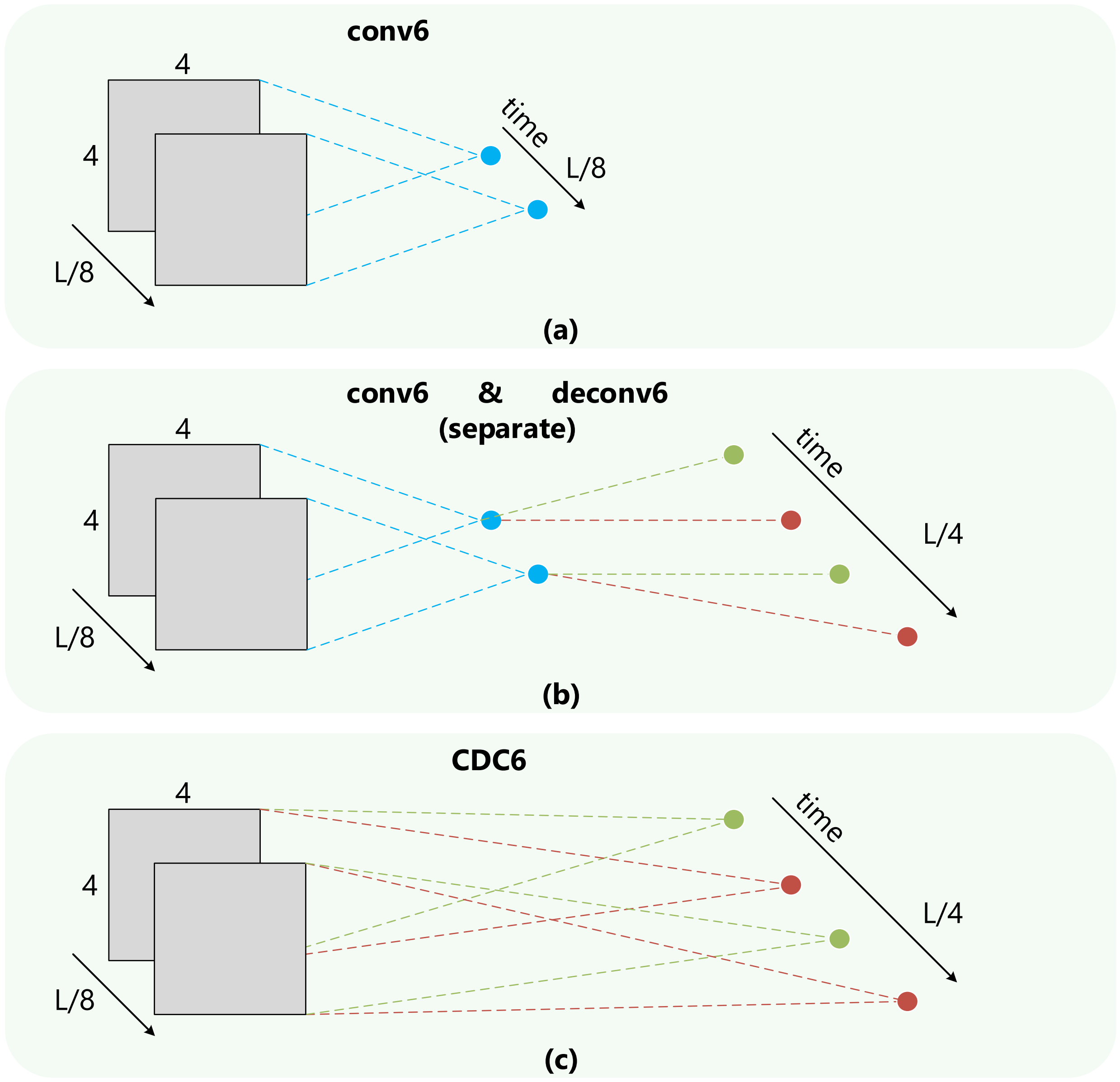


Figure 4.2: Illustration of how a filter in `conv6`, `deconv6`, `CDC6` operates on `pool5` output feature maps (grey rectangles) stacked in time. In each panel, dashed lines with the same color indicate the same filter sliding over time. Nodes stand for outputs.

upsample in time, as shown in Figure 4.2 (b), a straightforward solution adds a de-convolutional layer `deconv6` after `conv6` to double the temporal length while maintaining the spatial size. The kernel size of `deconv6` is 2. Therefore, the total number of parameters for this solution (separated `conv6` and `deconv6`) is 4x4+2=18.

However, this solution conducts temporal upsampling and spatial downsampling in a separate manner. Instead, we propose the CDC filter `CDC6` to jointly perform these two operations. As illustrated in Figure 4.2 (c), a `CDC6` filter consists of two independent convolutional filters (the red one and the green one) operating on the same input 4x4 feature map. Each of these convolutional filters has the same kernel size as the filter in `conv6` and separately outputs one single value. So each 4x4 feature map results in 2 outputs in time. As the CDC filter slides on $L/8$ feature maps of size 4x4 stacked in time, this input feature volume of temporal length $L/8$ is upsampled in time to $L/4$, and its spatial size is reduced to 1x1. Consequently, in space this CDC filter is equivalent to a 2D convolutional filter of kernel size 4x4; in time it has the same effect as a 1D de-convolutional filter of kernel size 2, stride 2, padding 0. The kernel size of such a joint filter in `CDC6` is 2x4x4=32, which is larger than the separate convolution and de-convolution solution (18).

Therefore, a CDC filter is more powerful for jointly modeling high-level semantics and temporal dynamics: each output in time comes from an independent convolutional kernel dedicated to this output (the red/green node corresponds to the red/green kernel); however, in the separate convolution and de-convolution solution, different outputs in time share the same high-level semantics (the blue node) outputted by one single convolutional kernel (the blue one).

Having more parameters makes the CDC filter harder to learn. To remedy this issue, we propose a method to adapt the pre-trained `FC6` layer in C3D to initialize `CDC6`. After we convert `FC6` to `conv6`, `conv6` and `CDC6` have the same number of channels (*i.e.* 4,096) and thus the same number of filters. Each filter in `conv6` can be used to initialize its corresponding filter in `CDC6`: the filter in `conv6` (the blue one) has the same kernel size as each of these two convolutional filters (the red one and the green one) in the `CDC6` filter and thus can serve as the initialization for them both.

Generally, assume that a CDC filter F of kernel size (k_l, k_h, k_w) takes the input receptive field X of height k_h and width k_w, and produces Y that consists of k_l successive outputs in time. For the example given in Figure 4.2 (c), we have $k_l = 2$, $k_h = 4$, $k_w = 4$. Given the indices $a \in \{1, ..., k_h\}$ and $b \in \{1, ..., k_w\}$ in height and width respectively for X and the index $c \in \{1, ..., k_l\}$ in time for Y: during the forward pass, we can compute Y by

$$Y\left[c\right] = \sum_{a=1}^{k_h} \sum_{b=1}^{k_w} F\left[c, a, b\right] \cdot X\left[a, b\right]; \tag{4.1}$$

during the back-propagation, our CDC filter follows the chain rule and propagates gradients from Y to X via

$$X\left[a, b\right] = \sum_{c=1}^{k_l} F\left[c, a, b\right] \cdot Y\left[c\right]. \tag{4.2}$$

A CDC filter F can be regarded as coupling a series of convolutional filters (each one has kernel size k_h in height and k_w in width) in time with a shared input receptive field X, and at the same time, F performs 1D de-convolution with kernel size k_l in time. In addition, the cross-channel mechanisms within a CDC layer and the way of adding biases to the outputs of the CDC filters follow the conventional strategies used in convolutional and de-convolutional layers.

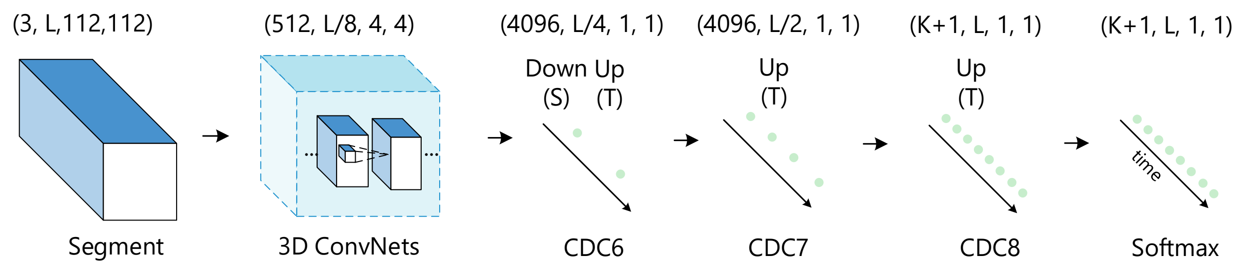


Figure 4.3: Architecture of a typical CDC network. Following the notations indicated in the footnote 1, the top row lists the shape of output data at each layer. (1) A video segment is first fed into 3D ConvNets and the temporal length reduces from L to $L/8$. (2) `CDC6` has kernel size (4, 4, 4), stride (2, 1, 1), padding (1, 0, 0), and therefore reduces both height and width to 1 while increases the temporal length from $L/8$ to $L/4$. Both `CDC7` and `CDC8` have kernel size (4, 1, 1), stride (2, 1, 1), padding (1, 0, 0), and hence both `CDC7` and `CDC8` further perform upsampling in time by a factor of 2, and thus the temporal length is back to L. (3) A frame-wise softmax layer is added on top of `CDC8` to obtain confidence scores for every frame. Each channel stands for one class.

4.3.3 Design of CDC network architecture

In Figure 4.3, we illustrate our CDC network for labeling every frame of a video. The final output shape of the CDC network is (K+1, L, 1, 1), where K+1 stands for K action categories plus the background class. As described in Section 4.3.1, from `conv1a` to `pool5`, the temporal length of an input segment has been reduced from L to $L/8$. On top of `pool5`, in order to make per-frame predictions, we adapt FC layers in C3D as CDC layers to perform temporal upsampling and spatial downsampling operations. Following previous de-convolution works [70; 68; 69], we upsample in time by a factor of 2 in each CDC layer, to gradually increase temporal length from $L/8$ back to L.

In the previous Section 4.3.2, we provide an example of how to adapt `FC6` as `CDC6`, performing temporal 1D de-convolution of kernel size 2, stride 2, padding 0. For `CDC6` in the CDC network, we construct a CDC filter with 4 convolutional filters instead of 2, and thus its temporal kernel size in time increases from 2 to 4. We set the corresponding stride to 2 and padding to 1. Now each 4x4 feature map produces 4 output nodes, and every two consecutive feature maps have 2 nodes overlapping in time. Consequently, the temporal length of input is still upsampled by `CDC6` from $L/8$ to $L/4$, but each output node sums contributions from two consecutive input feature maps, allowing temporal dynamics in input to be taken into account.

Likewise, we can adapt `FC7` as `CDC7`, as indicated in Figure 4.3. Additionally, we retain the Relu layers and the Dropout layers with 0.5 dropout ratio from C3D to attach to both `CDC6` and `CDC7`. `CDC8` corresponds to `FC8` but cannot be directly adapted from `FC8` because the classes in `FC8` and `CDC8` are different. Since each channel stands for one class, `CDC8` has K+1 channels. Finally, the `CDC8` output is fed into a frame-wise softmax layer `Softmax` to produce per-frame scores. During each mini-batch with N training segments, for the n-th segment, the `CDC8` output O_n has the shape (K+1, L, 1 ,1). For each frame, performing the conventional softmax operation and computing the softmax loss and gradient are independent of other frames. Corresponding to the t-th frame, the `CDC8` output $O_n\left[t\right]$ and `Softmax` output $P_n\left[t\right]$ both are vectors of K+1 values. Note that for the i-th class, $P_n^{(i)}\left[t\right] = \frac{e^{O_n^{(i)}[t]}}{\sum_{j=1}^{K+1} e^{O_n^{(j)}[t]}}$. The total loss $\mathcal{L}$ is defined as:

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{L} \left(-\log\left(P_n^{(z_n)}\left[t\right] \right) \right), \tag{4.3}$$

where z_n stands for the ground truth class label for the n-th segment. The total gradient w.r.t the output of i-th channel/class and t-th frame in `CDC8` is the summation over all N training segments of:

$$\frac{\partial \mathcal{L}}{\partial O_n^{(i)}[t]} = \begin{cases} \frac{1}{N} \cdot \left(P_n^{(z_n)}[t] - 1 \right) & \text{if } i = z_n \\ \frac{1}{N} \cdot P_n^{(i)}[t] & \text{if } i \neq z_n \end{cases} . \tag{4.4}$$

4.3.4 Training and prediction

Training data construction. In theory, because both the convolutional filter and the CDC filter slide over the input, they can be applied to input of arbitrary size. Therefore, our CDC network can operate on videos of variable lengths. Due to GPU memory limitations, in practice we slide a temporal window of 32 frames without overlap on the video and feed each window individually into the CDC network to obtain dense predictions in time. From the temporal boundary annotations, we know the label of every frame. Frames in the same window can have different labels. To prevent including too many background frames for training, we only keep windows that have at least one frame belonging to actions. Therefore, given a set of training videos, we obtain a training collection of windows with frame-level labels.

Optimization. We use stochastic gradient descent to train the CDC network with the aforementioned frame-wise softmax loss. Our implementation is based on Caffe [66] and C3D [46]. The learning rate is set to 0.00001 for all layers except for `CDC8` layer where the learning rate is 0.0001 since `CDC8` is randomly initialized. Following conventional settings [46; 67], we set momentum to 0.9 and weight decay to 0.005.

C3D [46] is trained on Sports-1M [14] and can be used to directly initialize `conv1a` to `conv5b`.

`CDC6` and `CDC7` are initialized by `FC6` and `FC7` respectively using the strategy described in the Section 4.3.2. In addition, since `FC8` in C3D and `CDC8` in the CDC network have the different number of channels, we randomly initialize `CDC8`. With such initialization, our CDC network turns out to be very easy to train and converges quickly, *i.e.* 4 training epochs (within half a day) on THUMOS'14 .

Fine-grained prediction and precise localization. During testing, after applying the CDC network on the whole video, we can make predictions for every frame of the video. Through thresholding on confidence scores and grouping adjacent frames of the same label, it is possible to cut the video into segments and produce localization results. But this method is not robust to noise, and designing temporal smoothing strategies turns out to be ad hoc and non-trivial. Recently, researchers developed some efficient segment proposal methods [67; 59] to generate a small set of candidate segments of high recall. Utilizing these proposals for our localization model not only bypasses the challenge of grouping adjacent frames, but also achieves considerable speedup during testing, because we only need to apply the CDC network on the proposal segments instead of the whole video.

Since these proposal segments only have coarse boundaries, we propose using fine-grained predictions from the CDC network to localize precise boundaries. First, to look at a wider interval, we extend each proposal segment's boundaries on both sides by the percentage α of the original segment length. We set α to 1/8 for all experiments. Then, similar to preparing training segments, we slide temporal windows without overlap on the test videos. We only need to keep test windows that overlap with at least one extended proposal segment. We feed these windows into our CDC network and generate per-frame action classes scores.

The category of each proposal segment is set to the class with the maximum average confidence score over all frames in the segment. If a proposal segment does not belong to the background class, we keep it and further refine its boundaries. Given the score sequence of the predicted class in the segment, we perform Gaussian kernel density estimation and obtain its mean μ and standard deviation σ. Starting from the boundary frame at each side of the extended segment and moving towards its middle, we shrink its temporal boundaries until we reach a frame with the confidence score no lower than μ - σ. Finally, we set the prediction score of the segment to the average confidence score of the predicted class over frames in the refined segment of boundaries.

4.4 Experiments

4.4.1 Per-frame labeling

We first demonstrate the effectiveness of our model in predicting accurate labels for every frame. Note that this task can accept an input of multiple frames to take into account temporal information. We denote our model as **CDC**.

THUMOS'14 [38]. The temporal action localization task in THUMOS Challenge 2014 involves

methods	mAP
Single-frame CNN [53]	34.7
Two-stream CNN [34]	36.2
LSTM [18]	39.3
MultiLSTM [61]	41.3
C3D + LinearInterp	37.0
Conv & De-conv	41.7
CDC (fix 3D ConvNets)	37.4
CDC	**44.4**

Table 4.1: Per-frame labeling mAP on THUMOS'14 .

20 actions. We use 2,755 trimmed training videos and 1,010 untrimmed validation videos (3,007 action instances) to train our model. For testing, we use all 213 test videos (3,358 action instances) which are not entirely background videos.

Evaluation metrics. Following conventional metrics [61], we treat the per-frame labeling task as a retrieval problem. For each action class, we rank all frames in the test set by their confidence scores for that class and compute Average Precision (AP). Then we average over all classes to obtain mean AP (mAP).

Comparisons. In Table 4.1, we first compare our CDC network (denoted by CDC) with some state-of-the-art models (results are quoted from [61]): (1) Single-frame CNN: the frame-level 16-layer VGG CNN model [53]; (2) Two-stream CNN: the frame-level two-stream CNN model proposed in [34], which has one stream for pixel and one stream for optical flow; (3) LSTM: the basic per-frame labeling LSTM model of 512 hidden units [18] on the top of VGG CNN `FC7` layer;

(4) MultiLSTM: a LSTM model developed by Yeung *et al.* [61] to process multiple input frames together with temporal attention mechanism and output predictions for multiple frames. Single-frame CNN only takes into account appearance information. Two-stream CNN models appearance and motion information separately. LSTM based models can capture temporal dependencies across frames but do not model motion explicitly. Our **CDC** model is based on 3D convolutional layers and CDC layers, which can operate on spatial and temporal dimensions simultaneously, achieving the best performance.

In addition, we compare CDC with other C3D based approaches that use different upsampling methods. (1) C3D + LinearInterp: we train a segment-level C3D using the same set of training segments whose segment-level labels are determined by the majority vote. During testing we perform linear interpolation to upsample segment-level predictions as frame-level. (2) Conv & De-conv: `CDC7` and `CDC8` in our CDC network keep the spatial data shape unchanged and therefore can be also regarded as de-convolutional layers. For `CDC6`, we replace it with a convolutional layer `conv6` and a separate de-convolutional layer `deconv6` as shown in Figure 4.2 (b). The CDC model outperforms these baselines because the CDC filter can simultaneously model high-level semantics and temporal action dynamics. We also evaluate the CDC network with fixed weights in 3D ConvNets and only fine-tune CDC layers, resulting in a minor performance drop. This implies that it is helpful to train CDC networks in an end-to-end manner so that the 3D ConvNets part can be trained to summarize more discriminative information for CDC layers to infer more accurate temporal dynamics.

IoU threshold	0.3	0.4	0.5	0.6	0.7
Karaman *et al.* [37]	0.5	0.3	0.2	0.2	0.1
Wang *et al.* [36]	14.6	12.1	8.5	4.7	1.5
Heilbron *et al.* [52]	-	-	13.5	-	-
Escorcia *et al.* [59]	-	-	13.9	-	-
Oneata *et al.* [35]	28.8	21.8	15.0	8.5	3.2
Richard and Gall [51]	30.0	23.2	15.2	-	-
Yeung *et al.* [60]	-	-	17.1	-	-
Yuan *et al.* [62]	33.6	26.1	18.8	-	-
S-CNN [67]	36.3	28.7	19.0	10.3	5.3
C3D + LinearInterp	36.0	26.4	19.6	11.1	6.6
Conv & De-conv	38.6	28.2	22.4	12.0	7.5
CDC (fix 3D ConvNets)	36.9	26.2	20.4	11.3	6.8
CDC	**40.1**	**29.4**	**23.3**	**13.1**	**7.9**

Table 4.2: Temporal action localization mAP on THUMOS'14 as the overlap IoU threshold used in evaluation varies from 0.3 to 0.7. - indicates that results are unavailable in the corresponding papers.

4.4.2 Temporal action localization

Given per-frame labeling results from the CDC network, we generate proposals, determine class category, and predict precise boundaries following Section 4.3.4. Our approach is applicable to any segment proposal method. Here we conduct experiments on THUMOS'14, and thus employ the publicly available proposals generated by the S-CNN proposal network [67], which achieves high recall on THUMOS'14 . Finally, we follow [61; 67] to perform standard post-processing steps such as non-maximum suppression.

Evaluation metrics. Localization performance is also evaluated by mAP. Each item in the rank list is a predicted segment. The prediction is correct when it has the correct category and its temporal overlap IoU with the ground truth is larger than the threshold. Redundant detections for the same ground truth instance are not allowed.

Comparisons. As shown in Table 4.2, **CDC** achieves much better results than all the other state-of-the-art methods. Compared to the proposed CDC model: the typical approach of extracting a set of features to train SVM classifiers and then applying the trained classifiers on sliding windows or segment proposals (Karaman *et al.* [37], Wang *et al.* [36], Oneata *et al.* [35], Escorcia *et al.* [59]) does not directly address the temporal localization problem. Systems encoding iDTF with FV (Heilbron *et al.* [52], Richard and Gall [51]) cannot learn spatio-temporal patterns directly from raw videos to make predictions. RNN/LSTM based methods (Yeung *et al.* [60], Yuan *et al.* [62]) are unable to explicitly capture motion information beyond temporal dependencies. S-CNN can effectively capture spatio-temporal patterns from raw videos but lacks the ability of adjusting boundaries from proposal candidates. With the proposed CDC filter, the CDC network can determine confidence scores at a fine granularity, beyond segment-level prediction, and hence precisely localize temporal boundaries. In addition, we employ per-frame predictions of other methods indicated in Table 4.1 (C3D + LinearInterp, Conv & De-conv, CDC with fixed 3D ConvNets) to perform temporal localization based on S-CNN proposal segments. As shown in Table 4.2, the performance of the CDC network is still better, because more accurate predictions at the same temporal granularity can be used to predict more accurate label and more precise boundaries for the same input proposal segment. In Figure 4.4, we illustrate how our model refines boundaries from segment proposal to precisely localize action instance in time.

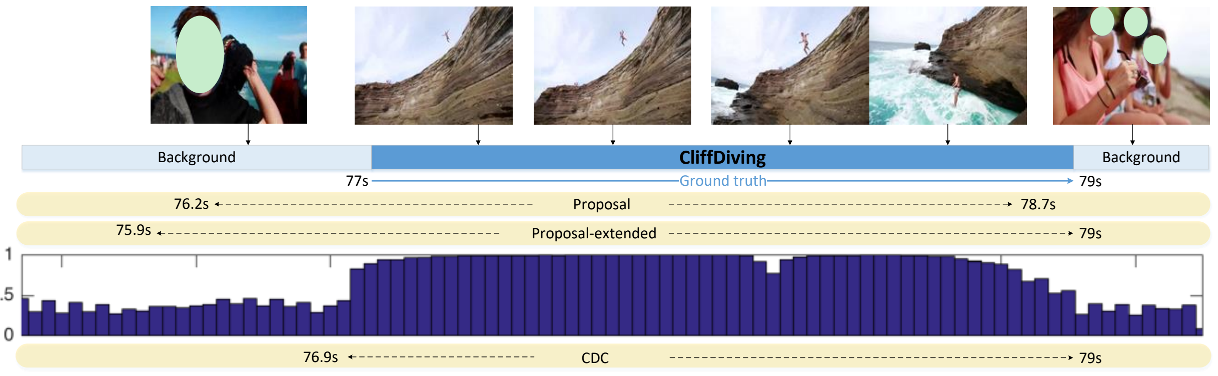


Figure 4.4: Visualization of the process of refining temporal boundaries for a proposal segment. Horizontal axis stands for time. From the top to the bottom: (1) frame-level ground truths for a CliffDiving instance in an input video with some representative frames; (2) a corresponding proposal segment; (3) the proposal segment after extension; (4) the per-frame score of detecting CliffDiving predicted by the CDC network; (5) the predicted action instance after the refinement using CDC.

4.4.3 Discussions

The necessity of predicting at a fine granularity in time. In Figure 4.5, we compare CDC networks predicting action scores at different temporal granularities. When the temporal granularity increases, mAP increases accordingly. This demonstrates the importance of predicting at a fine-granularity for achieving precise localization.

Efficiency analysis. The CDC network is compact and demands little storage, because it can be trained from raw videos directly to make fine-grained predictions in an end-to-end manner without

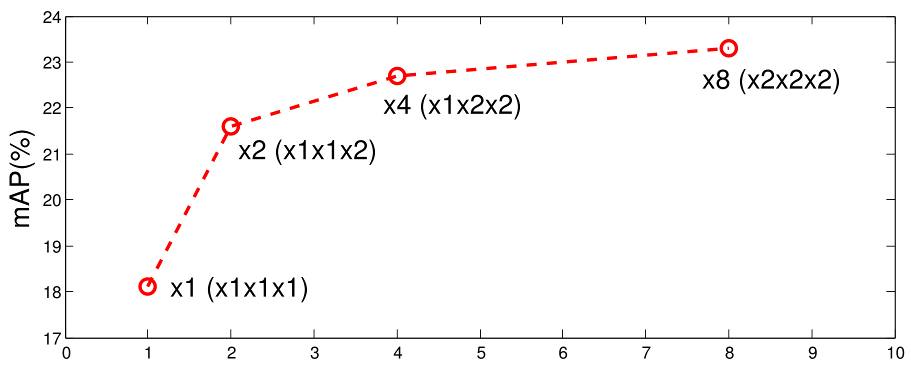


Figure 4.5: mAP gradually increases when the temporal granularity of CDC network prediction increases from x1 (one label for every 8 frames) to x8 (one label per frame). Each point corresponds to **x total upscaling factor (x** `CDC6` **upscaling factor x** `CDC7` **upscaling factor x** `CDC8` **upscaling factor)** in time. We conduct the evaluation on THUMOS'14 with IoU 0.5.

the need to cache intermediate features. A typical CDC network such as the example in Figure 4.3 only requires around 1GB storage.

Our approach is also fast. Compared with segment-level prediction methods such as S-CNN localization network [67], CDC has to perform more operations due to the need of making predictions at every frame. Therefore, when the proposal segment is long, CDC is less efficient for the sake of achieving more accurate boundaries. But in the case of short proposal segments, since these proposals usually are densely overlapped, segment-level methods have to process a large number of segments one by one. However, CDC networks only need to process each frame once, and thus it can avoid redundant computations. On a NVIDIA Titan X GPU of 12GB memory, the speed of a CDC network is around 500 Frames Per Second (FPS), which means it can process a 20s long video clip of 25 FPS within one second.

mAP	0.5	0.75	0.95	Average-mAP
before	45.1	4.1	0.0	16.4
after	45.3	26.0	0.2	23.8

Table 4.3: Temporal localization mAP on ActivityNet Challenge 2016 before and after the refinement step using our CDC network. We follow the official metrics to evaluate the average mAP.

Temporal activity localization. Furthermore, we found that our approach is also useful for localizing activities of high-level semantics and complex components. We conduct experiments on ActivityNet Challenge 2016 dataset [82; 83], which involves 200 activities, and contains around 10K training videos (15K instances) and 5K validation videos (7.6K instances). Each video has an average of 1.65 instances with temporal annotations. We train on the training videos and test on the validation videos. Since no activity proposal results of high quality exist, we apply the trained CDC network to the results of the first place winner [84] in this Challenge to localize more precise boundaries. As shown in Table 4.3, they achieve high mAP when the IoU in evaluation is set to 0.5, but mAP drops rapidly when the evaluation IoU increases. After using the per-frame predictions of our CDC network to refine temporal boundaries of their predicted segments, we gain significant improvements particularly when the evaluation IoU is high (*i.e* 0.75). This means that after the refinement, these segments have more precise boundaries and have larger overlap with ground truth instances.

4.5 Summary

Temporal action localization is an important yet challenging problem. Given a long, untrimmed video consisting of multiple action instances and complex background contents, we need not only to recognize their action categories, but also to localize the start time and end time of each instance. Many state-of-the-art systems use segment-level classifiers to select and rank proposal segments of pre-determined boundaries. However, we propose that a desirable model should move beyond segment-level and make dense predictions at a fine granularity in time to determine precise temporal boundaries. In order to detect at a fine granularity in time, we hypothesize that jointly modeling action semantics in space-time and fine-grained temporal dynamics can more accurately predict actions at frame-level. Thus we design a novel **Convolutional-De-Convolutional** (CDC) filter to perform the required temporal upsampling and spatial downsampling operations simultaneously. Our method not only improves the state-of-the-art mean Average Precision (mAP) result on THUMOS'14 from 41.3% to 44.4% for the per-frame labeling task, but also improves the state-of-the-art mAP for the temporal action localization task from 19.0% to 23.3% on THUMOS'14 and from 16.4% to 23.8% on ActivityNet v1.3. These results not only confirms the need of joint modeling in both space and time but also confirms the effectiveness of detecting at a fine granularity in time to determine precise temporal boundaries.

Part II

Action Detection in the Constrained Scenarios

Chapter 5

Weakly-supervised Action Detection

5.1 Introduction

Methods developed in Part I can be used to address the **Temporal Action Localization (TAL)** task in untrimmed videos. But these methods are proposed for the **fully-supervised** setting: the model training requires the full annotation of the ground truth temporal boundary (start time and end time) for each action instance. However, untrimmed videos are usually very long with substantial background content in time. Therefore, manually annotating temporal boundaries for a new large-scale dataset is very expensive and time-consuming [85], and thus might prohibit applying the fully-supervised methods to the new domains that lack enough training data with full annotations.

This motivates us to develop TAL methods that require significantly fewer ground truth annotations for training. As illustrated in Fig. 5.1, in this chapter we focus on the following scenario:

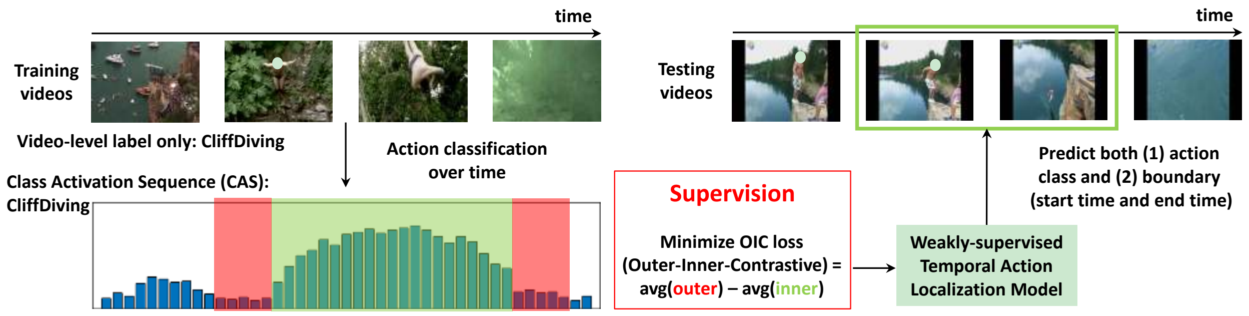

time
Training videos
Video-level label only: CliffDiving
Action classification over time
Class Activation Sequence (CAS): CliffDiving
Supervision
Minimize OIC loss (Outer-Inner-Contrastive) = avg(outer) – avg(inner)
Weakly-supervised Temporal Action Localization Model
time
Testing videos
Predict both (1) action class and (2) boundary (start time and end time)

Figure 5.1: We study the weakly-supervised temporal action localization task: during training we only have videos with the video-level labels, but during testing we need to predict both (1) the action class and (2) the temporal boundary of each action instance. In order to obtain the segment-level supervision for training the action localization model to predict the boundary directly, we design a novel **Outer-Inner-Contrastive (OIC)** loss based on the action Class Activation Sequence. We denote the predicted action segment boundary as the *inner boundary*. The *outer boundary* is obtained by extending the inner boundary to include its surrounding area. A desirable boundary prediction should have high activations in the inner **green** area but low activations in the outer **red** area. Consequently, the OIC loss can be used to approximately determine the needed segment-level supervision for training the localization model

during training, we only have the video-level labels, which are much easier to collect , compared to the boundary annotations; during testing, we still aim to predict both (1) the action class and (2) the temporal boundary (i.e. start time and end time) of each action instance. We refer this scenario as the **weakly-supervised** setting that this chapter works on.

Recently, a few methods have been proposed to tackle TAL in such a weakly-supervised setting. UntrimmedNet [86] and Hide-and-Seek [87] achieve the state-of-the-art performances and carry out the localization in a similar manner. Given a training video, several segments are randomly sampled and are fed into a network together to yield a video-level class prediction. During testing, the trained network is slided over time to produce the classification score sequence of being each action over time. The score sequence is similar to the Class Activation Map in [88] but just has one dimension, and thus we refer it as **Class Activation Sequence (CAS)**. Finally a simple thresholding method is applied on the CAS to localize each action instance in terms of the start time and the end time.

However, performing localization via thresholding in general may not be robust to noises in CAS: sometimes there are a few dips of low activations within an interval of high activations, using a large threshold might over-segment one action instance into several segments; but using a small threshold might include too many irrelevant backgrounds preceding and succeeding the action instance. One possible solution is improving the quality of CAS. Alternatively, instead of thresholding, many fully-supervised TAL methods detect action instances at the segment-level directly [67; 89]. Some works further employ boundary regression models to learn to predict more accurate boundaries [90; 91; 92; 93]. Thus, we design a framework called **AutoLoc** which can conduct direct boundary prediction via predicting the center location and the duration of each action instance.

But how to train the boundary prediction model without ground truth boundary annotations still remains unsolved. To address this challenge, we propose a novel **Outer-Inner-Contrastive (OIC)** loss to provide the needed segment-level supervision for training the boundary prediction model. Given the CAS of being the ground truth action, we denote the *inner boundary* as the boundary of a predicted action instance and we inflate the inner boundary slightly to obtain the *outer boundary*. As illustrated in Fig. 5.1, we propose an OIC loss as the average activation in the outer **red** area minus the average activation in the inner **green** area. By minimizing the OIC loss to find the area of high inner activations but low outer activations, we can make desirable localization of the salient interval on CAS, which is likely to be well-aligned with the ground truth segment. Equipped with the OIC loss, AutoLoc can automatically discover the segment-level supervision from the video-level annotations for training the boundary prediction model. In Sec. 5.5, we will experimentally compare with the state-of-the-art methods and also study several variants of our model.

In the rest of this chapter, we review the related work in Section 5.2, describe the network architecture and the training and testing pipelines of the proposed Segment-CNN framework in Section 5.3, present the experimental results in Section 5.5, and finally draw our summary in Section 5.6.

5.2 Related Work

5.2.1 Temporal Action Localization with Weak Supervision

Several large-scale video datasets have been created for TAL such as Charades [94; 95], ActivityNet [82], THUMOS [38; 40]. In order to obtain the ground truth temporal boundaries to provide full supervision for training the fully-supervised TAL models, substantial efforts are required for annotating each of such large-scale datasets. Therefore, it is useful and important to develop TAL models that can be trained with weak supervision only.

Video-level annotation is one kind of weak supervision that can be more easily collected and thus is quite interesting to the community. Sun *et al.* [49] was the first to consider TAL with only the video-level annotations available during training and the authors discovered the additional supervision from web images. Recently, Singh *et al.* designed Hide-and-Seek [87] to address the challenge that weakly-supervised detection methods usually focus on the most discriminative parts while neglect other relevant parts of the target instance. Wang *et al.* [86] proposed a framework called UntrimmedNet consisting of a classification module to perform action classification and a selection module to detect important temporal segments. These recent methods are effectively learning an action classification model during training in order to generate reasonably good Class Activation Sequence (CAS) over time. But in order to detect temporal boundaries, a simple thresholding is applied on the CAS during testing. Therefore, although these methods can excel at the video-level action recognition, the performance of temporal localization still has large room for improvement.

However, the fully-supervised TAL methods (boundary annotations available during training) have gone beyond the simple thresholding method. First, some researchers performed localization at segment-level: they first generated the candidate segments via sliding window or proposal methods, and then they classified each segment into certain actions [67; 91; 92; 93; 96]. Motivated by the success of single-shot object detection method [97; 98; 99], Lin *et al.* [90] removed the proposal stage and directly conducted TAL in a single-shot fashion to simultaneously predict temporal boundary and action class. Second, direct boundary prediction via anchor generation and boundary regression has been adapted from object detection [97; 98; 99; 9; 43] to fully-supervised TAL recently and proven to be quite effective in detecting more accurate boundaries [90; 100; 91; 92; 93]. This motivates us to generalize segment-level localization and direct boundary prediction to weakly-supervised TAL: we develop AutoLoc to first generate anchor segments and then regress their boundaries to obtain the predicted segments; in order to train the boundary regressors, we propose the OIC loss to provide the segment-level supervision.

5.2.2 Weakly-supervised Deep Learning Methods

Other types of weak supervision for action detection have also been explored in the past. For instance, Huang *et al.* [101] and Richard *et al.* [102] both utilized the order of actions as the supervision used during training. Mettes *et al.* [103] worked on the spatio-temporal action detection using only the point-level supervision for training.

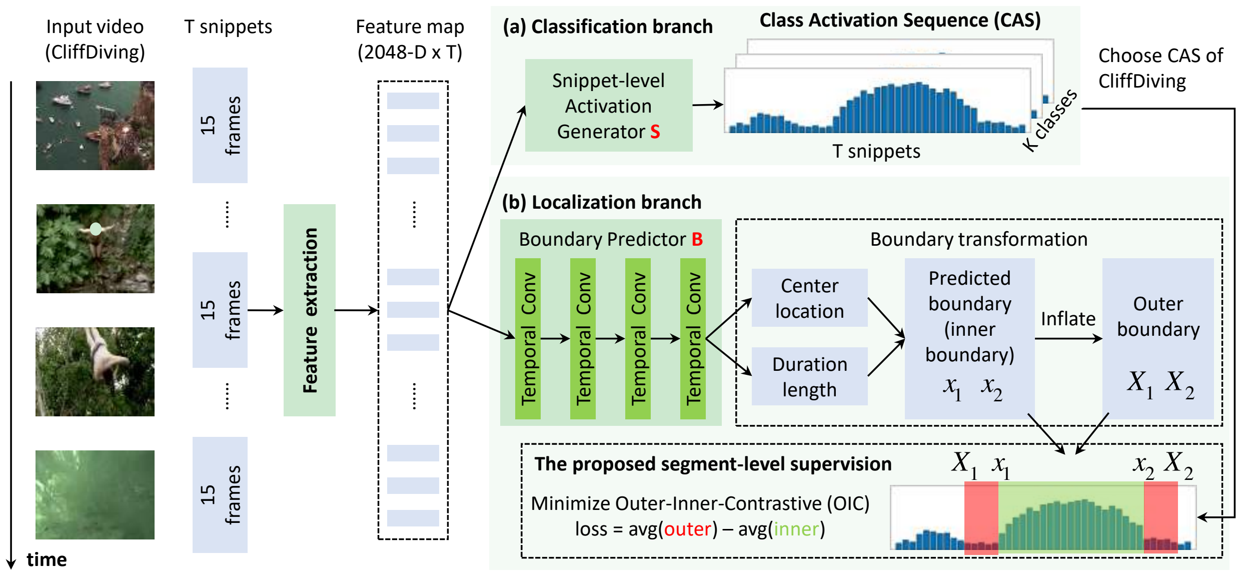
Weakly-supervised deep learning methods have been also widely studied in other vision tasks such as object detection [88; 104; 105; 106; 107; 108; 109; 110; 111; 112; 113; 114], semantic segmentation [115; 116; 117; 118], video captioning [119], visual relation detection [120], etc. As a counterpart of the weakly-supervised video TAL, the weakly-supervised image object detection has been significantly improved via combining Multiple Instance Learning (MIL) [121] and deep networks [105; 107; 109; 112; 113]: built upon Fast-RCNN [43], these methods first generated candidate proposals beforehand; then they employed deep networks to classify each proposal and the scores from all proposals were fused together to obtain one label prediction for the whole image to be compared with the image-level label. One of such MIL-based deep networks is ContextLocNet [113], which further inflated the prediction box to obtain its outer box to take into account the contextual information. Our work bypasses the costly proposal generation and predicts the boundaries from raw input videos in a single-shot fashion. Although we focus on video TAL in this chapter, it would be also interesting to adapt our method for image object detection in the future.

5.3 Outer-Inner-Contrastive Loss

In this Section, we formulate how to compute the proposed OIC loss during the network forward pass of AutoLoc and prove that the OIC loss is differentiable to the underlying boundary prediction model during the backward pass. The whole pipeline and details of AutoLoc will be presented in Sec. 5.4.

5.3.1 Forward

As illustrated by the bottom-right part in Fig. 5.2, for each predicted segment ϕ, we can compute its OIC loss. Each predicted segment ϕ consists of the action/inner boundary $[x_1, x_2]$, the inflated outer boundary $[X_1, X_2]$, and the action class k. These boundaries are at the snippet-level granularity (for example, boundary $x = 1$ corresponds to the location of the 1-st snippet). In order to fetch the corresponding snippet-level activation on the CAS, we round each boundary of continuous value to its nearest integer (i.e. the location of the nearest snippet). We denote the class activation at the x-th snippet on the CAS of action k as $f_k(x)$. The OIC loss of the prediction ϕ is defined as the



Input video (CliffDiving)
T snippets
15 frames
Feature extraction
Feature map (2048-D x T)
(a) Classification branch
Snippet-level Activation Generator S
Class Activation Sequence (CAS)
T snippets
K classes
Choose CAS of CliffDiving
(b) Localization branch
Boundary Predictor B
Temporal Conv
Boundary transformation
Center location
Duration length
Predicted boundary (inner boundary) x_1 x_2
Inflate
Outer boundary X_1 X_2
The proposed segment-level supervision
Minimize Outer-Inner-Contrastive (OIC) loss = avg(outer) – avg(inner)
X_1 x_1 x_2 X_2
time

Figure 5.2: The network architecture of AutoLoc. Given an input video during training, the video is chunked into T 15-frames-long snippets without overlap. We extract a feature vector of D dimension for each snippet. On top of the features, AutoLoc slides two separate branches over time: one classification branch for predicting the action scores of each snippet to obtain the **Class Activation Sequence (CAS)**; one localization branch for directly predicting the true action boundary which is denoted as the inner boundary and is inflated to obtain the outer boundary. Based on the CAS of the ground truth video-level action, we can compute the Outer-Inner-Contrastive loss (*the average activation in the outer red area minus the average activation in the inner green area*) to provide the needed segment-level supervision for training the boundary predictor

average activation $A_o(\phi)$ in the outer area minus the average activation $A_i(\phi)$ in the inner area:

$$\mathcal{L}_{\text{OIC}}(\phi) = A_o(\phi) - A_i(\phi) = \underbrace{\frac{\int_{X_1}^{X_2} f_k(u)\,du - \int_{x_1}^{x_2} f_k(u)\,du}{(X_2 - X_1 + 1) - (x_2 - x_1 + 1)}}_{A_o(\phi)} - \underbrace{\frac{\int_{x_1}^{x_2} f_k(u)\,du}{(x_2 - x_1 + 1)}}_{A_i(\phi)}. \tag{5.1}$$

During training, we set k to the ground truth action and we minimize $\mathcal{L}_{\text{OIC}}(\phi)$ to encourage high activations inside and penalize high activations outside.

5.3.2 Backward

We prove that the OIC loss is differentiable to the inner and outer boundaries. Therefore, the supervision discovered by the OIC loss can be back-propagated to the underlying boundary prediction model. Detailed derivation can be found in the supplementary material. The gradients corresponding to the predicted segment ϕ w.r.t its inner boundary $[x_1, x_2]$ are as follows:

$$\frac{\partial \mathcal{L}_{\text{OIC}}(\phi)}{\partial x_1} = \underbrace{\frac{f_k(x_1) - A_o(\phi)}{(X_2 - X_1 + 1) - (x_2 - x_1 + 1)}}_{\frac{\partial A_o(\phi)}{\partial x_1}} - \underbrace{\frac{A_i(\phi) - f_k(x_1)}{(x_2 - x_1 + 1)}}_{\frac{\partial A_i(\phi)}{\partial x_1}}; \tag{5.2}$$

$$\frac{\partial \mathcal{L}_{\text{OIC}}(\phi)}{\partial x_2} = \underbrace{\frac{A_o(\phi) - f_k(x_2)}{(X_2 - X_1 + 1) - (x_2 - x_1 + 1)}}_{\frac{\partial A_o(\phi)}{\partial x_2}} - \underbrace{\frac{f_k(x_2) - A_i(\phi)}{(x_2 - x_1 + 1)}}_{\frac{\partial A_i(\phi)}{\partial x_2}}. \tag{5.3}$$

The gradients corresponding to the predicted segment ϕ w.r.t its outer boundary $[X_1, X_2]$ are as follows:

$$\frac{\partial \mathcal{L}_{\text{OIC}}(\phi)}{\partial X_1} = \frac{\partial A_o(\phi)}{\partial X_1} = \frac{A_o(\phi) - f_k(X_1)}{(X_2 - X_1 + 1) - (x_2 - x_1 + 1)}; \tag{5.4}$$

$$\frac{\partial \mathcal{L}_{\text{OIC}}(\phi)}{\partial X_2} = \frac{\partial A_o(\phi)}{\partial X_2} = \frac{f_k(X_2) - A_o(\phi)}{(X_2 - X_1 + 1) - (x_2 - x_1 + 1)}. \tag{5.5}$$

Note that these gradients indeed have the physical meanings about how to adjust the boundaries. For example, in Equation 5.2, $\frac{\partial A_i(\phi)}{\partial x_1}$ represents how much the average inner activation $A_i(\phi)$ is higher than the activation $f_k(x_1)$ at the inner left boundary x_1. If the average inner activation is much higher than the activation at the inner left boundary x_1, x_1 is likely to belong to the background and thus we would like to move x_1 in the positive (right) direction. Similarly, $\frac{\partial A_o(\phi)}{\partial x_1}$ represents how much the activation at the inner left boundary x_1 is higher than the average outer activation. $\frac{\partial \mathcal{L}_{\text{OIC}}(\phi)}{\partial x_1}$ is the adversarial outcome of $\frac{\partial A_o(\phi)}{\partial x_1}$ and $\frac{\partial A_i(\phi)}{\partial x_1}$. Consequently, $\frac{\partial \mathcal{L}_{\text{OIC}}(\phi)}{\partial x_1}$ indicates how the model wants to adjust the inner left boundary x_1 eventually: if $\frac{\partial \mathcal{L}_{\text{OIC}}(\phi)}{\partial x_1} < 0$, x_1 moves in the positive (right) direction; if $\frac{\partial \mathcal{L}_{\text{OIC}}(\phi)}{\partial x_1} > 0$, x_1 moves in the negative (left) direction.

5.4 AutoLoc

In this Section, we walk through the pipeline of AutoLoc as illustrated in Fig. 5.2. The training and testing pipelines are very similar in AutoLoc. So we only explicitly distinguish the training and testing pipelines when any difference appears.

5.4.1 Input Data Preparation and Feature Extraction

Each input data sample fed into AutoLoc is one single untrimmed video. Following UntrimmedNet [86], for each input video, we first divide it into 15-frames-long snippets without overlap and extract feature for each snippet individually.

In particular, Temporal Segment Network (TSN) [122] is a state-of-the-art two-stream network for video analysis. UntrimmedNet [86] has been proven to be effective in training TSN classifier with only the video-level labels. Therefore, we first train an UntrimmedNet network (the soft version) in advance and then use the trained network as our backbone for feature extraction.

This backbone network consists of one spatial stream accepting RGB input and one temporal stream accepting Optical Flow input. For each stream, we employ the Inception network architecture with Batch Normalization [123] and extract the 1024-dimensional feature at the `global_pool` layer. Finally, for each snippet, we concatenate the extracted spatial feature and temporal feature into one feature vector of 2048 dimensions. For each input video of T snippets in total, we obtain a feature map of shape 2048 (channels) by T (snippets).

5.4.2 Classification Branch

The goal of the classification branch is to obtain the Class Activation Sequence (CAS). We build our Activation Generator **S** based on UntrimmedNet. On top of the `global_pool` layer, UntrimmedNet attaches one Fully Connected (FC) layer of K nodes to classify each snippet into K action categories and also attaches another Fully Connected (FC) layer of just 1 node to predict the attention score (importance) for each snippet. The corresponding scores from the spatial stream and the temporal stream are averaged to obtain the final score. For each video, we use these two FC layers in the UntrimmedNet that are trained beforehand to respectively extract a classification score sequence of shape K (actions) by T (snippets) and an attention score sequence of T dimensions. For each snippet, we set its classification scores of all classes to 0 when its attention score is lower than the threshold (7 is chosen via grid search on the THUMOS'14 training set and also works well on ActivityNet); then we regard such a gated classification score as the activation, which ranges within [0, 1]. Finally, for each video, we obtain its CAS of shape K (actions) by T (snippets).

5.4.3 Localization Branch

5.4.3.1 Overview

The goal of the localization branch is to learn a parametric model for predicting the segment boundary directly. Recent fully-supervised TAL methods [90; 100; 91; 92; 93] have shown the effectiveness of regressing anchors for direct boundary prediction: the anchor is a hypothesis of the possible segment; the predicted boundary is obtained by respectively regressing (1) the center location and (2) the temporal length of the anchor segment; multi-anchor mechanism is used to cover the possible segments of different temporal scales. Therefore, we design a localization network **B** to look at each temporal position on the feature map and output the needed two boundary regression values for each anchor. Then we regress the anchors using these regression values to obtain the predicted action boundaries (inner boundaries) and inflate the inner boundaries to obtain the outer boundaries. Finally, based on the CAS, we introduce an OIC layer equipped with the OIC loss to generate the final segment predictions.

5.4.3.2 Network Architecture of the Localization Network B

Given an input video, its feature map of shape 2048 channels by T snippets is fed into **B**. **B** first stacks 3 same temporal convolutional layers, which slide convolutional filters over time. Each temporal convolutional layer has 128 filters, which all have kernel size 3 in time with stride 1 and padding 1. Each temporal convolutional layer is followed by one Batch Normalization layer and one ReLU layer.

Finally, **B** adds one more temporal convolutional layer `pred` to output the boundary regression values. Filters in `pred` have kernel size 3 in time with stride 1 and padding 1. Similar to YOLO [98; 99], the boundary predicted by **B** is designed to be class-agnostic. This allows us to learn a generic boundary predictor, which may be used for generating action proposals for unseen actions in the future. Consequently, the total number of filters in `pred` is $2M$, where M is the number of anchor scales. For each anchor, **B** predicts two boundary regression values: (1) t_x indicating how to shift the center location of the anchor and (2) t_w indicating how to scale the length of the anchor.

5.4.3.3 Details of the Boundary Transformation

Since each temporal position on the feature map and each temporal position on the CAS both correspond to the same location of an input snippet, we make boundary predictions at the snippet-level granularity. We outline the boundary prediction procedure in Fig. 5.3.

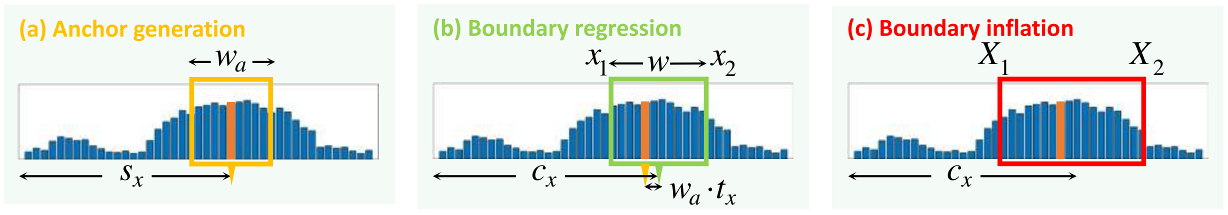


Figure 5.3: Illustration of the boundary prediction procedure which consists of three steps sequentially: (1) **anchor generation** to obtain the boundary hypothesis; (2) **boundary regression** to obtain the predicted boundary of the action segment (denoted as the inner boundary); (3) **boundary inflation** to obtain the outer boundary. The score sequence is CAS and the orange score bar indicates the temporal position that the boundary predictor currently looks at

Anchor generation. At the temporal position s_x on the feature map, we generate a hypothesized segment (anchor) of length w_a. In practice, we use multi-scale anchors. We determine their scales according to the time duration range of the ground truth segments in the training set.

Boundary regression. As aforementioned, for each anchor at the temporal position s_x, **B** predicts two boundary regression values t_x and t_w. We can obtain the predicted segment via regressing the center location $c_x = s_x + w_a \cdot t_x$ and the temporal length $w = w_a \cdot \exp(t_w)$. We denote the boundary of this predicted segment as the inner boundary, which can be computed by $x_1 = c_x - w/2$ and $x_2 = c_x + w/2$. Furthermore, we clip the predicted boundary x_1 and x_2 to fit into the range of the whole video.

Boundary inflation. A ground truth segment usually exhibits relatively higher activations on CAS within the inner area $[x_1, x_2]$ compared to the contextual area preceding x_1 and succeeding x_2. Therefore, we inflate the inner boundary by a ratio α to obtain the corresponding outer boundary $X_1 = x_1 - w \cdot \alpha$ and $X_2 = x_2 + w \cdot \alpha$. We experimentally find that setting α to 0.25 is a good choice.

5.4.3.4 The OIC layer for Obtaining the Final Predictions

Finally, we introduce an OIC layer which uses the OIC loss to measure how likely each segment contains actions and then removes the segments that are not likely to contain actions. During testing, this OIC layer outputs a set of predicted segments. During training, this OIC layer further computes the total OIC loss and back-propagates the gradients to the underlying boundary prediction model.

Concretely, given an input video, the classification branch generates its CAS and the localization branch predicts the candidate class-agnostic segments. Note that since all temporal convolutional layers in **B** slide over time with stride 1, the set of segments predicted at each temporal position on the feature map and the activations at each temporal position on the CAS are paired, corresponding to the same input snippet. Thus at the temporal position of each snippet, **B** has predicted M class-agnostic anchor segments. Then for each action, we iteratively go through the following steps on the CAS to obtain the final class-specific segment predictions. Note that during training we consider only the ground truth actions while during testing we consider all actions. If a temporal position has the activation lower than 0.1 on the CAS, we discard all the predictions corresponding to this temporal position. For each of the remaining positions, among its M anchor segment predictions, we only keep the one with the lowest OIC loss which means selecting the anchor of the most likely scale. Finally, for all the kept segment predictions, we remove the segment predictions with the OIC loss higher than -0.3. We perform Non-Maximum Suppression (NMS) over all segment predictions with overlap IoU threshold 0.4. All these thresholds are chosen by grid search on the THUMOS'14 training set and also work well on ActivityNet.

During training, the total loss is the summation of the OIC loss generated by each kept segment predictions. We can compute the gradients triggered by each kept segment prediction according to Sec. 5.3.2 and then accumulate them together to update the underlying boundary predictor **B**. During testing, all the kept segment predictions are outputted as our final segment predictions. Each segment prediction consists of (1) the predicted action class, (2) the confidence score which is set to 1 minus its OIC loss, and (3) the start time and the end time obtained by converting the inner boundary $[x_1, x_2]$ from the snippet-level granularity (continuous value before rounding to its nearest integer) to time.

5.5 Experiments

In this section, we first introduce two standard benchmarks and the corresponding evaluation metrics. Note that during training, we only use the video-level labels; during testing, we use the ground truth segments with boundary annotations for evaluating the performance of temporal action localization. We compare our method with the state-of-the-art methods and then conduct some ablation studies to investigate different variants of our method.

5.5.1 Datasets and Evaluation

5.5.1.1 THUMOS'14

The temporal action localization task in THUMOS'14 [38] contains 20 actions. Its validation set has 200 untrimmed videos. Each video contains at least one action. We use these 200 videos in the validation set for training. The trained model is tested on the test set which contains 213 videos.

5.5.1.2 ActivityNet v1.2

To facilitate comparisons, we follow Wang *et al.* [86] to use the ActivityNet [82] release version 1.2 which covers 100 activity classes. The training set has 4,819 videos and the validation set has

2,383 videos. We train on the training set and test on the validation set.

5.5.1.3 Evaluation Metrics

Given the testing videos, the system outputs a rank list of action segment predictions. Each prediction contains the action class, the starting time and the ending time, and the confidence score. We follow the conventions [38; 83] to evaluate mean Average Precision (mAP). Each prediction is regarded as correct only when (1) the predicted class is correct and (2) its temporal overlap IoU with the ground truth segment exceeds the evaluation threshold. We do not allow duplicate detections for the same ground truth segment.

5.5.2 Implementation Details

We implement our AutoLoc using Caffe [66]. We use the stochastic gradient descent algorithm to train AutoLoc. Through the experimental studies, we find that the training process can converge quickly on both THUMOS'14 and ActivityNet datasets after 1 training epoch. Following Faster R-CNN [9], during each mini-batch, we process one whole untrimmed video. The learning rate is initially set to 0.001 and is reduced by one order of magnitude for every 200 iterations. We set the weight decay to 0.0005. We choose anchors of the snippet-level length 1, 2, 4, 8, 16, 32 for THUMOS'14 and 16, 32, 64, 128, 256, 512 for ActivityNet. We use CUDA 8.0 and cuDNN v5. We use one single NVIDIA GeForce GTX TITAN X GPU.

Table 5.1: Comparisons with the state-of-the-art methods in terms of temporal localization mAP (%) under different IoU thresholds on THUMOS'14 test set. Weak supervision means training with the video-level labels only. Full supervision indicates that the segment-level boundary annotations are used during training

Supervision	IoU threshold	0.3	0.4	0.5	0.6	0.7
Full	Karaman *et al.* [37]	0.5	0.3	0.2	0.2	0.1
Full	Wang *et al.* [36]	14.6	12.1	8.5	4.7	1.5
Full	Heilbron *et al.* [52]	-	-	13.5	-	-
Full	Escorcia *et al.* [59]	-	-	13.9	-	-
Full	Oneata *et al.* [35]	28.8	21.8	15.0	8.5	3.2
Full	Richard and Gall [51]	30.0	23.2	15.2	-	-
Full	Yeung *et al.* [60]	36.0	26.4	17.1	-	-
Full	Yuan *et al.* [62]	33.6	26.1	18.8	-	-
Full	Yuan *et al.* [124]	36.5	27.8	17.8	-	-
Full	S-CNN [67]	36.3	28.7	19.0	10.3	5.3
Full	SST [96]	37.8	-	23.0	-	-
Full	CDC [125]	40.1	29.4	23.3	13.1	7.9
Full	Dai *et al.* [126]	-	33.3	25.6	15.9	9.0
Full	SSAD [90]	43.0	35.0	24.6	-	-
Full	TURN TAP [91]	44.1	34.9	25.6	-	-
Full	R-C3D [92]	44.7	35.6	28.9	-	-
Full	SS-TAD [89]	45.7	-	29.2	-	9.6
Full	Gao *et al.* [93]	50.1	**41.3**	**31.0**	19.1	9.9
Full	SSN [100]	**51.9**	41.0	29.8	**19.6**	**10.7**
Weak	Sun *et al.* [49]	8.5	5.2	4.4	-	-
Weak	Hide-and-Seek [87]	19.5	12.7	6.8	-	-
Weak	Wang *et al.* [86]	28.2	21.1	13.7	-	-
Weak	Ours - AutoLoc	**35.8**	**29.0**	**21.2**	**13.4**	**5.8**

5.5.3 Comparisons with the State-of-the-art

The results on THUMOS'14 are shown in Table 5.1. Our method significantly outperforms the state-of-the-art weakly-supervised TAL methods that are trained with the video-level labels only. Regarding to the recent weakly-supervised TAL methods (i.e. Hide-and-Seek [87] and Wang *et*

Table 5.2: Comparisons with the state-of-the-art methods in terms of temporal localization mAP (%) under different IoU thresholds on ActivityNet v1.2 validation set. Weak supervision means training with the video-level labels only. Full supervision indicates that the segment-level boundary annotations are used during training

Supervision	IoU threshold	0.5	0.55	0.6	0.65	0.7	
Full	SSN [100]	41.3	38.8	35.9	32.9	30.4	
Weak	Wang *et al.* [86]	7.4	6.1	5.2	4.5	3.9	
Weak	Ours - AutoLoc	**27.3**	**24.9**	**22.5**	**19.9**	**17.5**	

Supervision	IoU threshold	0.75	0.8	0.85	0.9	0.95	Avg
Full	SSN [100]	27.0	22.2	18.2	13.2	6.1	26.6
Weak	Wang *et al.* [86]	3.2	2.5	1.8	1.2	0.7	3.6
Weak	Ours - AutoLoc	**15.1**	**13.0**	**10.0**	**6.8**	**3.3**	**16.0**

al. [86]), although they can generate reasonably good CAS, TAL is done by applying simple thresholding on the CAS which might not robust be to noises in CAS. Our method directly predicts the segment boundary with the contextual information taken into account. Our method can even achieve better or comparable results to some fully-supervised methods (e.g. S-CNN [67]) that are trained with the segment-level boundary annotations. The results of SSN [100] correspond to the model of the same backbone network architecture as ours.

The results on ActivityNet v1.2 are shown in Table 5.2 and our method can achieve substantial improvements again. Wang *et al.* [86] did not report temporal localization results on ActivityNet in their paper. But their trained models and source codes have been released online publicly and thus we can evaluate their results on ActivityNet as well.

5.5.4 Discussions

In this Section, we address several questions quantitatively to analyze our model.

5.5.4.1 Q1: How Effective is the Proposed OIC Loss?

In order to evaluate the effectiveness of the proposed OIC loss, we enumerate all candidate segments at the snippet-level granularity (for example, a segment starting at the location of the 2-nd snippet and ending at the location of the 6-th snippet). We leverage the OIC loss to measure how likely each segment contains actions and then select the most likely ones. Concretely, for each segment, we compute its OIC loss of being each action. Then we follow Sec. 5.4.3.4 to remove segments with high OIC loss and remove duplicate predictions via NMS. We denote this approach as **OIC Selection**. As shown in Table 5.3, although not as good as AutoLoc, **OIC Selection** still significantly improves the state-of-the-art results [86]. Because the OIC loss explicitly favors the segment which has high activations inside and low activations outside, and also such a segment of low OIC loss is usually well aligned to the ground truth segment. This confirms the effectiveness of the proposed OIC loss.

5.5.4.2 Q2: How Important is Looking into the Contrast Between the Inner Area and the Outer Area?

The core idea of the OIC loss is encouraging high activations in the inner area while penalizing high activations in the outer area. We consider another variant that can also discover the segment-level supervision but does not model the contrast between inner and outer. Specifically, we change the OIC loss in AutoLoc to **Inner Only Loss**, which only encourages high activations inside the segment but does not look into the contextual area. As shown in Table 5.3, the performances drop a lot. Consequently, when designing the loss for training the boundary predictor, it is very important and effective to take into account the contrast between the inner area and the outer area.

Notably, the idea of looking into the contrast between inner and outer is related to the usage of **Laplacian of Gaussian (LoG)** filter for blob detection [127]. The operation of computing the OIC loss is effectively convolving the CAS with a step function as shown in Fig. 5.4, which can be regarded as a variant of the LoG filter for the sake of easing the network training. The integral of the LoG filter and the integral of the step function are both zero on the range $(-\mathrm{Inf}, +\mathrm{Inf})$. Further, we approach the scale selection in blob detection by the multi-anchor mechanism and the boundary regression method. Despite the simplicity of the OIC loss, it turns out to be quite effective in practice for localizing likely action segments.

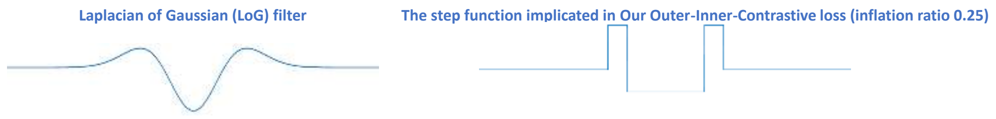

Laplacian of Gaussian (LoG) filter
The step function implicated in Our Outer-Inner-Contrastive loss (inflation ratio 0.25)

Figure 5.4: Illustration of the LoG filter and our OIC loss, which in effect is a step function

Table 5.3: Temporal localization mAP (%) under different IoU thresholds on ActivityNet v1.2 validation set. All approaches are trained with weak supervision

IoU threshold	0.5	0.55	0.6	0.65	0.7	
Wang *et al.* [86]	7.4	6.1	5.2	4.5	3.9	
Ours - AutoLoc	**27.3**	**24.9**	**22.5**	**19.9**	**17.5**	
Q1: OIC Selection	15.8	13.7	11.9	10.3	8.8	
Q2: Inner Only Loss	4.6	3.7	2.7	1.9	1.3	
Q3: Direct Optimization	21.8	19.6	17.8	15.8	13.8	
IoU threshold	**0.75**	**0.8**	**0.85**	**0.9**	**0.95**	**Avg**
Wang *et al.* [86]	3.2	2.5	1.8	1.2	0.7	3.6
Ours - AutoLoc	**15.1**	**13.0**	**10.0**	**6.8**	**3.3**	**16.0**
Q1: OIC Selection	7.5	6.4	5.1	3.6	2.2	8.5
Q2: Inner Only Loss	0.9	0.5	0.2	0.1	0.0	1.6
Q3: Direct Optimization	11.7	9.8	7.8	5.5	2.7	12.6

5.5.4.3 Q3: What is the Advantage of Learning a Model on the Training Videos Compared to Directly Optimizing the Boundaries on the Testing Videos?

AutoLoc trains a model on the training videos and then applies the trained model to perform inference on the testing videos. Alternatively, without training the boundary predictor **B** on the training videos, we can directly train/optimize **B** from scratch on each testing video individually: we follow the testing pipeline as described in Sec. 5.4.3.4 while we also conduct the back-propagation to update **B** to iteratively find likely segments on each testing video. We refer this approach as **Direct Optimization**. As shown in Table 5.3, its performance is not bad, which confirms the effectiveness of the OIC loss again. But it is still not as good as AutoLoc. Because **Direct Optimization** optimizes the predicted boundaries according to the testing video's CAS, which may not be very accurate. Eventually **Direct Optimization** overfits such an inaccurate CAS and thus results into imperfect boundary predictions. In AutoLoc, **B** has been trained on multiple training videos and thus is robust to the noises in CAS. Consequently, AutoLoc may still predicts good boundary even when the testing video's CAS is not perfect. Furthermore, **Direct Optimization** requires optimizing the boundary predictions on the testing video until convergence and thus its testing speed is much slower than AutoLoc. For example, on ActivityNet, **Direct Optimization** converges after 25 training iterations (25 forward passes and 25 backward passes). However, AutoLoc directly applies the trained model to do inference on the testing video and thus requires only one forward pass during testing.

5.6 Summary

Temporal Action Localization (TAL) in untrimmed video is important for many applications. But it is very expensive to annotate the segment-level ground truth (action class and temporal boundary). This raises the interest of addressing TAL with weak supervision, namely only video-level annotations are available during training). However, the state-of-the-art weakly-supervised TAL methods only focus on generating good Class Activation Sequence (CAS) over time but conduct simple thresholding on CAS to localize actions.

In this chapter, we argue that the thresholding method is ad-hoc and the right way is to directly predict the temporal boundary of each action instance. Therefore, we develop a novel weakly-supervised TAL framework called AutoLoc to predict segment boundary via regression. In order to obtain the segment-level supervision needed for training such a boundary predictor, we propose a novel Outer-Inner-Contrastive (OIC) loss to encourage the predicted segment to have high activations inside and low activations in its contextual area. Our method achieves dramatically improved performance: under the IoU threshold 0.5, our method improves mAP on THUMOS'14 from 13.7% to 21.2% and mAP on ActivityNet v1.2 from 7.4% to 27.3%, confirming our hypothesis regarding the need of direct boundary prediction. Our ablation studies also confirm the effectiveness of our design choices for the proposed OIC loss.

Chapter 6

Online Detection of Action Start

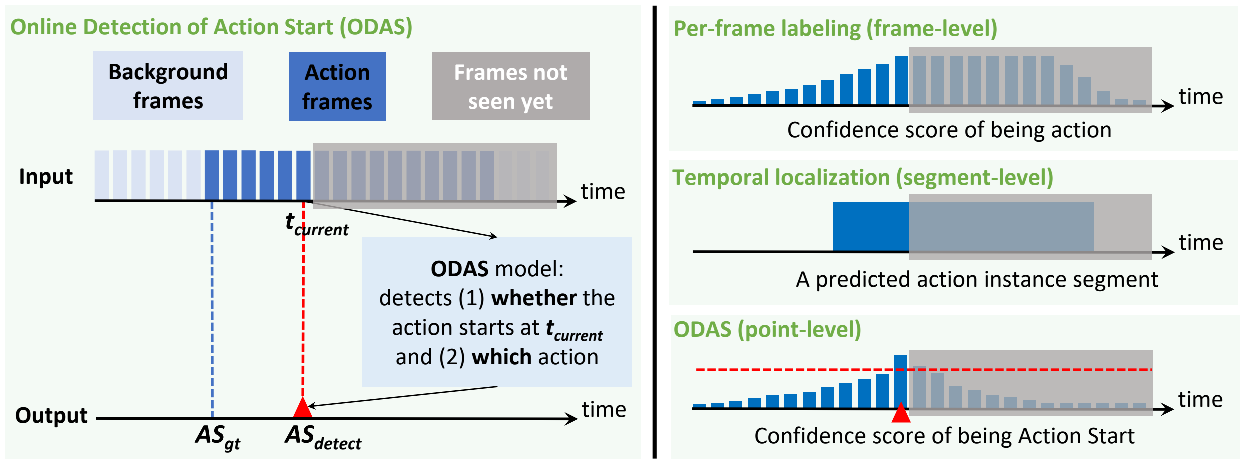
6.1 Introduction

In this chapter, we formulate a new action detection task called **Online Detection of Action Start (ODAS)** to investigate action understanding in a novel scenario with incomplete input data:

i. **Online detection** requires continuously monitoring the live video stream in real time. When a new video frame arrives, online detection system processes it immediately, without any side information or access to the future frames.

ii. We refer the start/onset of an action instance as its **Action Start (AS)**, which is a time point and is associated with one action instance.

iii. Following recent online detection works [128; 129], we target **untrimmed, long, unconstrained** videos with large amounts of complex background streams. ODAS differs from prior



Online Detection of Action Start (ODAS)
Background frames
Action frames
Frames not seen yet
Input
time
$t_{current}$
ODAS model: detects (1) whether the action starts at $t_{current}$ and (2) which action
Output
time
AS_{gt}
AS_{detect}
Per-frame labeling (frame-level)
time
Confidence score of being action
Temporal localization (segment-level)
time
A predicted action instance segment
ODAS (point-level)
time
Confidence score of being Action Start

Figure 6.1: *Left*: Illustration of the novel **Online Detection of Action Start (ODAS)** task; *Right*: comparisons with the conventional action detection tasks

works on early event detection such as [130; 131], which targeted relatively simple videos and assumed that each video contained only one action instance and which the action class is going to happen is known beforehand. ODAS targets the more practical setting that each video can contain multiple action instances and the action class in the testing video is not known in advance.

As illustrated in Fig. 6.1 left, ODAS aims to detect the occurrence and class of AS as soon as the action happens. ODAS is very important in many practical application scenarios, such as early alert generation. For example, the surveillance camera monitoring system needs to detect AS and then issue an alert as soon as possible to allow timely security response; autonomous driving car needs to detect AS of accidents happening in front of it as soon as possible so that the car can slow down or change course timely to avoid collision; robot looking after walking-impaired people shall detect AS of falling as soon as possible to provide assistance before the person has fallen down already. Consequently, in each of such scenarios, it is important to detect AS timely and accurately.

As illustrated in Fig. 6.1 right, ODAS differs from the conventional action detection tasks. Recent online detection works [128; 129] target the **per-frame labeling** task which aims to correctly classifying every frame into either the background class or certain action classes. Recent offline detection works [52; 59; 51; 60; 62; 67; 125; 90; 132; 100; 91; 92; 133; 126; 93; 96; 89; 124; 49; 134; 135; 136] target the **temporal localization** task which aims to predict a set of action segment instances in a long, untrimmed video. Despite lacking the need to correctly classify every frame (as in per-frame labeling) or localize the complete action segment instance (as in temporal localization), ODAS explicitly focuses on detecting AS, which is quite challenging as discussed in the next paragraph. Traditional methods for per-frame labeling and temporal localization can indeed be adapted for ODAS. But since they were originally designed to address different problems, the challenges in detecting AS have not been specifically considered and deeply investigated. Therefore, methods excelling at per-frame labeling or temporal localization might not perform well in ODAS.

In this chapter, we identify three **challenges** in training a good ODAS model and accordingly propose three novel **solutions**. **(Challenge 1)** As the example shown in Fig. 6.2, it is important to learn and detect characteristics that can correctly distinguish the start window from the background, which precedes AS and may share very similar scenes but without the actual occurrence of actions. Note that we follow the state-of-the-art video classification models such as C3D [46; 16], TSN [122], I3D [137] to accept short temporal sliding window as the network input. To address this challenge, we introduce an auxiliary generative network trained in an adversarial process

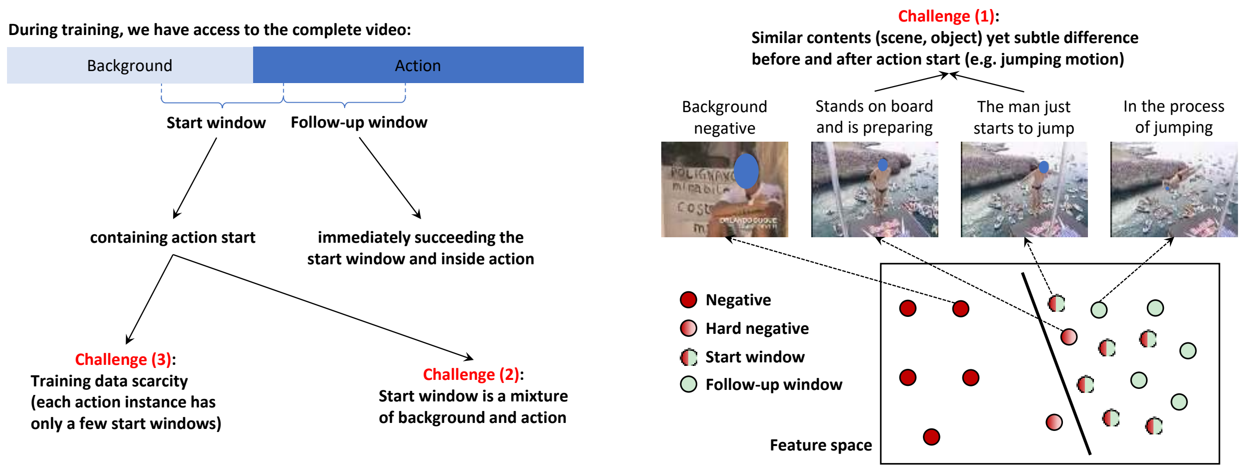

During training, we have access to the complete video:
Background
Action
Start window
Follow-up window
containing action start
immediately succeeding the start window and inside action
Challenge (3):
Training data scarcity (each action instance has only a few start windows)
Challenge (2):
Start window is a mixture of background and action
Challenge (1):
Similar contents (scene, object) yet subtle difference before and after action start (e.g. jumping motion)
Background negative
Stands on board and is preparing
The man just starts to jump
In the process of jumping
Negative
Hard negative
Start window
Follow-up window
Feature space

Figure 6.2: We identify three challenges in training a good ODAS model

to automatically **generate hard negative samples** during training. Although hard negative data may be rare in the training videos, our generator directly learns to model the distribution of hard negatives and thus can generate a much larger pool of hard negatives. **(Challenge 2)** We define the start window and its follow-up window in Fig. 6.2. A start window contains both action frames and background frames. Background preceding action can provide temporal contextual information but can also be confusing. Due to the shared contents (background scene and object), the feature of the start window may be closer to the preceding background window than the actual action window after the start. To remedy this issue, since the follow-up window is completely inside action, we propose to model the **temporal consistency** between the start window and its follow-up window during training. **(Challenge 3)** It is important to accurately classify start windows in ODAS. But each action instance only has a few training samples of start windows, and thus the number of training samples for start windows is much more scarce than others such as background windows and windows fully inside action. To address this issue, we design an **adaptive sampling** strategy to increase the percentage of start windows in each training batch. Our experiments in Sec. 6.6 will prove the effectiveness and necessity of each proposed method and putting three methods together results in significant performance gains.

In the rest of this chapter, we review the related work in Section 6.2, introduce our ODAS framework in Section 6.3, propose three novel methods to improve the capability of the backbone networks at detecting action in a timely manner in Section 6.4, discuss the evaluation metrics in Section 6.5, present the experimental results in Section 6.6, and finally summarize this chapter in Section 6.7.

6.2 Related Work

6.2.1 Temporal Action Localization

Given a long, untrimmed video, temporal action localization needs to temporally localize each action instance: we not only predict its category but also detect when it starts and ends. Although these methods for temporal localization were originally designed for the offline setting, some of them can be adapted to conduct temporal localization in an online manner. However, besides detecting AS, temporal localization requires detecting the Action End (AE) as well. To this end, many methods have to wait for seeing AE in order to localize the action segment as a whole so that can determine AS, resulting in high latency. Also, a good temporal localization method may excel at AE detection but perform poorly in AS detection (considering a detected segment that overlaps with the ground truth segment with IoU 0.7 and has the same AE as the ground truth). ODAS focuses on AS specifically.

6.2.2 Early Recognition and Detection

Similar to ODAS, early recognition and detection also aim to detect action as soon as it happens in streaming videos. Early recognition was effectively formulated as partial action classification [138; 139; 140; 141; 142; 143; 144]: the videos used in early recognition literatures are usually relatively short; during testing, they cut each video to only keep its first certain portion of the whole video, and then classify the cut video into the pre-defined action classes.

ODAS is more related to early detection. Hoai and De la Torre [130; 131] made attempts to detect actions in an online manner yet under a simplified setting (e.g., one action instance per video). Huang *et al.* [145] worked on a scenario that the background contents are simple (i.e. the person is standing and keeping still). In this chapter, like [128; 129], we focus on the realistic videos that are unconstrained and contain complex backgrounds of large variety. Ma *et al.* [19] approached the early detection task by cutting the first certain portion of the whole testing video and then conducting temporal localization on the cut video. Hence, besides detecting AS, this work also focused on detecting whether the action ends or not.

6.2.3 Online Action Detection

Recent works on online action detection are very close to ODAS. De Geest *et al.* [128] first simulated the online action detection problem using untrimmed, realistic videos and benchmarked the existing models. Gao *et al.* [129] designed a training strategy to encourage a LSTM-based Reinforced Encoder-Decoder (RED) Network to make correct frame-level label predictions as early as possible. But both of them formulated online action detection as online per-frame labeling task, which requires correctly classifying every frame rather than just detecting AS. A good per-frame labeling method might not be necessarily good at detecting AS (considering a per-frame labeling method correctly classifying frames in the 30%-100% portion of each action instance but misclassifying frames in its 0%-30% portion). Consequently, as for the applications that detecting AS is the most important task, ODAS is the best fit.

In addition, there are also works on spatio-temporally localizing actions in an online manner but also limited to short videos [146; 147]. Li *et al.* [148] and Liu *et al.* [149] leveraged Kinect sensors and performed detection based on the tracked skeleton information. Vondrick *et al.* [150] targeted future prediction, which is a more ambitious online detection task.

6.2.4 Generative Adversarial Network

The idea of training in an adversarial process was first proposed in [21] and has been adopted in many applications [151; 27; 152; 153; 154]. Generative Adversarial Network (GAN) [21; 155] consists of two networks trained simultaneously to compete with each other: a generator network

G that learns to generate fake samples indistinguishable from real data and a discriminator network D which is optimized to recognize whether input data samples are real or fake. To the best of our knowledge, we are the first to explore GAN for action detection.

6.3 Framework

In this Section, we introduce our ODAS framework as shown in Fig. 6.1. We follow the state-of-the-art video classification networks like C3D [46; 16], TSN [122], I3D [137] to accept temporal sliding windows as input. In particular, we set the window length to 16 frames and use C3D as our backbone network in Sec. 6.3 and Sec. 6.4 to help illustrate technical ideas.

We outline our ODAS framework by walking through the testing pipeline. During testing, when a new frame arrives at the current time t, we immediately feed the streaming window ending at t into our network. The network output at t consists of the semantic class c_t which could be either background or action $1, \dots, K$ (K is the total number of action classes) and the confidence score s_t. In order to detect AS, we compare the network outputs at $t-1$ and t. We generate an AS point prediction whenever the following conditions are all satisfied: (1) c_t is action; (2) $c_t \neq c_{t-1}$; (3) s_t exceeds the threshold obtained by grid search on the training set. Such an AS point prediction is associated with the time point t, the predicted class (set to c_t) and the confidence score (set to s_t).

As alternatives, we have also studied the approach of adding a proposal stage specifically for detecting action start and then classifying the action class. We found such an alternative approach is not as effective as the one outlined above.

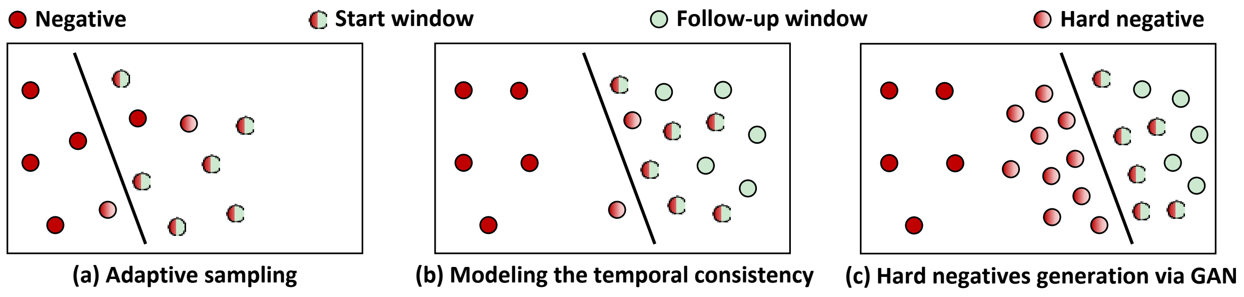

Negative
Start window
Follow-up window
Hard negative
(a) Adaptive sampling
(b) Modeling the temporal consistency
(c) Hard negatives generation via GAN

Figure 6.3: The effects of our proposed three training methods for ODAS. Each rectangle represents data distribution in the high-level feature space after training

6.4 Our Methods

As for training our ODAS model, the complete videos are available during training. We slide windows over time with a stride of 1 frame to first construct a set of training windows to be fed into the network. For each window, we assign its label as the action class of the last frame of the window. In this section, we propose three novel methods to improve the capability of the backbone networks at detecting action in a timely manner. We first illustrate our intuition of designing these methods and then present their formulations. We close this section by summarizing the full objective used for training.

6.4.1 Intuition

6.4.1.1 Adaptively sample the training data

Since we want to detect actions as soon as possible, it is important for ODAS to accurately classify start windows. This is a challenging task because the start window contains various background

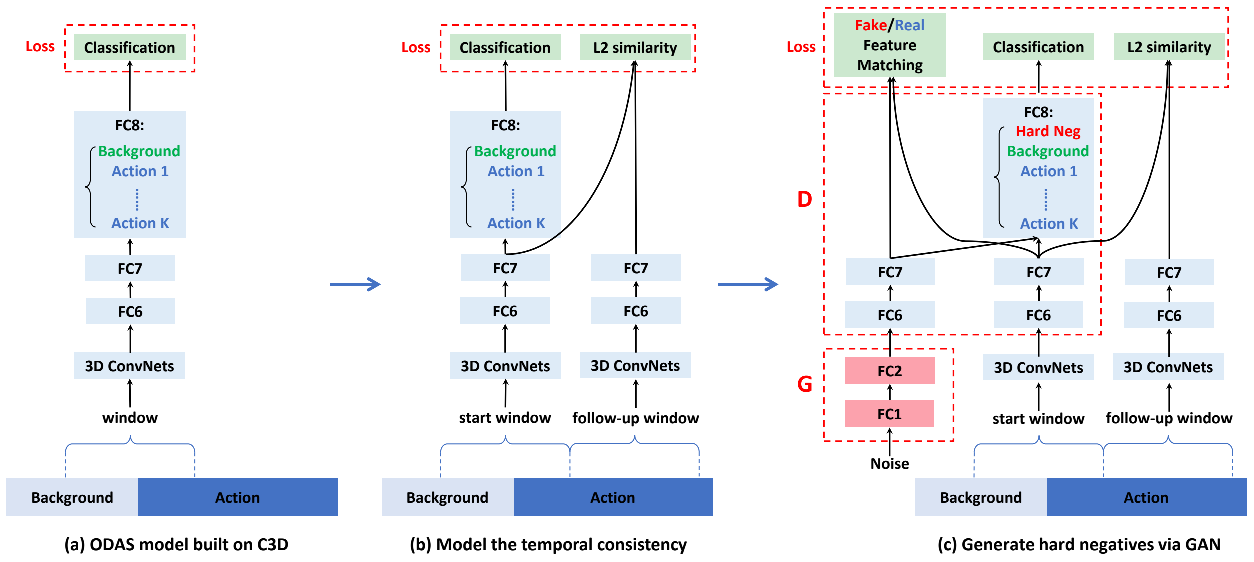


Figure 6.4: The network architectures of our ODAS models built on C3D and the proposed training objectives. (a) Our basic ODAS model consists of 3D ConvNets from `Conv1` to `Pool5` and 3 fully connected layers (`FC6`, `FC7`, `FC8`). We keep the same backbone network architecture as C3D while setting the number of nodes in `FC8` to $K + 1$, standing for K actions and background. The output of `FC8` is used for calculating multi-class classification softmax loss. (b) We model the temporal consistency between the start window and its paired follow-up window by adding a temporal consistency loss term to minimize the L2 similarity computed using their `FC7` activations. Two streams in this Siamese network share the same parameters. (c) Further, we design a GAN-based framework to automatically generate hard negative samples to help our model more accurately distinguish actions against negatives. G is generator and D is discriminator. G accepts random noise as input and output fake `Pool5` features. We add an additional class in `FC8` for fake samples. All blue blocks of the same name are the same layer and share weights.

contents, and the number of start windows is quite scarce. If we construct each training batch via randomly sampling out of all windows, the model might not see enough start windows during training in order to generalize to the testing data well. To solve this issue, we guide the ODAS model to pay more attention to start windows via adaptively sampling more start windows during each training batch. Using this strategy, the ODAS model can learn a better classification boundary to distinguish start windows against negatives more accurately, as illustrated in Fig. 6.3 (a).

6.4.1.2 Model the temporal consistency

As illustrated in Fig. 6.3 (a), the start window is a mixture of action frames and background frames. Therefore, in the feature space, start windows could be close to or even mixed with negatives. It is important to accurately distinguish start windows and negatives in ODAS so that the model can more timely detect action start when the video stream switches from negative to action. As illustrated in Fig. 6.3 (b), since the follow-up windows are completely inside action, they are far away from negative data in the feature space. Thus we explicitly model the **Temporal Consistency (TC)** between each start window and its follow-up window to encourage their feature similarity. Using this training method, start windows move closer to follow-up windows and thus become more separable from negatives.

6.4.1.3 Generate hard negative samples via GAN

As exampled in Fig. 6.2, it is important to train the ODAS model to capture the subtle differences that can serve as evidences for discriminating start windows from negatives preceding AS. As illustrated in Fig. 6.3 (b), hard negatives that have subtle differences with start windows might be close to start windows in the feature space. In order to learn a better decision boundary, we aim to identify such hard negative samples in the training set during training. However, exhaustively finding such samples is time-consuming because such hard negatives are rare and may even not exist in the training data. Therefore, we propose to train a model to automatically synthesize samples that are hard to distinguish from true start windows. As illustrated in Fig. 6.3 (c), equipped by these synthesized hard negatives, we can learn an even better ODAS model to discriminate start windows from negatives.

6.4.2 Adaptively Sample the Training Data

Concretely, we randomly sample half of the training batch from start windows and randomly sample the other half batch from the remaining windows, which can be backgrounds or windows completely inside actions. After each training batch is constructed, we can feed them into our network as shown in Fig. 6.4 (a) and train the network via minimizing the multi-class classification softmax loss $\mathcal{L}_{\text{classification}}$. We denote the set of start windows as $p_{\text{start}} = \{(x_s, y)\}$ where x_s is the start window to be fed into our model and y is its corresponding ground truth label. Similarly, we express the set of remaining windows as $p_{\text{notstart}} = \{(x_{ns}, y)\}$. The label space of y is $1, \ldots, K+1$ where the first K classes are actions and the $K+1$-th class stands for background. Our network takes x as input and predicts a vector $\{o_1, \ldots, o_{K+1}\}$ of $K+1$ dimension. Finally we apply the softmax function to obtain the normalized probability of being class i: $P_{\text{model}}(i|x) = \frac{e^{o_i}}{\sum_{k=1}^{K+1} e^{o_k}}$. We use $\mathbb{E}[\cdot]$ to represent expectation. The classification loss is defined as:

$$\begin{aligned}\mathcal{L}_{\text{classification}} = \quad & \mathbb{E}_{(x_s,y)\sim p_{\text{start}}}\left[-\log\left(P_{\text{model}}(y|x_s)\right)\right] \\ & +\mathbb{E}_{(x_{ns},y)\sim p_{\text{notstart}}}\left[-\log\left(P_{\text{model}}(y|x_{ns})\right)\right].\end{aligned} \tag{6.1}$$

6.4.3 Model the Temporal Consistency

Formally, we denote the training set of the paired start window and follow-up window as $p_{\text{startfollowup}} = \{(x_s, x_f, y)\}$, where x_s represents the start window and x_f is its associated follow-up window and

y is still the ground truth label. We model the temporal consistency via minimizing the similarity $\mathcal{L}_{\text{similarity}}$ measured by L2 distance of the feature representation between x_s and x_f:

$$\mathcal{L}_{\text{similarity}} = \mathbb{E}_{\left(x_s, x_f, y\right) \sim p_{\text{startfollowup}}} \left\| F\left(x_s\right) - F\left(x_f\right) \right\|_2^2, \tag{6.2}$$

where the function $F\left(\cdot\right)$ indicates extracting feature representation. As shown in Fig. 6.4 (b), we set $F\left(\cdot\right)$ to be the output of `FC7` because it is also the input to the final layer `FC8` for classification. Trained with this temporal consistency loss, the features of start windows become more distinguishable from negatives, which leads to the ODAS performance improvements as shown in Sec. 6.6.1.4.

6.4.4 Generate Hard Negative Samples via GAN

In order to separate start windows and hard negatives which have subtle differences from start windows, we design a GAN-based framework which synthesizes hard negative samples to assist ODAS model training.

6.4.4.1 Generator (G)

Since directly generating video is very challenging, we use GAN to generate features rather than raw videos. As shown in Fig. 6.4 (c), our GAN model has a fixed 3D ConvNets (from `Conv1` to `Pool5`) to extract real `Pool5` features from raw videos and also has a G to generate fake `Pool5` features. Upper layers serve as **Discriminator (D)**, which will be explained later.

G accepts a random noise z as input and learns to capture the true distribution of real start windows. Consequently, G has the potential to generate various fake `Pool5` samples which may not exist in the real training set but may appear during testing. This enables our model to continuously explore more discriminative classification boundary in the high-level feature space. Following [155], z is a 100-dimensional vector randomly drawn from the standard normal distribution. In practice, we find that a simple G consisting of two fully connected layers `FC1` and `FC2` works well. Each fully connected layer is followed by a BatchNorm layer and a ReLU layer.

When training G, the principle is to generate **hard** negative samples that are similar to real start windows. Conventional GANs utilize a binary real/fake classifier to provide supervision signal for training G. However, this method usually encounters an instability issue. Following [156], instead of adding a binary classifier, we require G to generate fake data matching the statistics of the real data. Specifically, the feature matching objective is forcing G to match the expectation of the real features on an intermediate layer of D (we use `FC7` layer as indicated in Fig. 6.4 (c)). Formally, we denote the feature extraction part of using the fixed 3D ConvNets as $\phi\left(\cdot\right)$ and the process from `Pool5` to `FC7` as $\psi\left(\cdot\right)$. The feature matching loss is defined as follows:

$$\mathcal{L}_{\text{matching}} = \left|\left|\mathbb{E}_{(x_s,y)\sim p_{\text{start}}}\left[\psi\left(\phi\left(x_s\right)\right)\right] - \mathbb{E}_{z\sim noise}\left[\psi\left(G\left(z\right)\right)\right]\right|\right|_2^2, \tag{6.3}$$

where $G\left(\cdot\right)$ denotes the generator, $p_{\text{start}} = \{(x_s, y)\}$ is the training set of start windows, x_s represents the start window, and y is the ground truth label.

6.4.4.2 Discriminator (D)

The principle for designing D is that the generated samples should be still separable from real start windows despite their similarity, so that the generated samples can be regarded as hard **negatives**. As shown in Fig. 6.4 (c), D consists of `FC6`, `FC7`, and `FC8`. Instead of adding a binary real/fake classifier, we add an additional node in `FC8` layer to represent the hard negative class, which is the ground truth label for the generated samples. Note that this additional class is used during training only and is removed during testing.

Similarly, some previous works also replaced the binary discriminator with a multi-class classifier that has an additional class for the fake samples [156; 157; 158]. However, their motivation is mainly extending GAN to the semi-supervised setting: the unlabeled real samples could belong to any class except fake. But in this chapter, we focus on generating hard negatives which should be similar to actions but dissimilar to backgrounds; meanwhile our D needs to distinguish hard negatives from not only actions but also from backgrounds.

Given a `Pool5` feature $\phi\left(x\right)$ either extracted from real data or generated by G, D accepts $\phi\left(x\right)$ as input and predicts a vector $\{o_1, \ldots, o_{K+2}\}$ which goes through a softmax function to get class probabilities: $P_{\text{D}}\left(i|\phi\left(x\right)\right) = \frac{e^{o_i}}{\sum_{k=1}^{K+2} e^{o_k}}$, where $i \in \{1, \ldots, K+2\}$. Regarding the real samples, we can calculate their corresponding classification loss $\mathcal{L}_{\text{real}}$ via extending $\mathcal{L}_{\text{classification}}$ defined in Eq. 6.1:

$$\begin{aligned} \mathcal{L}_{\text{real}} = \quad & \mathbb{E}_{(x_s,y)\sim P_{\text{start}}}\left[-\log P_D\left(y|\phi\left(x_s\right)\right)\right] \\ & +\mathbb{E}_{(x_{ns},y)\sim P_{\text{notstart}}}\left[-\log P_D\left(y|\phi\left(x_{ns}\right)\right)\right]. \end{aligned} \tag{6.4}$$

As for the generated fake samples, the loss is:

$$\mathcal{L}_{\text{fake}} = \mathbb{E}_{z\sim noise}\left[-\log P_D\left(K+2|G\left(z\right)\right)\right], \tag{6.5}$$

where $K+2$ represents the hard negative class.

6.4.5 The Full Objective

We first pre-train the whole network via minimizing $\mathcal{L}_{\text{classification}}+\lambda\cdot\mathcal{L}_{\text{similarity}}$, which combines the classification loss (Eq. 6.1) and the temporal consistency loss (Eq. 6.2) together with the weighting parameter λ. Based on such initialization, we train G and D in an alternating manner during each iteration: **When training G**, we fix D and train G so that the full objective is $\min\limits_{G}\mathcal{L}_{\text{G}}$. $\mathcal{L}_{\text{G}}$ contains only the feature matching loss (Eq. 6.3): $\mathcal{L}_{\text{G}} = \mathcal{L}_{\text{matching}}$. **When training D**, we fix G and train D so that the full objective is $\min\limits_{D}\mathcal{L}_{\text{D}}$. $\mathcal{L}_{\text{D}}$ contains the classification loss for both the real and fake samples (Eq. 6.4 and Eq. 6.5) and also the temporal consistency loss (Eq. 6.2): $\mathcal{L}_{\text{D}} = \mathcal{L}_{\text{real}} + \mathcal{L}_{\text{fake}} + \lambda\cdot\mathcal{L}_{\text{similarity}}$, where λ is the weight.

6.5 Evaluation

6.5.1 Conventional Protocols and Metrics

Hoai and De la Torre [130; 131] first worked on early detection and proposed three evaluation protocols to respectively evaluate classification accuracy, detection timeliness, and localization precision. As comprehensively discussed in [128], their protocols do not suit online detection in realistic, unconstrained videos, because their protocols were designed for a simplified setting: each video contains only one action instance of interest.

Therefore, as mentioned in Sec. 6.2, recent online detection works [128; 129] effectively worked on the per-frame labeling task and evaluated the frame-level classification mean Average Precision (mAP) or its calibrated version. In addition, temporal localization methods detect both the start and end times of each action instance and evaluate the segment-level detection mAP. As for ODAS, the performance of detecting AS can indeed affect both the frame-level mAP and the segment-level mAP. However, since the frame-level mAP is mainly used to evaluate the accuracy of classifying every frame and the segment-level mAP involves evaluating the correctness of detecting the end, both metrics are not exactly evaluating the performance in detecting action starts.

6.5.2 Proposed New Protocol and Metrics

In order to specifically evaluate ODAS performance, we propose a new evaluation protocol. We evaluate ODAS at the point level for each AS instance. As mentioned in Sec. 6.3, ODAS system outputs a rank list of detected AS points. Each AS point prediction is associated with the time point t, the predicted action class and the confidence score.

We aim to evaluate the detection accuracy and timeliness of ODAS system compared to the human annotated ground truths [1]. As for the timeliness, inspired by the segment-level mAP which measures the temporal overlap between the ground truth segment and the predicted segment, we measures the temporal offset (absolute distance) between the ground truth AS point and the predicted AS point.

We propose the **point-level AS detection mAP** to evaluate ODAS results. Each AS point prediction is counted as correct only when its action class is correct and its offset is smaller than the evaluation threshold. We evaluate the point-level AP for each action class and average over all action classes to get the point-level mAP. We do not allow duplicate detections for the same ground truth AS point.

6.6 Experiments

In order to simulate the ODAS setting, we employ standard benchmarks consisting of untrimmed videos to simulate the sequential arrival of video frames.

[1] According to the human annotations on THUMOS'14 test set [38], people usually can form agreement on when the action starts: out of 3,358 action instances, 3,259 instances (97%) have their AS time annotations agreed by multiple human annotators. The instances with ambiguous start times are excluded during evaluation.

6.6.1 Results on THUMOS'14

6.6.1.1 Dataset

THUMOS'14 [38] involves 20 actions and videos over 20 hours: 200 validation videos (3,007 action instances) and 213 test videos (3,358 action instances). These videos are untrimmed and contain at least one action instance. Each video has 16.8 action instances in average. We use the validation videos for training and use the test videos to simulate streaming videos for testing ODAS.

6.6.1.2 Metrics

We evaluate the point-level AS detection mAP at different temporal offset thresholds to compare ODAS systems under the different user tolerances or application requirements. Note that *AP depth at recall X%* means averaging the precisions at points on the P-R curve with the recall ranging from 0% to X%. The aforementioned default detection mAP is evaluated at AP depth at recall 100%. In order to look at the precision of top ranked predictions, we can evaluate detection mAP at different AP depth. At each AP depth, we average the detection mAP under different offset thresholds to obtain the average mAP.

6.6.1.3 Comparisons

As for **our approach**, our network architecture can be found in Fig. 6.4. Since we build our model upon C3D [46] which has been pre-trained on Sports-1M [14], we use it to initialize models shown in Fig. 6.4. Since `Pool5` output has 8,192 dimensions, we use 4,096 nodes for both `FC1` and `FC2` in G. We compare with the following baselines. (1) **Random guess**: we replace the network output scores for K actions and background mentioned in Sec. 6.3 via randomly splitting score 1 into $K+1$ numbers all within $[0,1]$. (2) **C3D w/o ours**: we use the C3D model which has exactly the same network architecture as our model used during testing but is trained without our proposed methods. (3) **RED**: Gao *et al.* [129] achieved the state-of-the-art performances on THUMOS'14 in online action detection task by encouraging the LSTM network to make correct frame-level predictions at the early part of a sequence. We obtained the results from the authors and evaluated based on our proposed protocol. (4) Per-frame labeling method - **CDC**: Shou *et al.* [125] designed a Convolutional-De-Convolutional Network to operate on the testing video in an online manner to output the per-frame classification scores, which can be used to determine AS points following the same pipeline as proposed in Sec. 6.3. (5) Temporal localization method - **TAG in SSN**: Zhao *et al.* [100] proposed an effective segment proposal method called temporal actionness grouping (TAG). Based on the actionness score sequence, TAG can be operated in an online manner to detect the start of a segment and then also detect its end. Thus TAG can be used for ODAS to generate class-agnostic AS point proposals. For fair comparisons, we determine the action score of each proposal by applying the AS classifier obtained by our best model (trained with all three methods). (6) Shot boundary detection methods [159; 160; 161] detect the change boundaries in the video which can be considered as AS proposals. Then we utilize our best classifier to classify

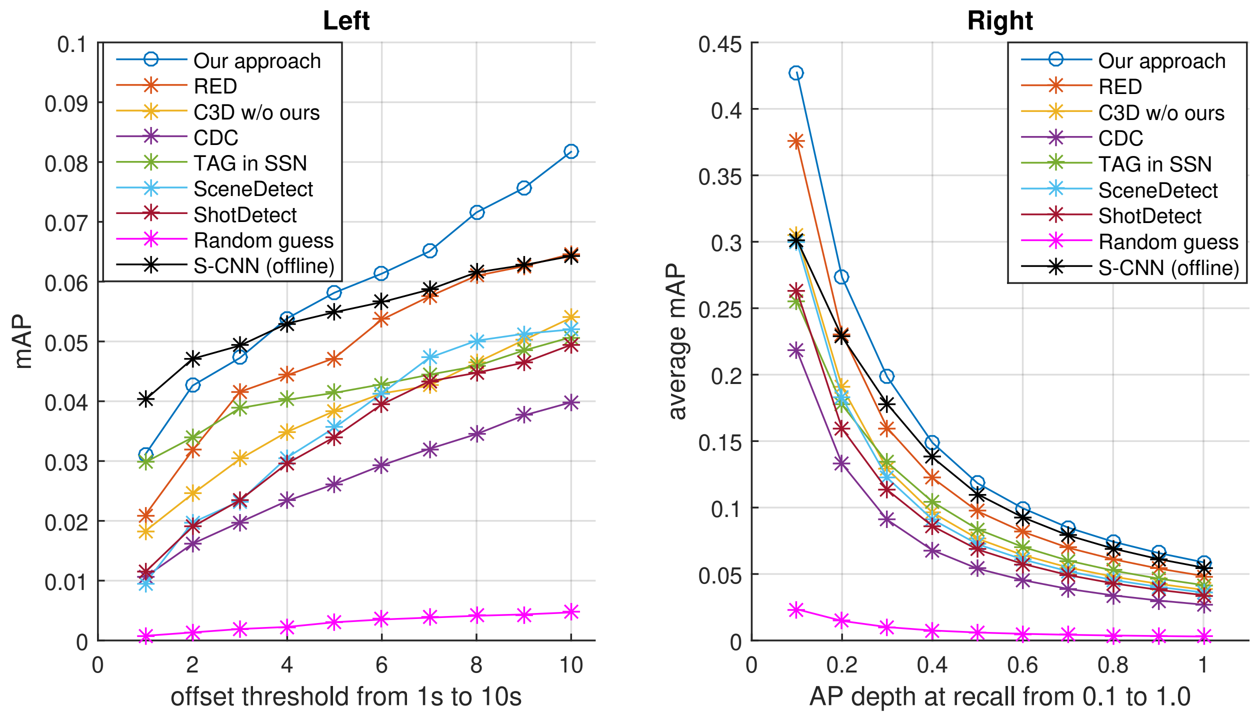


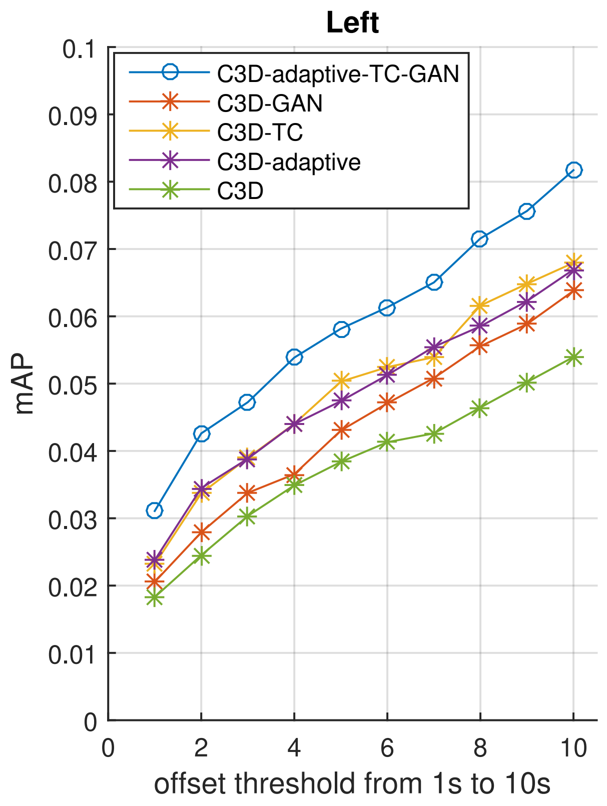
Figure 6.5: Experimental results on THUMOS'14 for comparisons with state-of-the-art methods on THUMOS'14. ***Left***: y-axis is the point-level AS detection mAP and x-axis is varying the offset threshold. ***Right***: y-axis is the average mAP averaged over offsets from 1s to 10s at AP depth at X% recall and x-axis is varying X% from 0.1 to 1.

each AS proposal. We employ two popular open-source shot detection methods **ShotDetect**[2] and **SceneDetect**[3] respectively for comparisons. (7) Offline detection method: **S-CNN** [67] uses the same C3D network architecture as our model but performs testing in offline.

Since the duration of action instance varies from <1s to >20s and thus during evaluation we vary the offset threshold from 1s to 10s. As shown in Fig. 6.5, when using the proposed training strategies specifically designed to tackle ODAS, **our approach** improves the **C3D w/o ours**

[2]https://github.com/johmathe/Shotdetect

[3]https://github.com/Breakthrough/PySceneDetect



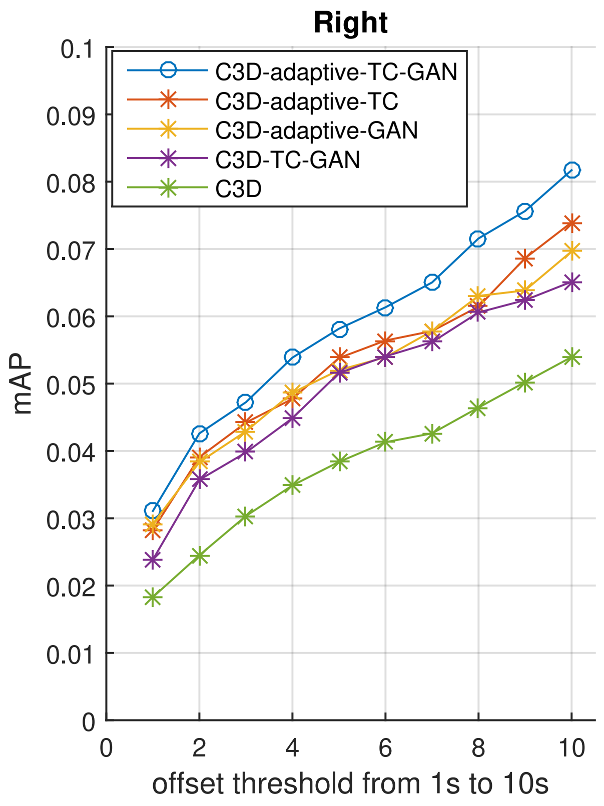


Figure 6.6: Experimental results on THUMOS'14 for ablation study of our methods. ***Left***: y-axis is the point-level AS detection mAP and x-axis is varying the offset threshold. ***Right***: y-axis is the average mAP averaged over offsets from 1s to 10s at AP depth at X% recall and x-axis is varying X% from 0.1 to 1.

baseline by a large margin.

Qualitative comparisons can be found in Fig. 6.7: (a) Once the BasketballDunk action starts, our approach correctly detects it sooner than C3D w/o ours; (b) In this Billiards example, our approach detects action start exactly when the Billiards action begins and much sooner than C3D w/o ours; (c) In this example consisting of a CricketBowling instance and a CricketShot instance back-to-back, our approach detects the start of CricketBowling timely and detects the start of CricketShot exactly when it begins, but C3D w/o ours misses the CricketBowling instance and

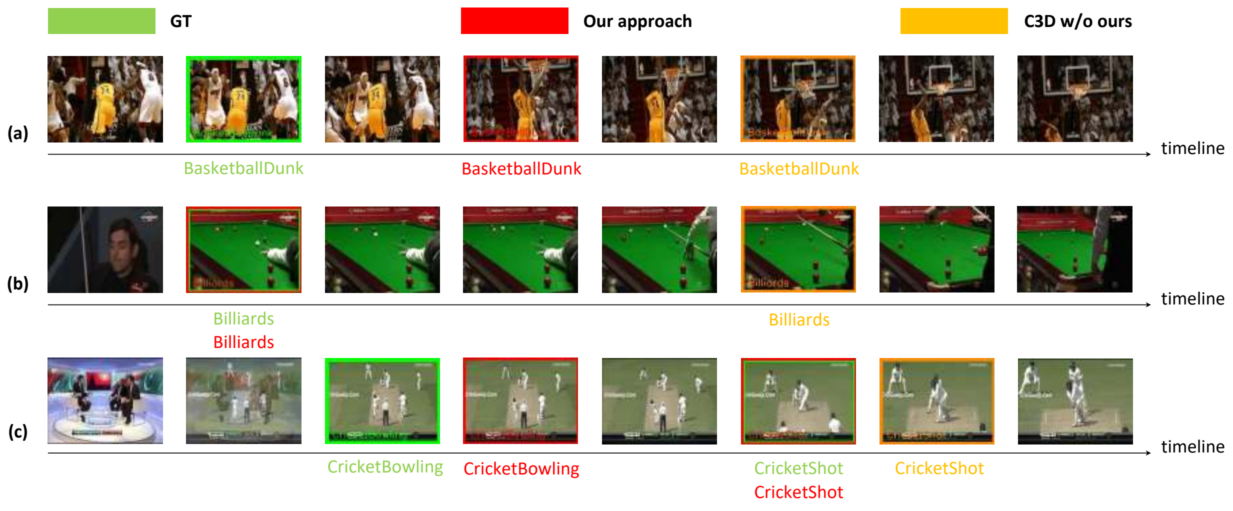


Figure 6.7: Qualitative comparisons on THUMOS'14. Green indicates the ground truth of AS; Red indicates the AS detected by the C3D-based ODAS model trained with our proposed methods; Orange indicates the AS detected by the same C3D model trained without our proposed methods

detects the CricketShot instance with delay.

Notably, **our approach** is far better than **random guess** and also outperforms **RED**, which is a state-of-the-art method developed very recently to detect action as soon as possible. Also, our method is better than other per-frame labeling method (i.e. **CDC**) and some class-agnostic start point detection methods (i.e. **TAG in SSN**, **ShotDetect**, **SceneDetect**). Furthermore, **our approach** can even achieve better results under high offset thresholds than the offline **S-CNN** method.

6.6.1.4 Ablation study of individual proposed method

We conduct in-depth study on THUMOS'14 to analyze the performance gain contributed by each proposed training method. In Fig. 6.6 left, we report the results on THUMOS'14 when training with only one of our proposed methods. All approaches in the following have the same network architecture during testing: **C3D** is trained without any proposed methods; **C3D-adaptive** is trained with the adaptive sampling strategy; **C3D-TC** is trained with modeling the temporal consistency; **C3D-GAN** is trained within our proposed GAN-based framework; **C3D-adaptive-TC-GAN** combines all three proposed methods together during training and achieves the best performance. These results indicate that all three proposed methods are effective in improving the ODAS model.

In Fig. 6.6 right, we report the results on THUMOS'14 when training without one of our proposed methods. We add additional approaches of the same network architecture during testing: **C3D-TC-GAN** is trained without the adaptive sampling strategy; **C3D-adaptive-GAN** is trained without modeling the temporal consistency; **C3D-adaptive-TC** is trained without our proposed GAN-based framework. These results indicate that all three proposed methods are necessary for training a good ODAS model.

6.6.2 Results on ActivityNet

6.6.2.1 Dataset

ActivityNet [82; 83] v1.3 involves 200 actions and untrimmed videos over 800 hours: around 10K training videos (15K instances) and 5K validation videos (7.6K instances). Each video has 1.7 action instances in average. We train on the training videos and evaluate ODAS using the validation videos.

Table 6.1: AS detection mAP (%) on ActivityNet when varying the offset threshold

offset threshold (s)	10	50	100
Random guess	0.06	0.14	0.17
SceneDetect	4.71	18.93	25.84
ShotDetect	6.10	24.35	33.76
TSN w/o ours	8.18	31.39	44.15
Our approach	**8.33**	**33.08**	**46.97**

6.6.2.2 Comparisons

As for our approach, given the superior performances of TSN on ActivityNet video classification task [122], following [93], for each window of 16 frames, we use TSN to extract a feature vector of 3,072 dimensions to serve as input to our network. Our backbone network for ActivityNet consists of three fully connected layers (i.e. `FC6`, `FC7`, `FC8`) that are the same as these in C3D, but we train this network directly from scratch. As for G, since the dimension of fake samples here is 3,072, we set `FC1` and `FC2` in G to be 2,048 dimensions.

The duration of action instance varies from $<$1s to $>$200s in ActivityNet and thus during evaluation we vary the temporal offset from 10s to 100s. As shown in Table 6.1, **our approach** significantly outperform the baseline methods again and improves **TSN w/o ours** which indicates that it also accepts TSN features as input and has the same testing network architecture as **our approach** but is trained without our proposed methods.

6.6.3 Efficiency

In terms of testing speed, unlike offline detection which evaluates how many frames can be processed per second simultaneously, it is important for ODAS to evaluate the detection delay which is the time duration between the system receives a new video frame and the system outputs the prediction for this frame. Our model in Fig. 6.4 (c) is able to respond within 0.16s on one single Titan X GPU. Further, our method can maintain similar mAP results even when the striding distance of the input window is increased to 8 frames, thus allowing real-time implementations.

6.7 Summary

In this chapter, we have proposed a novel Online Detection of Action Start task in a practical setting involving untrimmed, unconstrained videos. The goal is to detect the start of an action instance, with high categorization accuracy and low detection latency. We have designed three novel training methods for training effective ODAS models in detecting action timely and accurately: (1) hard negative samples generation based on GAN to distinguish ambiguous background, (2) explicitly modeling the temporal consistency between data around action start and data succeeding action start, and (3) adaptive sampling strategy to handle the scarcity of training data. Extensive experiments show that our proposed methods lead to significant performance gains and improve the state-of-the-art methods. An ablation study confirms the effectiveness and necessity of each proposed method. In the future, since the developed methods can address the specific challenges in detecting action starts, it would be interesting to further apply them to help address other online action detection tasks such as online per-frame labeling and online temporal action localization.

Part III

Action Understanding in the Compressed Domain

Chapter 7

Studying the Effects of Video Encoding Variations

7.1 Introduction

In real applications, videos are usually compressed by various video codecs such as MPEG-4 [162] for the sake of saving storage and accelerating transmission. In the previous chapters, we follow the mainstream action understanding methods to process the compressed video by first decoding it into a sequence of RGB image frames. However, these frames usually contain similar visual contents, resulting into redundant information to be processed. In addition, the state-of-the-art two-stream methods further extract optical flow data from RGB images for modeling motion. Consequently, this methodology of first decoding compressed video is quite time-consuming. As shown

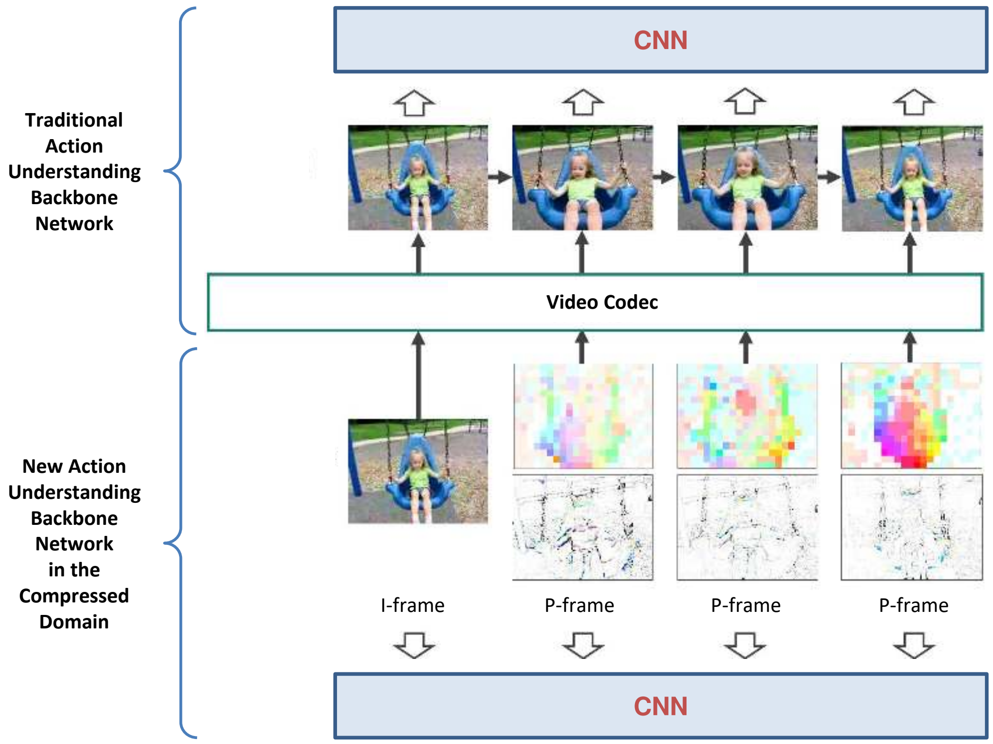

CNN
Traditional Action Understanding Backbone Network
Video Codec
New Action Understanding Backbone Network in the Compressed Domain
I-frame
P-frame
P-frame
P-frame
CNN

Figure 7.1: Comparisons between the conventional action understanding backbone networks in the decoded domain and the recently proposed action understanding backbone network in the compressed domain. Traditional backbone network first decodes the video and then feeds the RGB frames into a CNN. The new backbone network in the compressed domain operates on the compressed video directly.

in Figure 7.1, recently a new method called CoViAR [3] was proposed to directly perform action understanding in the compressed domain without decoding the compressed video. The CoViAR method exploits all data modalities in the compressed video, *i.e.* RGB I-frames, motion vectors and residuals to bypass RGB frame decoding. As shown in Figure 7.2, CoViAR achieves promising accuracy on the video-level action classification task and exhibits much faster speed than the traditional methods based on the decoded image frames, including a two-stream network (Temporal

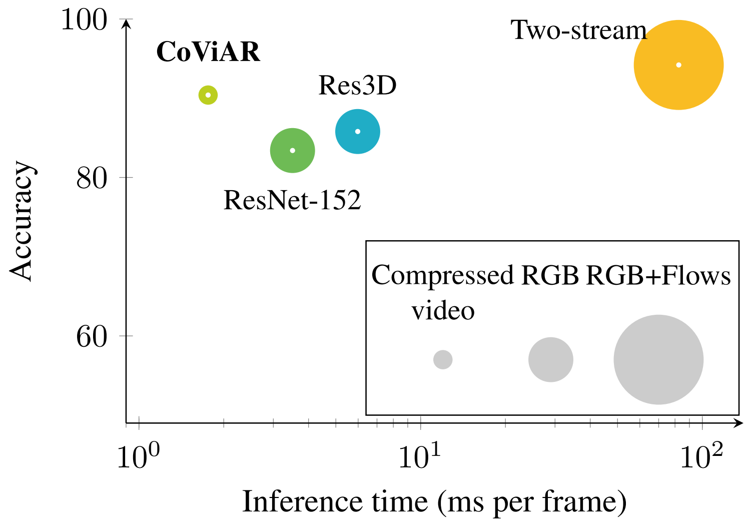


Figure 7.2: Per-frame processing speed and action classification accuracy on UCF-101. Node size denotes the input data size. Compared to three decoded images based methods, the new CoViAR method based on the compressed videos is both accurate and efficient. x-axis is in logarithmic scale.

Segment Network [163]), Res3D [16], and ResNet-152 [12]. Such a backbone network operating in the compressed domain can be integrated with any frameworks designed for addressing specific action understanding tasks such as those developed in Part I and Part II.

The CoViAR method requires that the training videos and testing videos are encoded by the same video codec with the same configurations. In reality, input videos could be compressed by different video codecs with different encoding configurations. This issue of encoding variation also exists in video transmission and storage. As shown in Figure 7.3, a practical strategy commonly used by real applications is transcoding the input videos of different formats into the same format for transmission and storage. Based on the transcoded videos of the same format (CoViAR uses

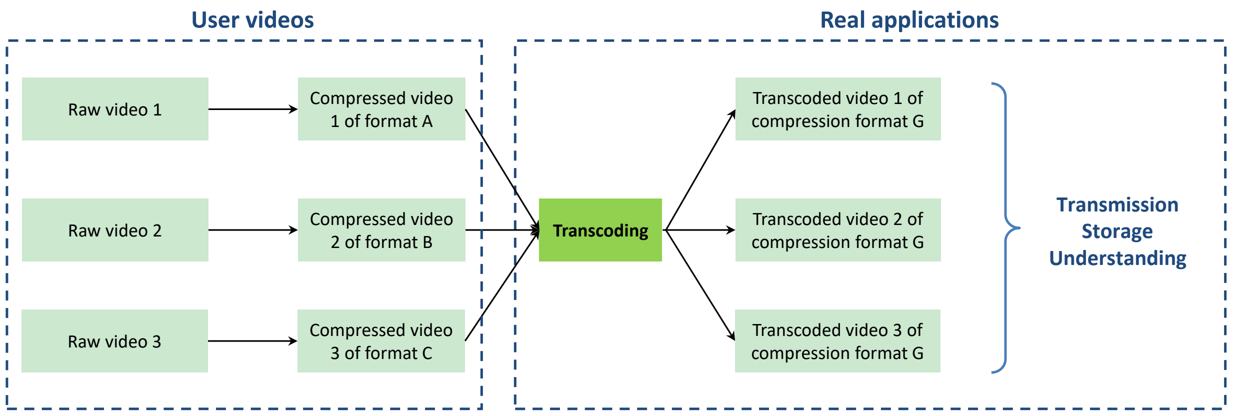

User videos
Real applications
Raw video 1
Raw video 2
Raw video 3
Compressed video 1 of format A
Compressed video 2 of format B
Compressed video 3 of format C
Transcoding
Transcoded video 1 of compression format G
Transcoded video 2 of compression format G
Transcoded video 3 of compression format G
Transmission
Storage
Understanding

Figure 7.3: The data flow of the video coding procedure in the realistic scenario: given raw videos, different users or their video capturing devices employ different formats for video compression and then send the compressed videos to the downstream real applications, which usually first transcode input videos of different formats into a preselected, generic format G and then perform transmission, storage or semantic understanding based on the format G.

MPEG-4 with the number of Group Of Pictures (GOP) set to 12 and Macroblock Size set to 16x16), we can perform action understanding using our models trained with videos of the same format.

However, the video encoding variation effect still remains unresolved. Because when compressing a raw video using different formats, despite being transcoded into the same format afterwards, the information loss and generated artifacts would be different. In this chapter, we aim to investigate training compressed video action understanding models like CoViAR to be robust to the variations of such information loss and artifacts caused by the video encoding variations.

To completely address the issue of video encoding variations, a straightforward solution is that we train individual models for different videos formats and for each testing video we apply the corresponding model trained with the videos transcoded from the same format. But this is impractical in reality because it is extremely inefficient that for each task or dataset we train a handful of models for the sake of covering all kinds of encoding formats. Alternatively, we can augment the training data to cover different kinds of encoding formats and training a generic model based on the augmented, diverse training dataset. Experimentally we find that such a generic model can indeed improve the baseline of training without data augmentation while it still performs not as good as training individual models. Finally, we conduct comprehensive ablation studies in order to understand how the variation of each compression factor, such as codec type, macroblock size and GOP size, affects the action understanding performance. We find that when training a generic model using videos transcoded from a certain format such as small macroblock size (*i.e.* 4x4) or large GOP size (*i.e.* 18), we can obtain performance comparable to training individual methods.

In the rest of this chapter, we review the compressed data modalities in Section 7.2, motivate the problem and summarize the possible encoding variations in Section 7.3, describe several approaches in Section 7.4, present and discuss the experimental results in Section 7.5, and finally draw our summary in Section 7.6.

7.2 Preliminaries

7.2.1 Data Modalities in the Compressed Domain

Prevailing video compression standards employs the Group Of Pictures (GOP) structure to encode the raw video into successive, non-overlapping GOPs. Frames or pictures within one GOP are compressed together. Each GOP begins with an I-frame (intra coded frame) whose RGB pixel values are stored. I-frame can be decoded independently with other frames.

The rest of frames within a GOP are P-frame (predictive coded frame) and/or B-frame (bi-predictive coded frame), containing motion-compensated difference information relative to the previously decoded frames. Each P-frame can only reference one frame which could be either I-frame or P-frame while each B frame can only reference two frames. In this thesis, we follow [3] to focus on the low-latency scenario which only involves P-frame without B-frame. Each P-frame stores motion vectors and residual errors: during encoding, the video codec divides a P-frame into macroblocks of size such as 16x16 and find the most similar image patch in the reference frame for each macroblock; the displacement between a macroblock in P-frame and its most similar image patch in the reference frame is regarded as the corresponding motion vector, which will be used in motion compensation during decoding; the pixel differences between a macroblock in P-frame and its most similar image patch in the reference frame are denoted as residual errors. During the decoding of a P-frame, the video codec performs motion compensation which effectively warps the reference frame using the motion vectors and then adds the residual errors to the motion-compensated reference frame to reconstruct the P-frame.

Consequently, three data modalities in the compression domain are suitable for action understanding: (1) RGB values of I-frame; (2) motion vectors and (3) residual errors of P-frame.

7.3 Problem Statement

7.3.1 Motivation

As shown in Figure 7.3, we follow the convention for video transmission and storage in real applications to perform action understanding based on transcoded videos of the same format. We follow the CoViAR [3], which is the only currently published method for action understanding in the compressed domain, to train models based on MPEG-4 [162] videos of GOP size 12 and macroblock size 16x16 (denoted as format G in this chapter). Thus, in Figure 7.3 we transcode videos of different formats to format G during the testing time using the FFmpeg tool [164]. Although the transcoded videos can be processed by the model trained on the videos of format G directly compressed from raw videos, due to the different information loss and artifacts generated during the process when the raw videos are compressed by different video codecs at the user end in Figure 7.3, the trained model achieves inferior accuracy compared to the individual models trained using the method presented in Section 7.4.1 (58.54% vs. 59.35% on the HMDB-51 video-level action classification task [165]), motivating us to handle such encoding variations at the user end. In this chapter, we aim to develop model training methods that are robust to such encoding variations so as to close the above performance gap.

7.3.2 Possible Encoding Variations

In this section, we discuss the possible encoding variations that are important in video compression. According to the encoding and decoding procedures introduced in the above Section 7.2.1, there are several major encoding configurations that are variable: **GOP size** is the number of frames in one GOP. The default value for MPEG-4 is 12 in FFmpeg. In this chapter, we study possible values of 6, 12, and 18. **Macroblock size** is the size of each macroblock in P-frame. The default value for MPEG-4 in FFmpeg is 16x16, which is a bit large. In this chapter, we study possible values of 4x4, 8x8, and 16x16. In addition, besides MPEG-4, we explore other possible **video codecs** including H.264 and H.265 which have same three data modalities and similar coding procedures with optimized details compared to MPEG-4. For H.264 and H.265, we use their default settings in FFmpeg which have adaptive GOP size and adaptive macroblock size.

During experiments as described in Section 7.5.2.1, we compress the raw videos using different compression formats to simulate the testing scenario in reality. We simulate the testing videos of the following formats: (1) MPEG-4 of GOP size 12 and macroblock size 16x16 (the format G); (2) MPEG-4 of GOP size 18 and macroblock size 16x16; (3) MPEG-4 of GOP size 6 and macroblock size 16x16; (4) MPEG-4 of GOP size 12 and macroblock size 4x4; (5) MPEG-4 of GOP size 12 and macroblock size 8x8; (6) H.264 of adaptive GOP size and adaptive macroblock size; (7) H.265 of adaptive GOP size and adaptive macroblock size.

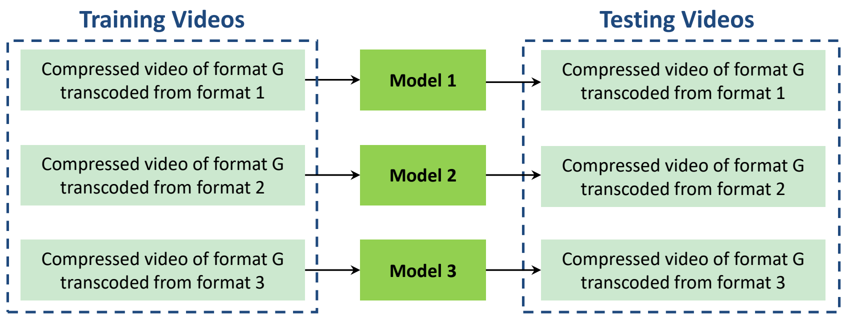
7.4 Approach

In order to improve the baseline method of training on videos of the format G compressed directly from raw videos following the CoViAR paper [3], we propose three training methods to handle the encoding variation issue in the following.

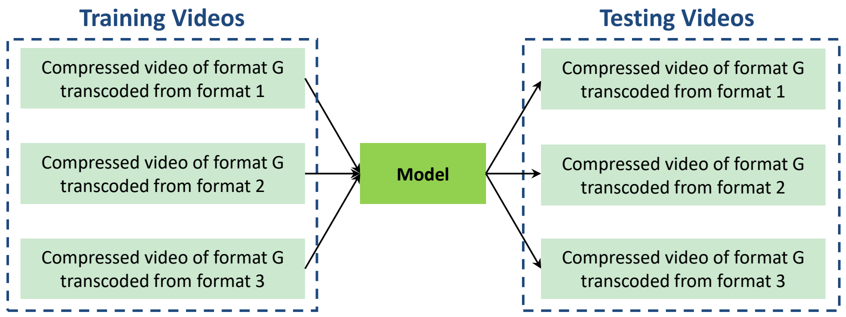
7.4.1 Training Individual Models

One straightforward solution that can bypass the encoding variation problem is training individual model for each format. Concretely, for each possible video compression format, we train a specific action understanding model independently with other formats as shown in Figure 7.4a. During testing, the model used for inference is set to be the one trained with videos transcoded from the same format as the input testing video. This solution, denoted as **training individual models**, is adhoc and very inefficient because it requires training one model for each compression format while during testing there are many possible formats.

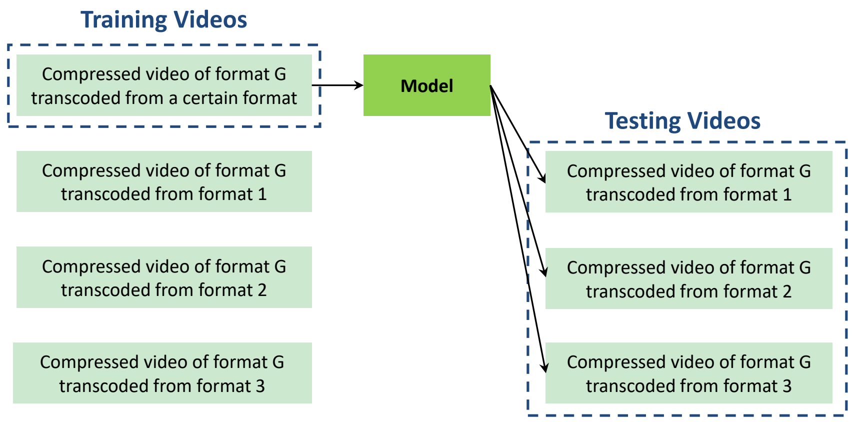
However, since a specific model is trained using videos transcoded from the same format, this method can bypass the encoding variation issue to obtain highly competitive accuracy. As shown in Section 7.5.3, this method achieves 59.35% overall accuracy on the HMDB-51 dataset, which can be regarded as our target for improving the baseline method of training on videos of the format G compressed directly from raw videos.


Training Videos
Testing Videos
Compressed video of format G transcoded from format 1
Compressed video of format G transcoded from format 2
Compressed video of format G transcoded from format 3
Model 1
Model 2
Model 3

(a) Training individual models for every formats


Training Videos
Testing Videos
Compressed video of format G transcoded from format 1
Compressed video of format G transcoded from format 2
Compressed video of format G transcoded from format 3
Model

(b) Training a generic model via data augmentation


Training Videos
Compressed video of format G transcoded from a certain format
Model
Testing Videos
Compressed video of format G transcoded from format 1
Compressed video of format G transcoded from format 2
Compressed video of format G transcoded from format 3

(c) Training a generic model using videos transcoded from a certain format

Figure 7.4: Illustrations of three approaches studied in this chapter. Videos used during training and testing are respectively highlighted by the corresponding dashed blue rectangles. (a) Training individual models for every formats. (b) Training a generic model using the augmented dataset consisting of videos of all possible formats. (c) Training a generic model using videos transcoded from a certain format. Experimentally we find that (c) achieves accuracy comparable to (a).

7.4.2 Training A Generic Model via Data Augmentation

In practice, ideally we should train and store only one generic model that will be applied to the testing videos transcoded from different formats. Data augmentation has turned out to be effective in addressing the scale variation in developing robust deep learning methods for image classification and action recognition. Therefore, we are motivated to train a generic model assisted by data augmentation: as shown in Figure 7.4a, during training we compress each raw video into different common formats such as those listed in Section 7.3.2 and then transcode them into the format G and finally feed them together for training one single model, which is applied to the testing videos transcoded from different formats.

As shown in Section 7.5.3, this data augmentation method leads to some performance gain while still performs not as good as training individual models (58.72% vs. 59.35%).

7.4.3 Training A Generic Model Using Videos Transcoded from A Certain Format

The performance gap between training a generic model via data augmentation and training individual models motivates us to further investigate how the variation of each configuration during the compression influences the final performance of action understanding. Similar to the baseline method which directly follows CoViAR to train one generic model using the compressed videos encoded from the raw format to the format G, we propose to, as illustrated in Figure 7.4c, train one generic model only using the compressed videos encoded from the raw format to a certain format and then transcoded to the format G. In order to perform ablation study for each compression configuration, we set such a certain format to the format G with one of the configuration values modified.

	split1	split2	split3	avg
MPEG-4 of GOP size 12 and macroblock size 16x16 (The format G used in the baseline method)	60.46	57.54	57.62	58.54
MPEG-4 of GOP size 12 and macroblock size 4x4	61.28	58.83	58.20	59.44

Table 7.1: Video-level action classification accuracy (%) on HMDB-51 for training videos transcoded from different macroblock sizes.

	split1	split2	split3	avg
MPEG-4 of GOP size 12 and macroblock size 16x16 (the format G used in the baseline method)	60.46	57.54	57.62	58.54
MPEG-4 of GOP size 6 and macroblock size 16x16	61.63	57.65	58.25	59.18
MPEG-4 of GOP size 18 and macroblock size 16x16	61.21	57.53	59.49	59.42

Table 7.2: Video-level action classification accuracy (%) on HMDB-51 for training videos transcoded from different GOP sizes.

First, we take a closer look at the macroblock size. Rather than setting it to a large size (*i.e.* 16x16) as in the format G, we explore a smaller macroblock size which can preserve more detailed information in the motion vectors to allow more accurate motion compensation. Table 7.1 shows that when compressing the training videos before transcoding, changing the macroblock size from 16x16 to 4x4 can improve the accuracy significantly.

Second, we vary the GOP size to study the effect of its variation. The default value in the format G is 12. We further experiment smaller and larger GOP sizes. A smaller GOP size reduces the artifacts generated during motion compensation. But note that when extracting the motion vectors and residual errors of a P-frame to be fed into the classification CNNs, we follow the CoViAR method to trace all motion vectors back to the reference I-frame and accumulate the residual errors on the way. Thus a larger GOP size allows the modeling of longer dependencies in time. As shown in Table 7.2, setting the GOP size to a smaller value (*i.e.* 6) and a larger value (*i.e.* 18) both can improve the accuracy. Setting the GOP size to 18 is slightly better than 6.

Finally, according the comparisons in Table 7.3, we find that the model trained using videos transcoded from a certain formats, such as MPEG-4 of GOP size 12 and macroblock size 4x4 or MPEG-4 of GOP size 18 and macroblock size 16x16, is quite robust enough to match the performance of training individual models, successfully addressing the encoding variation issue.

7.5 Experiments

7.5.1 Datasets and Evaluation Protocol

We evaluate on the HMDB-51 dataset [165], which contains 6,766 videos from 51 action categories and has 3 public train/test splits. All videos have single action label out of 51 classes. Thus we evaluate top-1 video-level action classification accuracy.

7.5.2 Implementation Details

7.5.2.1 Experimental Setup

Given the raw testing videos, in order to simulate the realistic scenario that testing videos can have various formats, we compress all the raw videos to each of the formats listed in Section 7.3.2 respectively. For each format we apply the trained model to obtain the overall accuracy and finally we compute the average accuracy for all formats.

7.5.2.2 Training

Following CoViAR [3], we employ a ResNet-152 classifier for I-frame and ResNet-18 classifiers for processing motion vectors and residual errors respectively. All videos are resized to 340x256 and are randomly cropped to 224x224 during each training batch. Learning rate starts from 0.001 and is divided by 10 when the accuracy plateaus. The batch size is set to 40. For each compressed data modality, we obtain the video-level prediction by averaging the predictions of three frames that are randomly sampled out of the whole video. The final prediction of the video-level label is obtained by fusing the predictions made by all three compressed data modalities.

7.5.2.3 Testing

During testing, we follow CoViAR [3] to obtain the video-level prediction for each testing video by averaging the predictions of 25 uniformly sampled frames. Instead of the random cropping, the center cropping and corner cropping are used. Specifically, as shown in Figure 7.3, given a testing video, we first transcode it into the format G (MPEG-4 of GOP size 12 and macroblock size 16x16) used in CoViAR [3] and then apply our trained model.

7.5.3 Comparisons among Different Approaches

	split1	split2	split3	avg
Training a generic model directly following CoViAR (The baseline method)	60.46	57.54	57.62	58.54
Training individual models	61.46	58.11	58.46	59.35
Training a generic model via data augmentation	59.79	57.72	58.64	58.72
Training a generic model using videos transcoded from a certain format (MPEG-4 of GOP size 12 and macroblock size 4x4)	61.28	58.83	58.2	59.44
Training a generic model using videos transcoded from a certain format (MPEG-4 of GOP size 18 and macroblock size 16x16)	61.21	57.53	59.49	59.42

Table 7.3: Video-level action classification accuracy (%) on HMDB-51 for comparisons among various methods for handle the encoding variation issue.

In Table 7.3, we compare the baseline method of directly following CoViAR and other three methods presented in Section 7.4:

(1) The baseline method is training a generic model using the videos compressed from the raw format to the common format G which is MPEG-4 of GOP size 12 and macroblock size 16x16. This method directly follows the CoViAR method and achieves low accuracy because it is sensitive to the video encoding variations.

(2) The method of training individual models trains a specific model dedicated to processing the testing video of each format, bypassing the issue of video encoding variations. Thus its accuracy

can be regarded as our target for improving the baseline method. But this method is not practical for real applications due to the need of training and storing a lot of models corresponding to all possible formats.

(3) Training a generic model via data augmentation achieves better accuracy than the baseline method while still performs not as good as training individual models.

(4) Training a generic model using videos transcoded from a certain format, such as MPEG-4 of GOP size 12 and macroblock size 4x4 or MPEG-4 of GOP size 18 and macroblock size 16x16, achieves performance comparable to training individual models, indicating that the trained model is robust enough already to handle the video encoding variation issue.

7.6 Summary

In this chapter, we focus on the backbone network for action understanding in the compressed domain, which can achieve a better trade-off between accuracy and efficiency compared to the traditional decoded video based methods as demonstrated in the previous work. However, the input videos during testing in practice are usually compressed by different video codecs with different encoding configurations. Following the common practice used for video transmission and storage, we first transcode the input videos of different formats into a preselected format and then training and testing action understanding models using the preselected format only.

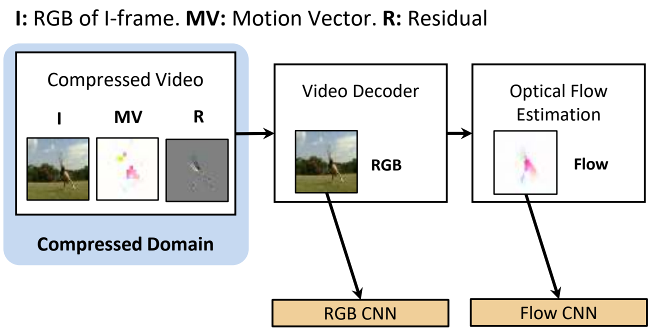
But the problem of video encoding variation still remains unresolved because the information loss and generated artifacts would be different when the raw video is compressed using different formats before transcoded into the preselected format. Thus we investigate how such different information loss caused by the video encoding variation issue would affect the performance of action understand. In order to address this encoding variation issue, we propose and compare three solutions: (1) training individual models that each model corresponds to one specific format, (2) training a generic model using the augmented dataset consisting of videos transcoded from all possible formats, and (3) training a generic model using videos transcoded from a certain format. Experimentally we find that training a generic model using videos transcoded from a certain format such as small macroblock size (*i.e.* 4x4, accuracy 59.44% on HMDB-51) or large GOP size (*i.e.* 18, accuracy 59.42% on HMDB-51), can obtain performance comparable to training individual methods (accuracy 59.35% on HMDB-51). This experimental finding indicates that a generic model trained using videos transcoded from a certain format is robust enough to the different information loss and thus this simple method can successfully address the encoding variation issue for action understanding in the compressed domain.

Chapter 8

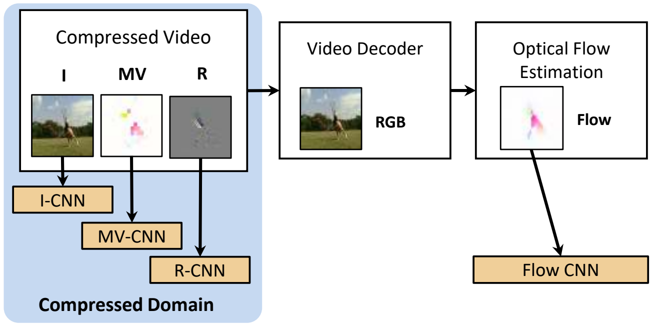
Learning to Generate Discriminative Motion Cues

8.1 Introduction

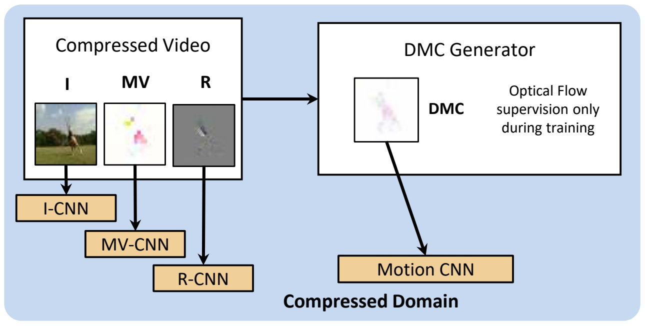
Video is a rich source of visual content as it not only contains appearance information in individual frames, but also temporal motion information across consecutive frames. Previous work has shown that modeling motion is important to various action understanding tasks, such as action recognition [34; 2; 166] and action localization [167; 168; 67; 169; 170; 90; 171]. Currently, methods achieving state-of-the-art results usually follow the two-stream network framework [34; 137; 33], which consists of two Convolutional Neural Networks (CNNs), one for the decoded RGB images and one for optical flow, as shown in Figure 8.1a. These networks can operate on either sin-



(a) Two-stream network (TSN)



(b) CoViAR + Flow



(c) DMC-Net (ours)

Figure 8.1: Illustrations of (a) the two-stream network, (b) the recent CoViAR method that achieves high accuracy via fusing compressed video data and optical flow, and (c) our proposed DMC-Net. Unlike *CoViAR+Flow* that requires video decoding of RGB images and flow estimation, our DMC-Net operates exclusively in the compressed domain at inference time while using optical flow to learn to capture discriminative motion cues at training time.

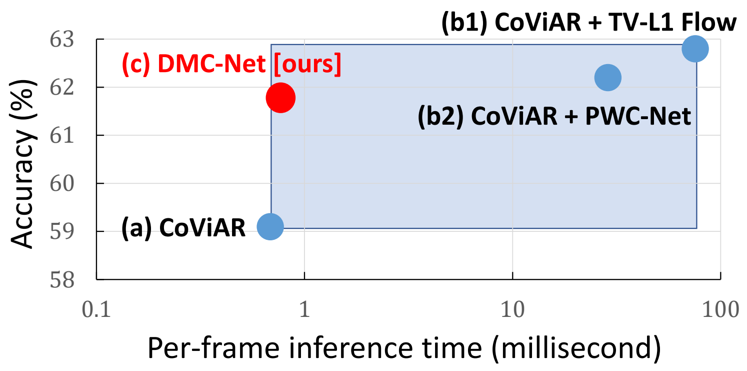


Figure 8.2: Comparing inference time and accuracy for different methods on HMDB-51. **(a)** Compressed video based method *CoViAR* is very fast. **(b)** But in order to reach high accuracy, *CoViAR* has to follow two-stream networks to add the costly optical flow computation, either using TV-L1 or PWC-Net. **(c)** The proposed *DMC-Net* not only operates exclusively in the compressed domain, but also is able to achieve high accuracy while being two orders of magnitude faster than methods that use optical flow. The blue box denotes the improvement room from *CoViAR* to *CoViAR + TV-L1 Flow*; x-axis is in logarithmic scale.

gle frames (2D inputs) or clips (3D inputs) and may utilize 3D spatio-temporal convolutions [15; 33].

Extracting optical flow, however, is very slow and often dominates the overall processing time of video analysis tasks. Recent work [3; 172; 173] avoids optical flow computation by exploiting the motion information from compressed videos encoded by standards like MPEG-4 [162]. Such methods utilize the motion vectors and residuals already present in the compressed video to model motion. The recently proposed CoViAR [3] method, for example, contains three independent CNNs operating over three modalities in the compressed video, *i.e.* RGB image of I-frame (**I**), low-resolution Motion Vector (**MV**) and Residual (**R**). The predictions from individual CNNs are combined by late fusion. CoViAR runs extremely fast while modeling motion features (see Figure 8.1b). However, in order to achieve state-of-the-art accuracy, late fusion with optical flow is further needed (see Figure 8.2).

This performance gap is due to the motion vector being less informative and discriminative than flow. First, the spatial resolution of the motion vector is substantially reduced (*i.e.* 16x) during video encoding, and fine motion details, which are important to discriminate actions, are permanently lost. Second, employing two CNNs to process motion vectors and residuals separately ignores the strong interaction between them. Because the residual is computed as the difference between the raw RGB image and its reference frame warped by the motion vector. The residual is often well-aligned with the boundary of moving object, which is more important than the motion at other locations for action recognition according to [174]. Jointly modeling motion vectors and residuals, which can be viewed as coarse-scale and fine-scale motion feature respectively, can exploit the encoded motion information more effectively.

To address those issues, we propose a novel approach to learn to generate a **Discriminative Motion Cue (DMC)** representation by refining the noisy and coarse motion vectors. We develop a lightweight DMC generator network that operates on stacked motion vectors and residuals. This generator requires training signals from different sources to capture discriminative motion cues and incorporate high-level recognition knowledge. In particular, since flow contains high resolution and accurate motion information, we encourage the generated DMC to resemble optical flow by a pixel-level reconstruction loss. We also use an adversarial loss [21] to approximate the distribution of optical flow. Finally, the DMC generator is also supervised by the downstream action recognition classifier in an end-to-end manner, allowing it to learn motion cues that are discriminative for recognition.

During inference, the DMC generator is extremely efficient with merely *0.23* GFLOPs, and takes only *0.106 ms* per frame which is negligible compared with the time cost of using flow. In Figure 8.1c, we call our full model *DMC-Net*. Although optical flow is required during training, our method operates *exclusively in the compressed domain* at inference time and runs two orders of magnitude faster than methods using optical flow, as shown in Figure 8.2.

In the rest of this chapter, we review the related work in Section 8.2, describe the proposed DMC-Net in Section 8.3, present the experimental results in Section 8.4, and finally draw our summary in Section 8.5.

8.2 Related Work

8.2.1 Motion in the Compressed Video Action Recognition

Recently, a number of approaches that utilize the information present in the compressed video domain have been proposed for video-level action classification. In the pioneering works [173; 172], Zhang *et al.* replace the optical flow stream in two-stream methods by a motion vector stream, but it still needed to decode RGB image for P-frame and ignored other motion-encoding modalities in compressed videos such as the residual maps. The CoViAR method [3] proposed to exploit all data modalities in compressed videos, *i.e.* RGB I-frames, motion vectors and residuals to bypass RGB frame decoding. However, CoViAR fails to achieve performance comparable to that of two-stream methods, mainly due to the low-resolution of the motion vectors and the fact that motion vectors and residuals, although highly related, are processed by independent networks. We argue that, when properly exploited, the compressed video modalities have enough signal to allow us to capture more discriminative motion representation. We therefore explicitly learn such representation as opposed to relying on optical flow during inference.

8.2.2 Motion Representation and Optical Flow Estimation

Traditional optical flow estimation methods explicitly model the displacement at each pixel between successive frames [175; 176; 177; 178]. In the last years CNNs have successfully been trained to estimate the optical flow, including FlowNet [179; 180], SpyNet [181] and PWC-Net [182], and achieve low End-Point Error (EPE) on challenging benchmarks, such as MPI Sintel [183] and KITTI 2015 [184]. The Im2Flow work [185] also shows optical flow can be hallucinated from still images. Recent work however, shows that accuracy of optical flow does not strongly correlate with accuracy of video recognition [186]. Thus, motion representation learning methods focus more on generating discriminative motion cues. Fan *et al.* [187] proposed to transform TV-L1 optical flow algorithm into a trainable sub-network, which can be jointly trained with downstream recognition network. Ng *et al.* [188] employs fully convolutional ResNet model to generate pixel-wise prediction of optical flow, and can be jointly trained with recognition network. Unlike optical flow estimation methods, our method does not aim to reduce EPE error. Also different from all above methods of motion representation learning which take decoded RGB frames as input, our method refines motion vectors in the compressed domain, and requires much less model capacity to generate discriminative motion cues.

8.3 Approach

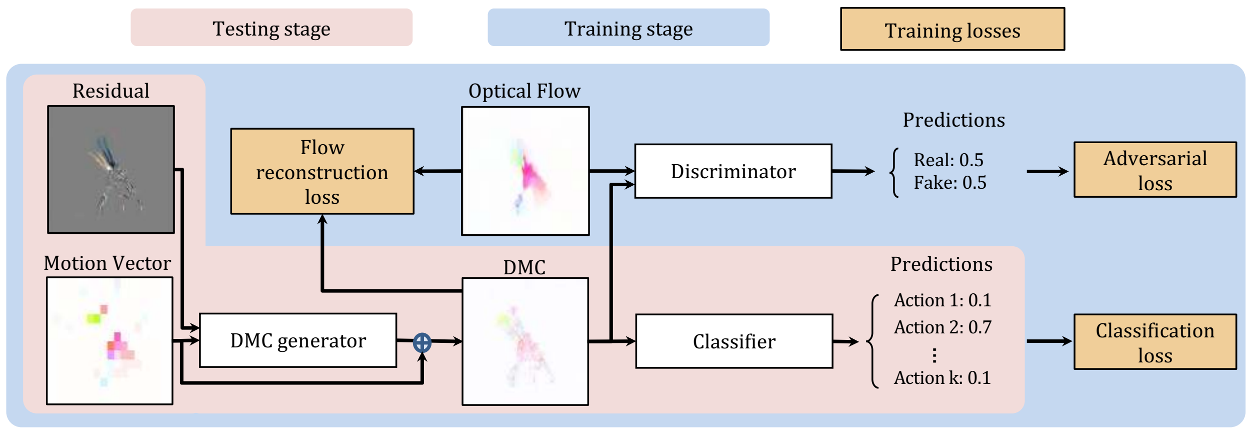


Figure 8.3: The framework of our Discriminative Motion Cue Network (DMC-Net). Given the stacked residual and motion vector as input, the DMC generator reduces noise in the motion vector and captures more fine motion details, outputting a more discriminative motion cue representation which is used by a small classification network to classify actions. In the training stage, we train the DMC generator and the action classifier jointly using three losses. In the test stage, only the modules highlighted in pink are used.

In this section, we present our approach for generating *Discriminative Motion Cues (**DMC**)* from compressed video. The overall framework of our proposed **DMC-Net** is illustrated in Figure 8.3. In Section 8.3.1, we introduce the basics of compressed video and the notations we use. Then we design the DMC generator network in Section 8.3.2. Finally we present the training objectives in Section 8.3.3 and discuss inference in Section 8.3.4.

8.3.1 Basics and Notations of Compressed Video

We follow CoViAR [3] and use MPEG-4 Part2 [162] encoded videos where every I-frame is followed by 11 consecutive P-frames. Three data modalities are readily available in MPEG-4 compressed video: (1) RGB image of I-frame (**I**); (2) Motion Vector (**MV**) records the displacement of each macroblock in a P-frame to its reference frame and typically a frame is divided into 16x16 macroblocks during video compression; (3) Residual (**R**) stores the RGB difference between a P-frame and its reference I-frame after motion compensation based on MV. For a frame of height H and width W, I and R have shape (3, H, W) and MV has shape (2, H, W). But note that MV has much lower resolution in effect because its values within the same macroblock are identical.

8.3.2 The Discriminative Motion Cue Generator

8.3.2.1 Input of the Generator

Existing compressed video based methods directly feed motion vectors into a classifier to model motion information. This strategy is not effective in modeling motion due to the characteristics of MV: (1) MV is computed based on simple block matching, making MV noisy and (2) MV has substantially lower resolution, making MV lacking fine motion details. In order to specifically handle these characteristics of MV, we aim to design a lightweight generation network to reduce noise in MV and capture more fine motion details, outputting DMC as a more discriminative motion representation.

Network Architecture	GFLOPs
C3D [15]	38.5
Res3D-18 [16]	19.3
ResNet-152 [12]	11.3
ResNet-18 [12]	1.78
DMC generator (PWC-Net [182])	36.15
DMC generator [ours]	0.23

Table 8.1: Computational complexity of different networks. Input has height 224 and width 224.

Layer	Input size	Output size	Filter config
`conv0`	5, 224, 224	8, 224, 224	8, 3x3, 1, 1
`conv1`	13, 224, 224	8, 224, 224	8, 3x3, 1, 1
`conv2`	21, 224, 224	6, 224, 224	6, 3x3, 1, 1
`conv3`	27, 224, 224	4, 224, 224	4, 3x3, 1, 1
`conv4`	31, 224, 224	2, 224, 224	2, 3x3, 1, 1
`conv5`	33, 224, 224	2, 224, 224	2, 3x3, 1, 1

Table 8.2: The architecture of our Discriminative Motion Cue (DMC) generator network which takes stacked motion vector and residual as input. Input/output size follows the format of #channels, height, width. Filter configuration follows the format of #filters, kernel size, stride, padding.

To accomplish this goal, MV alone may not be sufficient. According to [174], the motion nearby object boundary is more important than the motion at other locations for action recognition. We also notice R is often well-aligned with the boundary of moving objects. Moreover, R is strongly correlated with MV as it is computed as the difference between the original frame and its reference I-frame compensated using MV. Therefore, we propose to stack MV and R as input into the DMC generator, as shown in Figure 8.3. This allows utilizing the motion information in MV and R as well as the correlation between them, which cannot be modeled by separate CNNs as in the current compressed video works [3; 172; 173].

8.3.2.2 Generator Network Architecture

. Quite a few deep generation networks have been proposed for optical flow estimation from RGB images. One of these works is PWC-Net [182], which achieves SoTA performance in terms of both End Point Error (EPE) and inference speed. We therefore choose to base our generator design principles on the ones used by PWC-Net. It is worth noting that PWC-Net takes decoded RGB frames as input unlike our proposed method operating only in the compressed domain.

Directly adopting the network architecture of the flow estimator network in PWC-Net for our DMC generator leads to high GFLOPs as indicated in Table 8.1. To achieve high efficiency, we have conducted detailed architecture search experimentally to reduce the number of filters in each convolutional layer of the flow estimator network in PWC-Net, achieving the balance between accuracy and complexity. Furthermore, since our goal is to refine MV, we propose to add a shortcut connection between the input MV and the output DMC, making the generator to directly predict the refinements which are added on MV to obtain DMC.

Table 8.2 shows the network architecture of our DMC generator: 6 convolutional layers are stacked sequentially with all convolutional layers densely connected [13]. Every convolutional filter has a 3x3 kernel with stride 1 and padding 1. Each convolutional layer except `conv5` is followed by a Leaky ReLU [189] layer, where the negative slope is 0.1.

As shown in Table 8.1, our DMC generator only requires 0.63% GFLOPs used by the flow estimator in PWC-Net if it were adopted to implement our DMC generator. Also, Table 8.1 compares our DMC generator with other popular network architectures for video analysis including frame-level models (ResNet-18 and ResNet-152 [12]) and clip-level models (C3D [15] and Res3D [16]). We observe that the complexity of DMC generator is orders of magnitude smaller compared to that of other architectures, which makes it running much faster. We have explored a strategy of using two consecutive networks to respectively rectify errors in MV and capture fine motion details while this did not achieve better accuracy.

8.3.3 Flow-guided, Discriminative Motion Cues

Compared to MV, optical flow exhibits more discriminative motion information because: (1) Unlike MV is computed using simple block matching, nowadays dense flow estimation is computed progressively from coarse scales to fine scales [190]. (2) Unlike MV is blocky and thus misses fine details, flow keeps the full resolution of the corresponding frame. Therefore we propose to guide the training of our DMC generator using optical flow. To this end, we have explored different ways and identified three effective training losses as shown in Figure 8.3 to be presented in the following: a flow reconstruction loss, an adversarial loss, and a downstream classification loss.

8.3.3.1 Optical Flow Reconstruction Loss

First, we minimize the per-pixel difference between the generated DMC and its corresponding optical flow. Following Im2Flow [185] which approximates flow from a single RGB image, we use the Mean Square Error (MSE) reconstruction loss $\mathcal{L}_{\mathrm{mse}}$ defined as:

$$\mathcal{L}_{\mathrm{mse}} = \mathbb{E}_{\boldsymbol{x}\sim p} \left\| \mathcal{G}_{\mathrm{DMC}}(\boldsymbol{x}) - \mathcal{G}_{\mathrm{OF}}(\boldsymbol{x}) \right\|_2^2 , \tag{8.1}$$

where p denotes the set of P-frames in the training videos, $\mathbb{E}$ stands for computing expectation, $\mathcal{G}_{\text{DMC}}(\boldsymbol{x})$ and $\mathcal{G}_{\text{OF}}(\boldsymbol{x})$[1] respectively denote the DMC and optical flow for the corresponding input frame $\boldsymbol{x}$ sampled from p. Since only some regions of flow contain discriminative motion cues that are important for action recognition, we have explored weighting the flow reconstruction loss to encourage attending to the salient regions of flow. But this strategy does not achieve better accuracy.

8.3.3.2 Adversarial Loss

As pointed out by previous works [26], the MSE loss implicitly assumes that the target data is drawn from a Gaussian distribution and therefore tends to generate smooth and blurry outputs. This in effect results in less sharp motion representations especially around boundaries, making the generated DMC less discriminative. Generative Adversarial Networks (GAN) [21] has been proposed to minimize the Jensen$-$Shannon divergence between the generative model and the true data distribution, making these two similar. Thus in order to help our DMC generator learn to approximate the distribution of optical flow data, we further introduce an adversarial loss. Note that unlike GAN which samples from random noise, adversarial loss samples from the input dataset, which already has large variability [26].

Let our DMC generator $\mathcal{G}_{\text{DMC}}$ be the **Generator** in the adversarial learning process. As shown in Figure 8.3, a **Discriminator** $\mathcal{D}$ is introduced to compete with $\mathcal{G}_{\text{DMC}}$. $\mathcal{D}$ is instantiated by a

[1]We relax the notational rigor and use $\mathcal{G}_{OF}(\boldsymbol{x})$ to refer to the optical flow corresponding to the frame $\boldsymbol{x}$, although for many optical flow algorithms the input would be a pair of frames.

binary classification network that takes as input either **real** optical flow or **fake** samples generated via our DMC generator. Then $\mathcal{D}$ outputs a two-dimensional vector that is passed through a softmax operation to obtain the probability $P_{\mathcal{D}}$ of the input being *Real*, *i.e.* flow versus *Fake*, *i.e.* DMC. $\mathcal{G}_{\text{DMC}}$ and $\mathcal{D}$ are trained in an alternating manner: $\mathcal{G}_{\text{DMC}}$ is fixed when $\mathcal{D}$ is being optimized, and vice versa.

During training $\mathcal{D}$, $\mathcal{G}_{\text{DMC}}$ is fixed and is only used for inference. $\mathcal{D}$ aims to classify the generated DMC as Fake and classify flow as Real. Thus the adversarial loss for training $\mathcal{D}$ is:

$$\begin{aligned}\mathcal{L}^{D}_{\text{adv}} =& \mathbb{E}_{\boldsymbol{x}\sim p}[-\log P_{\mathcal{D}}(\text{Fake}|\mathcal{G}_{\text{DMC}}(\boldsymbol{x})) \\ & - \log P_{\mathcal{D}}(\text{Real}|\mathcal{G}_{\text{OF}}(\boldsymbol{x}))],\end{aligned} \tag{8.2}$$

where p denotes the set of P-frames in the training set and $\mathcal{G}_{\text{DMC}}(\boldsymbol{x})$ and $\mathcal{G}_{\text{OF}}(\boldsymbol{x})$ respectively represent the DMC and optical flow for each input P-frame $\boldsymbol{x}$.

During training $\mathcal{G}_{\text{DMC}}$, $\mathcal{D}$ is fixed. $\mathcal{G}_{\text{DMC}}$ is encouraged to generate DMC that is similar and indistinguishable with flow. Thus the adversarial loss for training $\mathcal{G}_{\text{DMC}}$ is:

$$\mathcal{L}^{G}_{\text{adv}} = \mathbb{E}_{\boldsymbol{x}\sim p}[-\log P_{\mathcal{D}}(\text{Real}|\mathcal{G}_{\text{DMC}}(\boldsymbol{x}))], \tag{8.3}$$

which can be trained jointly with the other losses designed for training the DMC generator in an end-to-end fashion, as presented in Section 8.3.3.3.

Through the adversarial training process, $\mathcal{G}_{\text{DMC}}$ learns to approximate the distribution of flow data, generating DMC with more fine details and thus being more similar to flow. Those fine details usually capture discriminative motion cues and are thus important for action recognition.

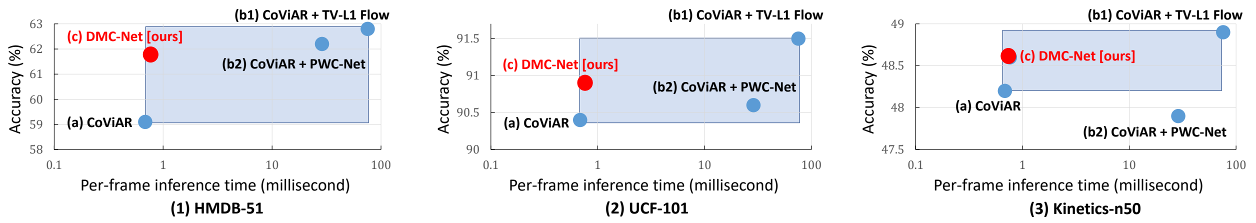


Figure 8.4: Accuracy vs. speed on 3 benchmarks. Results on UCF-101 and HMDB-51 are averaged over 3 splits. (b1) and (b2) use ResNet-18 to classify flow and (c) also uses ResNet-18 to classify DMC. The proposed *DMC-Net* not only operates exclusively in the compressed domain, but also is able to achieve higher accuracy than (a) while being two orders of magnitude faster than methods that use optical flow. The blue area indicates the improvement room from (a) to (b1).

8.3.3.3 The Full Training Objective Function

Semantic classification loss. As our final goal is to create motion representation that is discriminative with respect to the downstream action recognition task, it is important to train the generator jointly with the follow-up action classifier. We employ the softmax loss as our action classification loss, denoted as $\mathcal{L}_{\text{cls}}$.

The full training objective. Our whole model is trained with the aforementioned losses putting together in an end-to-end manner. The training process follows the alternating training procedure stated in Section 8.3.3.2. During training the discriminator, $\mathcal{D}$ is trained while the DMC generator $\mathcal{G}_{\text{DMC}}$ and the downstream action classifier are fixed. The full training objective is to minimize the adversarial loss $\mathcal{L}_{\text{adv}}^{D}$ in Equation 8.2. During training the generator $\mathcal{G}_{\text{DMC}}$, $\mathcal{D}$ is fixed while the DMC generator $\mathcal{G}_{\text{DMC}}$ and the downstream action classifier are trained jointly with the following full training objective to be minimized:

$$\mathcal{L}_{\text{cls}} + \alpha \cdot \mathcal{L}_{\text{mse}} + \lambda \cdot \mathcal{L}_{\text{adv}}^{G}, \tag{8.4}$$

where $\mathcal{L}_{\text{mse}}$ is given by Equation 8.1, $\mathcal{L}^{G}_{\text{adv}}$ is given by Equation 8.3, and α, λ are balancing weights.

8.3.4 Inference

As shown in Figure 8.3, despite having three losses jointly trained end-to-end, our DMC-Net is actually quite efficient during inference: basically first the generator outputs DMC and then the generated DMC is fed into the classification network to make action class prediction. We compare our inference speed with other methods in Section 8.4.4.

8.4 Experiments

In this section, we first detail our experimental setup, present quantitative analysis of our model, and finally compare with state-of-the-art methods.

8.4.1 Datasets and Evaluation

UCF-101 [64]. This dataset contains 13,320 videos from 101 action categories, along with 3 public train/test splits.

HMDB-51 [165]. This dataset contains 6,766 videos from 51 action categories, along with 3 public train/test splits.

Kinetics-n50. From the original Kinetics-400 dataset [137], we construct a subset referred as **Kinetics-n50** in this chapter. We keep all 400 categories. For each class, we randomly sample 30 videos from the original training set as our training videos and randomly sample 20 videos from the original validation set as our testing videos.

Evaluation protocol. All videos in the above datasets have single action label out of multiple classes. Thus we evaluate top-1 video-level class prediction accuracy.

8.4.2 Implementation Details

8.4.2.1 Training

For I, MV, and R, we follow the exactly same setting as used in CoViAR [3]. Note that I employs ResNet-152 classifier; MV and R use ResNet-18 classifier. To ensure efficiency, DMC-Net also uses ResNet-18 to classify DMC in this chapter unless we explicitly point out. To allow apple-to-apple comparisons between DMC and flow, we also choose frame-level ResNet-18 classifier as the flow CNN shown in Figure 8.1b. TV-L1 [176] is used for extracting optical flow to guide the training of our DMC-Net. All videos are resized to 340$\times$256. Random cropping of 224$\times$224 and random flipping are used for data augmentation.

We first train the DMC generator for 1 epoch using the flow reconstruction loss only with the classification network fixed. Then we include the classification loss to train both the generator and classifier end-to-end for 49 epochs. In the total loss indicated by Equation 8.4, we set α to 10 to balance weights. The overall learning rate is set to 0.01 and it is divided by 10 whenever the total training loss plateaus. All layers in the classification network except its last layer have the learning rate set to be 100x smaller. Then we use the above trained model as the initialization for training our whole model with all three losses including the adversarial loss. Our whole model consists of the generator, the classifier and the discriminator now. In the total loss indicated by Equation 8.4, we set α to 10 and set λ to 1. The overall learning rate is set to 0.01 and it is divided by 10 whenever the total training loss plateaus. All layers in the classification network except its last layer have the learning rate set to be 100x smaller. Based on the network architectures for the discriminator used in a popular GAN implementation repository [2], we experimented with various number of filters in each layer and various number of layers. Finally we identified a network architecture for implementing our discriminator which achieves accuracy comparable to more complicated architectures. This discriminator's architecture consists of a stack of 2D convolutional layers with a two-way Fully Connected layer at the end, as shown in the following Figure 8.5.

8.4.2.2 Testing

For I, MV, and R, we follow the exactly same setting as in CoViAR [3]: 25 frames are uniformly sampled for each video; each sampled frame has 5 crops augmented with flipping; all 250 ($25\times2\times5$) score predictions are averaged to obtain one video-level prediction. For DMC, we following the same setting except that we do not use cropping and flipping, which shows comparable accuracy but requires less computations. Finally, we follow CoViAR [3] to obtain the final prediction via fusing prediction scores from all modalities (*i.e.* I, MV, R, and DMC).

[2]https://github.com/eriklindernoren/PyTorch-GAN/tree/master/implementations

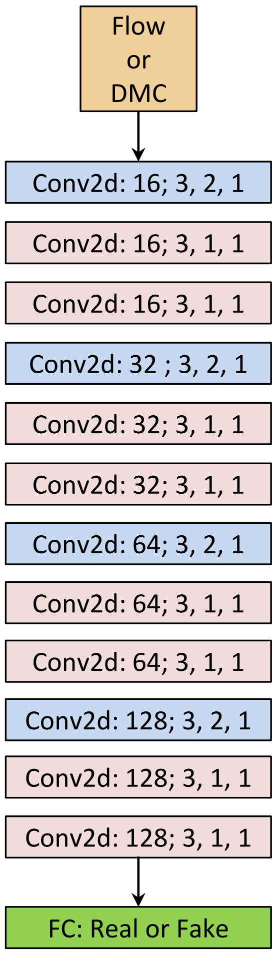


Figure 8.5: The network architecture of our discriminator. We denote each 2D convolutional layer in the format of #filters; kernel size, stride, padding.

8.4.3 Model Analysis

How much gain DMC-Net can improve over CoViAR? Figure 8.4 reports accuracy on all three datasets. **CoViAR + TV-L1** and **CoViAR + PWC-Net** follow two-stream methods to include an optical flow stream computed by TV-L1 [190] and PWC-Net [182] respectively. **CoViAR + TV-L1** can be regard as our upper bound for improving accuracy because TV-L1 flow is used to

		Two-Stream Method (RGB+Flow)		Compressed Video Based Methods	
		BN-Inception	ResNet152	CoViAR	DMC-Net [ours]
Time (ms)	Preprocess	75.0	75.0	0.46	0.46
	CNN (**S**)	1.6	7.5	0.59	0.89
	Total (**S**)	76.6	82.5	1.05	1.35
	CNN (**C**)	0.9	4.0	0.22	0.30
	Total (**C**)	75.9	79.0	0.68	0.76
FPS	CNN (**C**)	1111.1	250.0	4545.4	3333.3
	Total (**C**)	13.1	12.6	1470.5	1315.7

(a) DMC-Net vs. Two-stream methods and CoViAR

	Generator	Generator + Cls.
	Time (ms) / FPS	Time (ms) / FPS
Deepflow [191]	1449.2 / 0.7	1449.5 / 0.7
Flownet2.0 [180]	220.8 / 4.5	221.0 / 4.5
TVNet [187]	83.3 / 12.0	83.5 / 12.0
PWC-Net [182]	28.6 / 35.0	28.8 / 34.8
DMC-Net [ours]	**0.1 / 9433.9**	**0.3 / 3333.3**

(b) DMC-Net vs. flow estimation methods

Table 8.3: Comparisons of per-frame inference speed. **(a)** Comparing our DMC-Net to the two-stream methods and the CoViAR method. We consider two scenarios of forwarding multiple CNNs sequentially and concurrently, denoted by **S** and **C** respectively.. **(b)** Comparing our DMC-Net to deep network based optical flow estimation and motion representation learning methods. CNNs in DMC-Net are forwarded concurrently. All networks have batch size set to 1. For the classifier (denoted as Cls.), all methods use ResNet-18.

guide the training of **DMC-Net**. By only introducing a lightweight DMC generator, our **DMC-Net** significantly improves the accuracy of **CoViAR** to approach **CoViAR + Flow**. Figure 8.6 shows that the generated DMC has less noisy signals such as those in the background area and DMC captures fine and sharp details of motion boundary, leading to the accuracy gain over **CoViAR**.

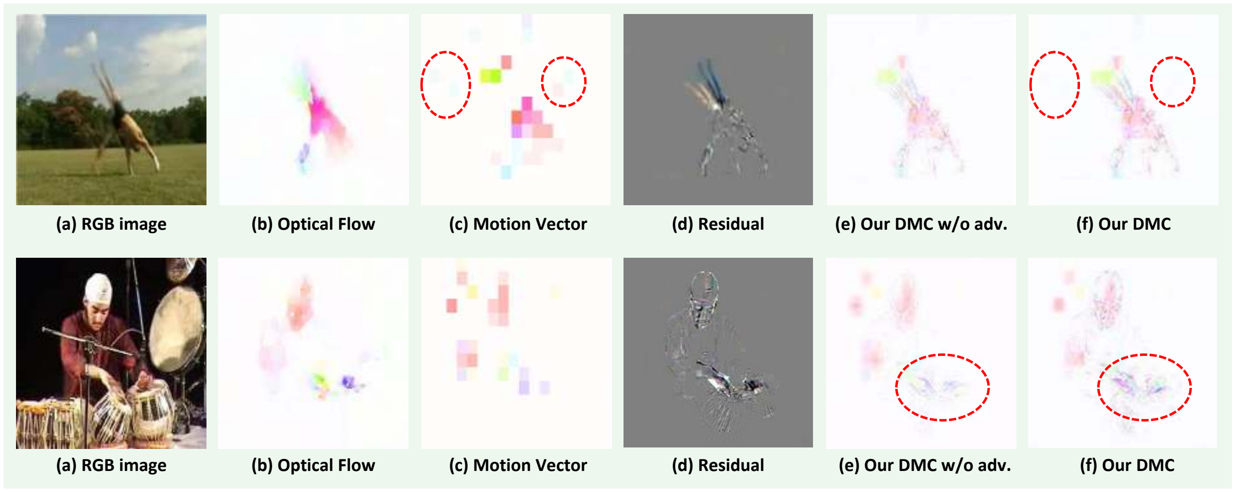


Figure 8.6: A Cartwheel example (top) and a PlayingTabla (bottom) example. All images in one row correspond to the same frame. For the Cartwheel example, these noisy blocks in the background (highlighted by two red circles) are reduced in our DMC. For the PlayingTabla example, our DMC exhibits sharper and more discriminative motion cues around hands (highlighted by the red circle) than our DMC w/o the adversarial loss during training. Better viewed in color.

How effectiveness is each proposed loss? As indicated by Figure 8.4 (1), on HMDB-51 the accuracy is 59.1% for **CoViAR** and 62.8% for **CoViAR + TV-L1 Flow** and 62.2% for **CoViAR + PWC-Net Flow**. For our DMC-Net, when only using the classification loss, the accuracy is 60.5%; when using the classification loss and the flow reconstruction loss, the accuracy is improved to 61.5%; when further including the adversarial training loss as presented in Section 8.3.3.3, DMC-Net eventually achieves 61.8% accuracy.

8.4.4 Inference Speed

Following [3], we measure the average per-frame running time, which consists of the time for data pre-processing and the time for CNN forward pass. For the CNN forward pass, both the scenarios

	HMDB-51	UCF-101
Compressed video based methods		
EMV-CNN [173]	51.2 (split1)	86.4
DTMV-CNN [172]	55.3	87.5
CoViAR [3]	59.1	90.4
DMC-Net (ResNet-18) [ours]	62.8	90.9
DMC-Net (I3D) [ours]	71.8	92.3
Decoded video based methods *(RGB only)*		
Frame-level classification		
ResNet-50 [12]	48.9	82.3
ResNet-152 [12]	46.7	83.4
Motion representation learning		
ActionFlowNet (2-frames) [188]	42.6	71.0
ActionFlowNet [188]	56.4	83.9
PWC-Net (ResNet-18) + CoViAR [182]	62.2	90.6
TVNet [187]	71.0	94.5
Spatio-temporal modeling		
C3D [15]	51.6	82.3
Res3D [16]	54.9	85.8
ARTNet [192]	70.9	94.3
MF-Net [193]	74.6	96.0
S3D [194]	75.9	96.8
I3D RGB [137]	74.8	95.6
I3D RGB + DMC-Net (I3D) [ours]	77.8	96.5
Decoded video based methods *(RGB + Flow)*		
Two-stream [34]	59.4	88.0
Two-Stream fusion [195]	65.4	92.5
I3D [137]	80.7	98.0
R(2+1)D [33]	78.7	97.3

Table 8.4: Accuracy averaged over all three splits on HMDB-51 and UCF-101 for both state-of-the-art compressed video based methods and decoded video based methods.

of forwarding multiple CNNs sequentially and concurrently are considered. Detailed results can be found in Table 8.3 (a). Results of **two-stream methods** are quoted from [3]. Due to the need of decoding compressed video into RGB frames and then computing optical flow, its pre-process takes much longer time than compressed video based methods. **DMC-Net** accepts the same inputs as **CoViAR** and thus **CoViAR** and **DMC-Net** have the same pre-processing time. As for the CNN forwarding time of compressed video based methods, we measure **CoViAR** and **DMC-Net** using the exactly same implementation as stated in Section 8.4.2 and the same experimental setup: we use one NVIDIA GeForce GTX 1080 Ti and set the batch size of each CNN to 1 while in practice the speed can be further improved to utilize larger batch size. Despite adding little computational overhead on CoViAR, DMC-Net is still significantly faster than the conventional **two-stream methods**.

Deepflow [191], **Flownet** [180] and **PWC-Net** [182] have been proposed to accelerate optical flow estimation by using deep networks. **TVNet** [187] was proposed to generate even better motion representation than flow with fast speed. Those estimated flow or generated motion representation can replace optical flow used in two-stream methods to go through a CNN for classification. We combine these methods with a ResNet-18 classifier in Table 8.3 (b). We can see that our DMC generator runs much faster than these state-of-the-art motion representation learning methods.

8.4.5 Comparisons with Compressed Video Methods

As shown in the top section of Table 8.4, **DMC-Net** outperforms all other methods that operate in the compressed video domain, *i.e.* **CoViAR** [3], **EMV-CNN** [173] and **DTMV-CNN** [172]. Our method outperforms methods like [173; 172] that the output of the MV classifier is trained to approximate the output of the optical flow classifier. We believe this is because of the fact that approximating the classification output directly is not ideal, as it does not explicitly address the issues that MV is noisy and low-resolutional. By generating a more discriminative motion representation DMC, we are able to get features that are highly discriminative for the downstream recognition task. Furthermore, our DMC-Net can be combined with these classification networks of high capacity and trained in an end-to-end manner. **DMC-Net (I3D)** replaces the classifier from ResNet-18 to I3D, achieving significantly higher accuracy and outperforming a number of methods that require video decoding. Note that the speed of **DMC-Net (I3D)** and the speed of **DMC-Net (ResNet-18)** are not directly comparable. ResNet-18 is a frame-level classifier: given an input frame, **DMC-Net (ResNet-18)** can classify it with the speed at 0.76ms. However, I3D is a clip-level classifier: during testing, we follow [137] to feed 250 frames concurrently into a I3D classifier to obtain one action class prediction. The per-frame inference time of **DMC-Net (I3D)** is 0.79ms which is slightly slower yet very close to **DMC-Net (ResNet-18)** (*i.e.* 0.76ms). But in order to make one action prediction, **DMC-Net (I3D)** needs to take 0.79x250=197.5ms while **DMC-Net (ResNet-18)** only takes 0.76ms with the need of only one input frame.

8.4.6 Comparisons with Decoded Video Methods

In this section we compare DMC-Net to approaches that require decoding all RGB images from compressed video. Some only use the RGB images, while others adopt the two-stream method [34] and further require computing flow.

RGB only. As shown in Table 8.4, decoded video methods only based on RGB images can be further divided into three categories. **(1) Frame-level classification**: 2D CNNs like ResNet-50 and ResNet-152 [12] have been experimented in [196] to classify each frame individually and then employ simple averaging to obtain the video-level prediction. Due to lacking motion information, frame-level classification underperforms **DMC-Net**. **(2) Motion representation learning**: In Table 8.4, we evaluate **PWC-Net (ResNet-18) + CoViAR** which feeds estimated optical flow into a ResNet-18 classifier and then fuses the prediction with **CoViAR**. The accuracy of **PWC-Net (ResNet-18) + CoViAR** is not as good as DMC-Net because our generated DMC contains more discriminative motion cues that are complementary to MV. For TVNet [187], the authors used BN-Inception [123] to classify the generated motion representation and then fuse the prediction with a RGB CNN. The accuracy of TVNet is better **DMC-Net (ResNet-18)** thanks to using a strong classifier but is worse than our **DMC-Net (I3D)**. **(3) Spatio-temporal modeling**: There are also a lot of works using CNN to model the spatio-temporal patterns across multiple RGB frames to implicitly capture motion patterns. It turns out that our **DMC-Net** discovers motion cues that are complementary to such spatio-temporal patterns: **I3D RGB + DMC-Net (I3D)** improves **I3D RGB** via incorporating predictions from our **DMC-Net (I3D)**.

RGB + Flow. As shown in Table 8.4, the state-of-the-art accuracy is belonging to the two-stream methods [197; 33], which combine predictions made from a RGB CNN and an optical flow CNN. But as discussed in Section 8.4.4, extracting optical flow is quite time-consuming and thus these two-stream methods are much slower than our **DMC-Net**.

8.5 Summary

Motion has shown to be useful for action understanding, where motion is typically represented by optical flow. However, computing flow from video frames is very time-consuming. Recent works directly leverage the motion vectors and residuals readily available in the compressed video to represent motion at no cost. While this avoids flow computation, it also hurts accuracy since the motion vector is noisy and has substantially reduced resolution, which makes it a less discriminative motion representation.

We hypothesize that a fast deep network exclusively operating in the compressed domain can learn to generate a more Discriminative Motion Cue **(DMC)** representation compared to motion vectors, remedying the aforementioned issues. Thus, we propose a lightweight DMC generator network, which reduces noises in motion vectors and captures fine motion details. Since optical flow is a more accurate motion representation, we hypothesize that optical flow and semantic action labels can be used to supervise the training of DMC generator. To this end, we train the DMC generator to approximate flow using a reconstruction loss and a adversarial loss, jointly with the downstream action classification task.

On three action recognition benchmarks (HMDB-51 [165], UCF-101 [198] and a subset of Kinetics [197]), our full system **DMC-Net**, consisting of the generator and the classifier, obtains high accuracy close to that of using optical flow and runs two orders of magnitude faster than using optical flow at inference time. These results confirm the feasibility of generating Discriminative Motion Cue representation in the compressed domain without sacrificing speed. Our ablation studies confirm the effectiveness of leveraging both optical flow and semantic action labels as training supervision.

In the future, it would be interesting to further detect scene change in the compressed domain. If the scene changes, the motion vectors may not be able to capture the true moving trajectory of each macroblock and thus are not good for semantic action classification. Therefore, the generated DMC, which aims to refine motion vectors, is likely to fail for capturing discriminative motion information. Consequently, when the scene changes, it is better to rely on residual errors and I-frame rather than motion vectors and DMC for action classification. In addition, since the generated DMC aims at capturing discriminative motion representation rather than reconstructing perfect optical flow, it would be interesting to investigate how to predict optical flow in the compressed domain, targeting very low flow reconstruction error only without the requirement of high classification accuracy. This is useful for applications that only need optical flow but do not need to understand the semantics of actions. One potential solution is to remove the classification loss in our DMC-Net and feed the RGB image of I-frame into the DMC generator as well to leverage more input information.

Part IV

Conclusion

Chapter 9

Conclusion

9.1 Summary of Contributions

This thesis is dedicated to developing deep learning based methods to understand actions in video. To this end, we have focused on three main challenges: action detection at fine granularities in time (Part I), action detection in the constrained scenarios (Part II), and action understanding in the compressed domain (Part III).

For action detection at fine granularities in time, we have proposed new frameworks for detecting at segment-level and frame-level respectively. For segment-level action detection, we have hypothesized that we need a multi-stage processes: (1) instead of directly performing classification based on exhaustive scanning, doing segment proposal before classification will filter out unlikely segments to be fed into classification, reducing false alarms; (2) instead of the conventional Softmax classification loss, training a localization network with a new loss, which takes into account the overlap between the predicted segment and ground truth segment, will boost the localization accuracy. Thus, we have introduced an effective multi-stage framework called Segment-CNN, which are the first to exploit 3D ConvNets with multi-stage processes for temporal action localization in untrimmed, long videos in the wild. When the overlap threshold used in evaluation is set to 0.5, our proposed improves mAP on MEXaction2 from 1.7% to 7.4% and mAP on THUMOS 2014 from 15.0% to 19.0%, confirming the effectiveness of our multi-stage framework. The ablation studies have further validated the need of eliminating unlikely candidate segments by doing segment proposal first and showed that the localization network is important for improving the localization accuracy. For frame-level action detection, we have hypothesized that jointly modeling action semantics in space-time and fine-grained temporal dynamics can more accurately predict actions at frame-level. Therefore, we have proposed a novel Convolutional-De-Convolutional (CDC) filter to simultaneously perform spatial downsampling (for spatio-temporal semantic abstraction) and temporal upsampling (for detect at a finer granularity in time), and designed a CDC network to predict actions at frame-level. Our CDC network improves mAP on THUMOS'14 from 41.3% to 44.4% for the per-frame labeling task, confirming the need of joint modeling in both space and time. Further, we have hypothesized that the frame-level predictions can be used to detect precise segment boundary for the temporal action localization task. To this end, we have leveraged such frame-level predictions to refine temporal boundaries of segment proposals. Our method significantly improves the state-of-the-art mAP for the temporal action localization task from 19.0% to

23.3% on THUMOS'14 and from 16.4% to 23.8% on ActivityNet v1.3, validating the usefulness of making predictions at frame-level for detecting precise segment boundary.

For action detection in the constrained scenarios, we have respectively investigated the scenarios of incomplete supervision and incomplete input data. For the scenario of incomplete supervision, we have targeted weakly-supervised action detection and hypothesized that it is important to specifically address the localization task at the segment level by direct segment boundary prediction. Therefore, we have proposed AutoLoc which is the first weakly-supervised temporal action localization framework that can directly predict the temporal boundary of each action instance with only the video-level annotations available during training. To provide training supervision for such a boundary prediction model, we have hypothesized that a likely action segment would have high activations within its boundary and low activations in its contextual area. Thus we have designed a novel OIC loss to automatically discover the segment-level supervision by looking into the contrast between activations inside the predicted segment and its outside contextual area. Our method significantly improves mAP on THUMOS'14 from 13.7% to 21.2% and mAP on ActivityNet v1.2 from 7.4% to 27.3%, confirming our hypothesis regarding the need of direct boundary prediction. Our ablation studies also have validated our design principles for the proposed OIC loss which looks into activations inside and outside the predicted action segment. For the scenario of incomplete input data, we have proposed a novel task called Online Detection of Action Start (ODAS) in untrimmed, unconstrained, streaming videos to specifically focus on detecting Action Start timely and accurately. We have designed three novel methods for training effective ODAS models: (1)

generating hard negative samples based on GAN to assist ODAS models in discriminating start windows from negatives, (2) modeling the temporal consistency between the start window and its follow-up window to encourage their feature similarity, and finally (3) adaptively sampling start windows more frequently to address the training sample unbalance issue. Extensive comparisons on THUMOS'14 and ActivityNet have demonstrated the superiority of our approach over the conventional methods designed for online detection, per-frame labeling, temporal localization, and shot boundary detection in specifically solving the ODAS problem. The ablation studies have confirmed the effectiveness and necessity of each proposed training method.

For action understanding in the compressed domain, we have studied two imperative issues in developing the backbone networks which can benefit all kinds of action understanding tasks. First, we have studied the effects of video encoding variations for action understanding. We have experimentally compared several methods for training data preparation and found that the model trained with video encoded by some certain formats can be robust enough to handle the testing video encoding variance. Second, we have hypothesized that a fast deep network exclusively operating in the compressed domain can learn to generate a more discriminative motion representation compared to motion vectors. In order to supervise the training of such a generator, we have hypothesized that optical flow and semantic action labels can provide the needed supervision. Thus we have proposed DMC-Net, a novel and highly efficient framework can predict discriminative motion cues by learning to approximate optical flow and being trained jointly with the action classifier. During inference, it runs two orders of magnitude faster than estimating flow. We have extensively evaluated DMC-Net on 3 action recognition benchmarks and demonstrated that DMC-Net can significantly shorten the performance gap between state-of-the-art compressed video based methods with and without optical flow, confirming the feasibility of generating discriminative motion representation in the compressed domain without sacrificing speed. The ablation studies have confirmed the contributions from leveraging both optical flow and semantic action labels as training supervision.

9.2 Open Issues

Despite the significant contributions made in this thesis, there remain exciting open issues for video action understanding backed up by deep learning techniques.

9.2.1 Spatio-temporal Action Localization

Beyond action detection at segment-level and frame-level, there have been explorations about localizing action in both space and time simultaneously. Jain *et al.* [199] and Soomro *et al.* [200] built their work on supervoxel. Recently, researchers treat this as a tracking problem [201; 202] by leveraging object detectors [203], especially human detectors [204; 205; 202; 206] to detect regions of interest in each frame and then output sequences of bounding boxes. Dense trajectories have also been exploited for extracting the action tubes [207; 208]. Jain *et al.* [209] added object encodings to help action localization.

In the future, it would be interesting to extend our techniques developed in Part I from 1D in time only to 3D in both space and time. Spatio-temporal localization requires exhaustive annotations for objects of interest on every frame as training data. Thus it would also be interesting to also extend our weakly-supervised detection framework in Chapter 5 to discover the needed supervision.

9.2.2 Complex Event Detection

A complex event usually consists of a collection of actions and contains high-level semantics. Methods developed in this thesis focus on recognizing and detecting actions, which are necessary stepping stones towards detecting complex events in video. In the future, it is important to model the interactions and dependencies between different actions contained in one single complex event. To this end, one promising technical direction is leveraging attention models such as graph convolution [210] and self-attention mechanics [211]. Another technical direction could be adopting state-of-the-art pattern mining techniques to identify actions that appear together frequently.

9.2.3 Joint Video Compression and Understanding

In Part III, we have developed techniques to utilize data modalities in the compressed domain to directly perform action understanding. Nowadays, prevailing video codecs (*e.g.* MPEG-4, HEVC/H.265, AVC/H.264) are hand designed, preventing the joint optimization of the video compression codecs and the downstream action understanding models. There have been a few recent attempts to develop deep learning based video compression models [212; 213]. This trend of developing deep learning based video codecs makes it possible to jointly model video compression and understanding in an end-to-end fashion: optimizing compression with action understanding can allow the compression model to allocate less bits for the areas that are not perceptually important; optimizing action understanding with compression can provide more semantically discriminative compressed data modalities as input for the downstream action understanding models, improving the accuracy of action understanding.

Part V

Bibliography

Bibliography

[1] I. Laptev, "On space-time interest points," *International journal of computer vision*, 2005.

[2] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *Proceedings of the IEEE international conference on computer vision*, pp. 3551–3558, 2013.

[3] C.-Y. Wu, M. Zaheer, H. Hu, R. Manmatha, A. J. Smola, and P. Krähenbühl, "Compressed video action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[4] D. Weinland, R. Ronfard, and E. Boyer, "A survey of vision-based methods for action representation, segmentation and recognition," in *Computer Vision and Image Understanding*, 2011.

[5] R. Poppe, "A survey on vision-based human action recognition," in *Image and vision computing*, 2010.

[6] J. K. Aggarwal and M. S. Ryoo, "Human activity analysis: A review," in *ACM Computing Surveys*, 2011.

[7] G. Cheng, Y. Wan, A. N. Saudagar, K. Namuduri, and B. P. Buckles, "Advances in human action recognition: A survey," 2015.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012.

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, 2015.

[10] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," 2016.

[11] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, 2015.

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

[13] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[14] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[15] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 4489–4497, 2015.

[16] D. Tran, J. Ray, Z. Shou, S.-F. Chang, and M. Paluri, "Convnet architecture search for spatiotemporal feature learning," *arXiv preprint arXiv:1708.05038*, 2017.

[17] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[18] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[19] S. Ma, L. Sigal, and S. Sclaroff, "Learning activity progression in lstms for activity detection and early detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[20] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, “Social lstm: Human trajectory prediction in crowded spaces,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems*, 2014.

[22] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” *arXiv preprint*, 2017.

[23] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[24] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, *et al.*, “Photo-realistic single image super-resolution using a generative adversarial network.,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, p. 4, 2017.

[25] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, “Context encoders: Feature learning by inpainting,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2536–2544, 2016.

[26] M. Mathieu, C. Couprie, and Y. LeCun, “Deep multi-scale video prediction beyond mean square error,” *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[27] C. Vondrick and A. Torralba, “Generating the future with adversarial transformers,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[28] A. X. Lee, R. Zhang, F. Ebert, P. Abbeel, C. Finn, and S. Levine, “Stochastic adversarial video prediction,” *arXiv preprint arXiv:1804.01523*, 2018.

[29] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, “Action Recognition by Dense Trajectories,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[30] H. Wang and C. Schmid, “Action recognition with improved trajectories,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2013.

[31] F. Perronnin, J. Sánchez, and T. Mensink., “Improving the fisher kernel for large-scale image classification,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2010.

[32] J. Sánchez, T. Mensink, F. Perronnin, and T. Mensink., “Image classification with the fisher vector: Theory and practice,” 2013.

[33] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, “A closer look at spatiotemporal convolutions for action recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6450–6459, 2018.

[34] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” in *Advances in Neural Information Processing Systems*, 2014.

[35] D. Oneata, J. Verbeek, and C. Schmid, “The lear submission at thumos 2014,” in *ECCV THUMOS Workshop*, 2014.

[36] L. Wang, Y. Qiao, and X. Tang, “Action recognition and detection by combining motion and appearance features,” in *ECCV THUMOS Workshop*, 2014.

[37] S. Karaman, L. Seidenari, and A. D. Bimbo, “Fast saliency based pooling of fisher encoded dense trajectories,” in *ECCV THUMOS Workshop*, 2014.

[38] Y.-G. Jiang, J. Liu, A. R. Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar, “THUMOS challenge: Action recognition with a large number of classes.” `http://crcv.ucf.edu/THUMOS14/`, 2014.

[39] J. Yuan, Y. Pei, B. Ni, P. Moulin, and A. Kassim, “Adsc submission at thumos challenge 2015,” in *CVPR THUMOS Workshop*, 2015.

[40] A. Gorban, H. Idrees, Y.-G. Jiang, A. R. Zamir, I. Laptev, M. Shah, and R. Sukthankar, “THUMOS challenge: Action recognition with a large number of classes.” `http://www.thumos.info/`, 2015.

[41] D. Oneata, J. Verbeek, and C. Schmid, “Action and event recognition with fisher vectors on a compact feature set,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2013.

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

[43] R. Girshick, “Fast r-cnn,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[44] W. Kuo, B. Hariharan, and J. Malik, “Deepbox: Learning objectness with convolutional networks,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[45] S. Ji, W. Xu, M. Yang, and K. Yu, “3d convolutional neural networks for human action recognition,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2013.

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

[47] K.-T. Lai, D. Liu, M.-S. Chen, and S.-F. Chang, “Recognizing complex events in videos by learning key static-dynamic evidences,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

[48] K.-T. Lai, F. X. Yu, M.-S. Chen, and S.-F. Chang, “Video event detection by inferring temporal instance labels,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[49] C. Sun, S. Shetty, R. Sukthankar, and R. Nevatia, “Temporal localization of fine-grained actions in videos by domain transfer from web images,” in *Proceedings of the ACM international conference on Multimedia (MM)*, 2015.

[50] Z. Lan, M. Lin, X. Li, A. G. Hauptmann, and B. Raj, “Beyond gaussian pyramid: Multi-skip feature stacking for action recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[51] A. Richard and J. Gall, “Temporal action detection using a statistical language model,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[52] F. C. Heilbron, J. C. Niebles, and B. Ghanem, "Fast temporal activity proposals for efficient detection of human actions in untrimmed videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

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

[54] Z. Xu, Y. Yang, and A. G. Hauptmann, "A discriminative cnn video representation for event detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[55] D. Oneata, J. Verbeek, and C. Schmid., "Efficient action localization with approximately normalized fisher vectors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[56] A. Stoian, M. Ferecatu, J. Benois-Pineau, and M. Crucianu, "Fast action localization in large scale video archives," in *TCSVT*, 2015.

[57] A. Gaidon, Z. Harchaoui, and C. Schmid, "Actom sequence models for efficient action detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[58] A. Gaidon, Z. Harchaoui, and C. Schmid, "Temporal localization of actions with actoms," in *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2013.

[59] V. Escorcia, F. C. Heilbron, J. C. Niebles, and B. Ghanem, "Daps: Deep action proposals for action understanding," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[60] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei, "End-to-end learning of action detection from frame glimpses in videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[61] S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and L. Fei-Fei, "Every moment counts: Dense detailed labeling of actions in complex videos," *arXiv preprint arXiv:1507.05738*, 2015.

[62] J. Yuan, B. Ni, X. Yang, and A. Kassim, "Temporal action localization with pyramid of score distribution features," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[63] C. Lea, A. Reiter, R. Vidal, and G. D. Hager, “Segmental spatiotemporal cnns for fine-grained action segmentation,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[64] K. Soomro, A. R. Zamir, and M. Shah, “UCF101: A dataset of 101 human actions classes from videos in the wild,” in *CRCV-TR-12-01*, 2012.

[65] “Mexaction2.” `http://mexculture.cnam.fr/xwiki/bin/view/Datasets/Mex+action+dataset`, 2015.

[66] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” in *Proceedings of the ACM international conference on Multimedia (MM)*, 2014.

[67] Z. Shou, D. Wang, and S.-F. Chang, “Temporal action localization in untrimmed videos via multi-stage cnns,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[68] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[69] E. Shelhamer, J. Long, and T. Darrell, “Fully convolutional networks for semantic segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2016.

[70] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Deep end2end voxel2voxel prediction,” in *CVPR Workshop on Deep Learning in Computer Vision*, 2016.

[71] M. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, “Deconvolutional networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[72] M. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

[73] H. Noh, S. Hong, and B. Han, “Learning deconvolution network for semantic segmentation,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[74] S. Hong, H. Noh, and B. Han, “Decoupled deep neural network for semi-supervised semantic segmentation,” in *Advances in Neural Information Processing Systems*, 2015.

[75] S. Liu, X. Qi, J. Shi, H. Zhang, and J. Jia, "Multi-scale patch aggregation (mpa) for simultaneous detection and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[76] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2016.

[77] G. Lin, C. Shen, A. van den Hengel, and I. Reid, "Efficient piecewise training of deep structured models for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[78] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr, "Conditional random fields as recurrent neural networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[79] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[80] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[81] T. O'Haver, "Introduction to signal processing - deconvolution." `https://terpconnect.umd.edu/~toh/spectrum/Deconvolution.html`, 2007.

[82] F. C. Heilbron, V. Escorcia, B. Ghanem, and J. C. Niebles, "Activitynet: A large-scale video benchmark for human activity understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[83] "Activitynet challenge 2016." `http://activity-net.org/challenges/2016/`, 2016.

[84] R. Wang and D. Tao, "Uts at activitynet 2016," in *CVPR ActivityNet Workshop*, 2016.

[85] H. Zhao, Z. Yan, H. Wang, L. Torresani, and A. Torralba, "Slac: A sparsely labeled dataset for action classification and localization," *arXiv preprint arXiv:1712.09374*, 2017.

[86] L. Wang, Y. Xiong, D. Lin, and L. Van Gool, “Untrimmednets for weakly supervised action recognition and detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[87] K. K. Singh and Y. J. Lee, “Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[88] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Learning deep features for discriminative localization,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

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

[90] T. Lin, X. Zhao, and Z. Shou, “Single shot temporal action detection,” in *Proceedings of the 2017 ACM on Multimedia Conference*, pp. 988–996, ACM, 2017.

[91] J. Gao, Z. Yang, C. Sun, K. Chen, and R. Nevatia, “Turn tap: Temporal unit regression network for temporal action proposals,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[92] H. Xu, A. Das, and K. Saenko, “R-c3d: region convolutional 3d network for temporal activity detection,” in *IEEE Int. Conf. on Computer Vision (ICCV)*, pp. 5794–5803, 2017.

[93] J. Gao, Z. Yang, and R. Nevatia, “Cascaded boundary regression for temporal action detection,” in *Proceedings of the British Machine Vision Conference (BMVC)*, 2017.

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

[95] G. A. Sigurdsson, O. Russakovsky, A. Farhadi, I. Laptev, and A. Gupta, “Much ado about time: Exhaustive annotation of temporal data,” in *HCOMP*, 2016.

[96] S. Buch, V. Escorcia, C. Shen, B. Ghanem, and J. C. Niebles, “Sst: Single-stream temporal action proposals,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6373–6382, IEEE, 2017.

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

[98] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[99] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

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

[101] D.-A. Huang, L. Fei-Fei, and J. C. Niebles, "Connectionist temporal modeling for weakly supervised action labeling," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[102] A. Richard, H. Kuehne, and J. Gall, "Weakly supervised action learning with rnn based fine-to-coarse modeling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[103] P. Mettes, J. C. van Gemert, and C. G. Snoek, "Spot on: Action localization from pointly-supervised proposals," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[104] Y. Zhu, Y. Zhou, Q. Ye, Q. Qiu, and J. Jiao, "Soft proposal networks for weakly supervised object localization," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[105] M. Shi, H. Caesar, and V. Ferrari, "Weakly supervised object localization using things and stuff transfer," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[106] D. Kim, D. Yoo, I. S. Kweon, *et al.*, "Two-phase learning for weakly supervised object localization," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[107] Z. Jie, Y. Wei, X. Jin, J. Feng, and W. Liu, “Deep self-taught learning for weakly supervised object localization,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[108] T. Durand, T. Mordan, N. Thome, and M. Cord, “Wildcat: Weakly supervised learning of deep convnets for image classification, pointwise localization and segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[109] P. Tang, X. Wang, X. Bai, and W. Liu, “Multiple instance detection network with online instance classifier refinement,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[110] J. Zhang, Z. Lin, J. Brandt, X. Shen, and S. Sclaroff, “Top-down neural attention by excitation backprop,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[111] C. Sun, M. Paluri, R. Collobert, R. Nevatia, and L. Bourdev, “Pronet: Learning to propose object-specific boxes for cascaded neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[112] H. Bilen and A. Vedaldi, “Weakly supervised deep detection networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[113] V. Kantorov, M. Oquab, M. Cho, and I. Laptev, “Contextlocnet: Context-aware deep network models for weakly supervised localization,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[114] A. Gudi, N. van Rosmalen, M. Loog, and J. van Gemert, “Object-extent pooling for weakly supervised single-shot localization,” in *Proceedings of the British Machine Vision Conference (BMVC)*, 2017.

[115] A. Khoreva, R. Benenson, J. Hosang, M. Hein, and B. Schiele, “Simple does it: Weakly supervised instance and semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[116] S. Hong, D. Yeo, S. Kwak, H. Lee, and B. Han, “Weakly supervised semantic segmentation using web-crawled videos,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[117] G. Papandreou, L.-C. Chen, K. P. Murphy, and A. L. Yuille, “Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[118] A. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei, “Whats the point: Semantic segmentation with point supervision,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[119] Z. Shen, J. Li, Z. Su, M. Li, Y. Chen, Y.-G. Jiang, and X. Xue, “Weakly supervised dense video captioning,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[120] H. Zhang, Z. Kyaw, J. Yu, and S.-F. Chang, “Ppr-fcn: weakly supervised visual relation detection via parallel pairwise r-fcn,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[121] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez, “Solving the multiple instance problem with axis-parallel rectangles,” 1997.

[122] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. V. Gool, “Temporal segment networks: Towards good practices for deep action recognition,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[123] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

[124] Z. Yuan, J. C. Stroud, T. Lu, and J. Deng, “Temporal action localization by structured maximal sums,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[125] Z. Shou, J. Chan, A. Zareian, K. Miyazawa, and S.-F. Chang, “Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[126] X. Dai, B. Singh, G. Zhang, L. S. Davis, and Y. Q. Chen, “Temporal context network for activity localization in videos,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[127] T. Lindeberg, "Feature detection with automatic scale selection," *International Journal of Computer Vision (IJCV)*, 1998.

[128] R. D. Geest, E. Gavves, A. Ghodrati, Z. Li, C. Snoek, and T. Tuytelaars, "Online action detection," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[129] J. Gao, Z. Yang, and R. Nevatia, "Red: Reinforced encoder-decoder networks for action anticipation," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2017.

[130] M. Hoai and F. De la Torre, "Max-margin early event detectors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[131] M. Hoai and F. De la Torre, "Max-margin early event detectors," in *International Journal of Computer Vision (IJCV)*, 2014.

[132] F. C. Heilbron, W. Barrios, V. Escorcia, and B. Ghanem, "Scc: Semantic context cascade for efficient action detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[133] A. Dave, O. Russakovsky, and D. Ramanan, "Predictive-corrective networks for action detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[134] G. A. Sigurdsson, S. Divvala, A. Farhadi, and A. Gupta, "Asynchronous temporal fields for action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[135] Z. Yang, J. Gao, and R. Nevatia, "Spatio-temporal action detection with cascade proposal and location anticipation," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2017.

[136] J. Gao, C. Sun, Z. Yang, and R. Nevatia, "Tall: Temporal activity localization via language query," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[137] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[138] Y. Kong, D. Kit, and Y. Fu, “A discriminative model with multiple temporal scales for action prediction,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

[139] G. Yu, J. Yuan, and Z. Liu, “Predicting human activities using spatio-temporal structure of interest points,” in *Proceedings of the ACM international conference on Multimedia (MM)*, 2012.

[140] Y. Cao, D. Barrett, A. Barbu, S. Narayanaswamy, H. Yu, A. Michaux, Y. Lin, S. Dickinson, J. Mark Siskind, and S. Wang, “Recognize human activities from partially observed videos,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[141] M. S. Ryoo, “Human activity prediction: Early recognition of ongoing activities from streaming videos,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2011.

[142] Y. Kong, Z. Tao, and Y. Fu, “Deep sequential context networks for action prediction,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[143] T. Lan, T.-C. Chen, and S. Savarese, “A hierarchical representation for future action prediction,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

[144] M. S. Aliakbarian, F. Saleh, M. Salzmann, B. Fernando, L. Petersson, and L. Andersson, “Encouraging lstms to anticipate actions very early,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[145] D. Huang, S. Yao, Y. Wang, and F. De La Torre, “Sequential max-margin event detectors,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

[146] G. Singh, S. Saha, and F. Cuzzolin, “Online real time multiple spatiotemporal action localisation and prediction on a single platform,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

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

[148] Y. Li, C. Lan, J. Xing, W. Zeng, C. Yuan, and J. Liu, "Online human action detection using joint classification-regression recurrent neural networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[149] C. Liu, Y. Li, Y. Hu, and J. Liu, "Online action detection and forecast via multitask deep recurrent neural networks," in *ICASSP*, 2017.

[150] C. Vondrick, H. Pirsiavash, and A. Torralba, "Anticipating the future by watching unlabeled video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[151] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

[152] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[153] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[154] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[155] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[156] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems*, 2016.

[157] J. T. Springenberg, "Unsupervised and semi-supervised learning with categorical generative adversarial networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[158] Z. Dai, Z. Yang, F. Yang, W. W. Cohen, and R. Salakhutdinov, "Good semi-supervised learning that requires a bad gan," in *Advances in Neural Information Processing Systems*, 2017.

[159] J. S. Boreczky and L. A. Rowe, “Comparison of video shot boundary detection techniques,” *Journal of Electronic Imaging*, 1996.

[160] K. Warhade, S. N. Merchant, U. B. Desai, *et al.*, *Video Shot Boundary Detection*. River Publishers, 2011.

[161] A. F. Smeaton, P. Over, and A. R. Doherty, “Video shot boundary detection: Seven years of trecvid activity,” *Computer Vision and Image Understanding*, 2010.

[162] D. Le Gall, “Mpeg: A video compression standard for multimedia applications,” *Communications of the ACM*, 1991.

[163] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao, “Towards good practices for very deep two-stream convnets,” *arXiv preprint arXiv:1507.02159*, 2015.

[164] “Ffmpeg: A complete, cross-platform solution to record, convert and stream audio and video..” `https://www.ffmpeg.org/`.

[165] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, “Hmdb: a large video database for human motion recognition,” in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2556–2563, IEEE, 2011.

[166] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, “Learning realistic human actions from movies,” in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, IEEE, 2008.

[167] Z. Shou, H. Gao, L. Zhang, K. Miyazawa, and S.-F. Chang, “Autoloc: Weaklysupervised temporal action localization in untrimmed videos,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 162–179, 2018.

[168] Z. Shou, J. Chan, A. Zareian, K. Miyazawa, and S.-F. Chang, “Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos,” in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pp. 1417–1426, IEEE, 2017.

[169] Y.-W. Chao, S. Vijayanarasimhan, B. Seybold, D. A. Ross, J. Deng, and R. Sukthankar, “Rethinking the faster r-cnn architecture for temporal action localization,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1130–1139, 2018.

[170] Z. Shou, J. Pan, J. Chan, K. Miyazawa, H. Mansour, A. Vetro, X. Giro-i Nieto, and S.-F. Chang, "Online detection of action start in untrimmed, streaming videos," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[171] T. Lin, X. Zhao, and Z. Shou, "Temporal convolution based action proposal: Submission to activitynet 2017," *arXiv preprint arXiv:1707.06750*, 2017.

[172] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang, "Real-time action recognition with deeply transferred motion vector cnns," *IEEE Transactions on Image Processing*, 2018.

[173] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang, "Real-time action recognition with enhanced motion vector cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[174] L. Sevilla-Lara, Y. Liao, F. Guney, V. Jampani, A. Geiger, and M. J. Black, "On the integration of optical flow and action recognition," *arXiv preprint arXiv:1712.08416*, 2017.

[175] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[176] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime tv-l 1 optical flow," in *Joint Pattern Recognition Symposium*, pp. 214–223, Springer, 2007.

[177] J. L. M. B. D Sun, S Roth, "Learning optical flow," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2008.

[178] A. Bruhn, J. Weickert, and C. Schnörr, "Lucas/kanade meets horn/schunck: Combining local and global optic flow methods," *International journal of computer vision*, vol. 61, no. 3, pp. 211–231, 2005.

[179] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2758–2766, 2015.

[180] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *IEEE conference on computer vision and pattern recognition (CVPR)*, vol. 2, p. 6, 2017.

[181] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, p. 2, IEEE, 2017.

[182] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[183] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European Conf. on Computer Vision (ECCV)* (A. Fitzgibbon et al. (Eds.), ed.), Part IV, LNCS 7577, pp. 611–625, Springer-Verlag, Oct. 2012.

[184] M. Menze, C. Heipke, and A. Geiger, "Joint 3d estimation of vehicles and scene flow," in *ISPRS Workshop on Image Sequence Analysis (ISA)*, 2015.

[185] R. Gao, B. Xiong, and K. Grauman, "Im2flow: Motion hallucination from static images for action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[186] L. Sevilla-Lara, Y. Liao, F. Guney, V. Jampani, A. Geiger, and M. J. Black, "On the integration of optical flow and action recognition," in *German Conference on Pattern Recognition (GCPR)*, Oct. 2018.

[187] L. Fan, W. Huang, S. E. Chuang Gan, B. Gong, and J. Huang, "End-to-end learning of motion representation for video understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6016–6025, 2018.

[188] J. Y.-H. Ng, J. Choi, J. Neumann, and L. S. Davis, "Actionflownet: Learning motion representation for action recognition," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1616–1624, IEEE, 2018.

[189] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, p. 3, 2013.

[190] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime tv-l1 optical flow," in *Joint Pattern Recognition Symposium*, 2007.

[191] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "Deepflow: Large displacement optical flow with deep matching," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013.

[192] L. Wang, W. Li, W. Li, and L. Van Gool, “Appearance-and-relation networks for video classification,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[193] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng, “Multi-fiber networks for video recognition,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[194] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy, “Rethinking spatiotemporal feature learning for video understanding,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[195] C. Feichtenhofer, A. Pinz, and A. Zisserman, “Convolutional two-stream network fusion for video action recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

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

[197] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, *et al.*, “The kinetics human action video dataset,” *arXiv preprint arXiv:1705.06950*, 2017.

[198] K. Soomro, A. R. Zamir, and M. Shah, “UCF101: A dataset of 101 human actions classes from videos in the wild,” *arXiv preprint arXiv:1212.0402*, 2012.

[199] M. Jain, J. van Gemert, H. Jégou, P. Bouthemy, and C. Snoek, “Action localization with tubelets from motion,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[200] K. Soomro, H. Idrees, and M. Shah, “Action localization in videos through context walk,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[201] P. Weinzaepfel, Z. Harchaoui, and C. Schmid, “Learning to track for spatio-temporal action localization,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[202] G. Gkioxari and J. Malik, “Finding action tubes,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[203] M. Jain, J. van Gemert, T. Mensink, and C. Snoek, “Objects2action: Classifying and localizing actions without any video example,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[204] Z. Jiang, Z. Lin, and L. S. Davis, “A unified tree-based framework for joint action localization, recognition and segmentation,” in *Computer Vision and Image Understanding*, 2013.

[205] A. Kläser, M. Marszałek, C. Schmid, and A. Zisserman, “Human focused action localization in video,” in *Trends and Topics in Computer Vision*, 2012.

[206] G. Yu and J. Yuan, “Fast action proposals for human action detection and search,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[207] M. M. Puscas, E. Sangineto, D. Culibrk, and N. Sebe, “Unsupervised tube extraction using transductive learning and dense trajectories,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[208] J. van Gemert, M. Jain, E. Gati, and C. Snoek, “Apt: Action localization proposals from dense trajectories,” in *Proceedings of the British Machine Vision Conference (BMVC)*, 2015.

[209] M. Jain, J. van Gemert, and C. Snoek, “What do 15,000 object categories tell us about classifying and localizing actions?,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[210] X. Wang and A. Gupta, “Videos as space-time region graphs,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[211] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

[212] C.-Y. Wu, N. Singhal, and P. Krähenbühl, “Video compression through image interpolation,” *arXiv preprint arXiv:1804.06919*, 2018.

[213] O. Rippel, S. Nair, C. Lew, S. Branson, A. G. Anderson, and L. Bourdev, “Learned video compression,” *arXiv preprint arXiv:1811.06981*, 2018.

Part VI

Appendix

Appendix A

Publications

- **Zheng Shou**, Xudong Lin, Yannis Kalantidis, Laura Sevilla-Lara, Marcus Rohrbach, Shih-Fu Chang, Zhicheng Yan. *DMC-Net: Generating Discriminative Motion Cues for Fast Compressed Video Action Recognition.* IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.
- **Zheng Shou**, Hang Gao, Lei Zhang, Kazuyuki Miyazawa, Shih-Fu Chang. *AutoLoc: Weakly-supervised Temporal Action Localization in Untrimmed Videos.* European Conference on Computer Vision (ECCV), 2018.
- **Zheng Shou***, Junting Pan*, Jonathan Chan, Kazuyuki Miyazawa, Hassan Mansour, Anthony Vetro, Xavi Gir-i-Nieto, Shih-Fu Chang. *Online Detection of Action Start in Untrimmed, Streaming Videos.* European Conference on Computer Vision (ECCV), 2018.

- **Zheng Shou**, Jonathan Chan, Alireza Zareian, Kazuyuki Miyazawa, Shih-Fu Chang. *CDC: Convolutional-De-Convolutional Networks for Precise Temporal Action Localization in Untrimmed Videos.* IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

- **Zheng Shou**, Dongang Wang, Shih-Fu Chang. *Temporal Action Localization in Untrimmed Videos via Multi-stage CNNs.* IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

- Hang Gao, **Zheng Shou**, Alireza Zareian, Hanwang Zhang, Shih-Fu Chang. *Low-shot Learning via Covariance-Preserving Adversarial Augmentation Network.* Thirty-second Conference on Neural Information Processing Systems (NeurIPS), 2018.

- Du Tran, Jamie Ray, **Zheng Shou**, Shih-Fu Chang, Manohar Paluri. *ConvNet Architecture Search for Spatiotemporal Feature Learning.* Technical Report. 2017.