Action Recognition, Temporal Localization and Detection in Trimmed and Untrimmed Video

Rui Hou
University of Central Florida

ACTION RECOGNITION, TEMPORAL LOCALIZATION AND DETECTION IN TRIMMED AND UNTRIMMED VIDEOS

by

RUI HOU
M.E. Arizona State University, 2012
B.S. Beijing University of Posts and Telecommunications, 2010

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2019

Major Professor: Mubarak Shah

ABSTRACT

Automatic understanding of videos is one of the most active areas of computer vision research. It has applications in video surveillance, human computer interaction, video sports analysis, virtual and augmented reality, video retrieval etc. In this dissertation, we address four important tasks in video understanding, namely action recognition, temporal action localization, spatial-temporal action detection and video object/action segmentation. This dissertation makes contributions to above tasks by proposing. First, for video action recognition, we propose a category level feature learning method. Our proposed method automatically identifies such pairs of categories using a criterion of mutual pairwise proximity in the (kernelized) feature space, and a category-level similarity matrix where each entry corresponds to the one-vs-one SVM margin for pairs of categories. Second, for temporal action localization, we propose to exploit the temporal structure of actions by modeling an action as a sequence of sub-actions and present a computationally efficient approach. Third, we propose 3D Tube Convolutional Neural Network (TCNN) based pipeline for action detection. The proposed architecture is a unified deep network that is able to recognize and localize action based on 3D convolution features. It generalizes the popular faster R-CNN framework from images to videos. Last, an end-to-end encoder-decoder based 3D convolutional neural network pipeline is proposed, which is able to segment out the foreground objects from the background. Moreover, the action label can be obtained as well by passing the foreground object into an action classifier. Extensive experiments on several video datasets demonstrate the superior performance of the proposed approach for video understanding compared to the state-of-the-art.

Our greatest glory is not in never falling, but in rising every time we fall.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Semantically understanding actions in videos continuously attract both academic computer vision researchers and industrial AI developers. Understanding actions include several aspects, such as action recognition, action detection or action segmentation. While humans are able to understand the videos and actions effortlessly, this is still a challenging task for computers due to the complex scene and intra-class variation.

There are multiple video action understanding tasks. One of the basic problems is video action recognition, which takes video as input and outputs a label per video. Action recognition is good at dealing with actions in trimmed video, but for long videos with multiple scenes and actions, which is a common case in real-world, most of the action recognition approaches will have difficulties in recognizing the actions. To better understand long videos, temporal action localization is introduced, which is able to divide a long video into several short clips based on the start/end timestamp of an action as well as recognize the action. Temporal action localization has been widely used in surveillance video. It saves tremendous human labor by detecting the anomalies automatically. However, temporal action localization is difficult in videos when there are multiple actions in the same scene. In cases where localizing action in both spatial and temporal domains are desired, we need to solve video action detection problem. The goal of video action detection is to generate one bounding box per action instance per frame. It has a variety of applications such as sports analysis and human-machine interaction. Recently, with the popularity of augmented reality, there is an increasing demand for pixel-level action localization, *e.g.*in virtual make-up or AR photo. Video action segmentation is designed to assign an action label for each pixel in video, thus localizing actions with fine contour.

This dissertation makes following contributions to deal with the above challenges in action recognition, temporal action localization, spatial-temporal action detection and video action/object segmentation:

- We propose discriminative and mutually nearest (DaMN) features for action recognition. The proposed approach is able to learn discriminative category-level features in a data driven manner. It automatically identifies similar category pairs through a criterion of mutual pairwise proximity in the feature space and a category-level similarity matrix where each entry corresponds to the one-vs-one SVM margin for pairs of categories. By augmenting regular one-vs-rest classifiers with a selection of two-vs-rest classifier outputs in late fusion manner, video level label is generated. Our experiments show that DaMN outperforms related approaches in direct comparisons, not only on video action recognition but also on their original image dataset tasks.

- We propose a temporal action localization approach using sub-action discovery. We exploit the temporal structure of actions by modeling an action as a sequence of sub-actions. The proposed sub-action discovery algorithm is novel and fully automatic, where the number of sub-actions for each action as well as their types are automatically determined from the training videos. To localize an action, an objective function combining appearance, duration and temporal structure of sub-actions is optimized as a shortest path problem in a network flow formulation.

- We propose an end-to-end deep network, Tube Convolutional Neural Network (T-CNN), for action detection. The proposed architecture is a unified deep network that is able to recognize and localize action based on 3D convolution features. A video is first divided into equal length clips and next for each clip a set of tube proposals are generated based on 3D Convolutional Network (ConvNet) features. Finally, the tube proposals of different clips are linked together employing network flow and spatial-temporal action detection is performed using these linked video proposals. Extensive experiments on several video datasets demonstrate the superior performance of T-CNN for classifying and localizing actions in both trimmed and untrimmed videos compared to state-of-the-arts.

- We propose an encoder-decoder based end-to-end 3D CNN framework for video object segmentation. The proposed approach adopts 3D CNN to aggregate spatial and temporal information in a single stream. To efficiently process video, we propose 3D separable convolution for the pyramid pooling module and decoder, which dramatically reduces the number of operations while maintaining the performance.

1.1 Action Recognition

Attributes are mid-level visual concepts, such as “smiling”, “brittle”, or “quick” that are typically employed to characterize categories at a semantic level. In recent years, attributes have been successfully applied to a variety of computer vision problems including face verification [48], image retrieval [15], action recognition [57], image-to-text generation [2]. Category-level attributes are popular not only because they can represent the shared semantic properties of visual classes but because they can leverage information from known categories to enable existing classifiers to generalize to novel categories for which there exists limited training data.

Ideally, attributes should capture human-interpretable semantic characteristics that can be reliably recognized by machines from visual data. However, the focus on human-interpretation means that developing attribute classifiers typically demands a labor-intensive process involving manual selection of attribute labels and collection of suitable training data by domain experts (e.g., [50]).

Our stance is that while human interpretability of attributes is obviously desirable, it should be treated as a secondary goal. Thus, we seek fully automated methods that learn discriminative category-level features to serve as useful mid-level representations, directly from data.

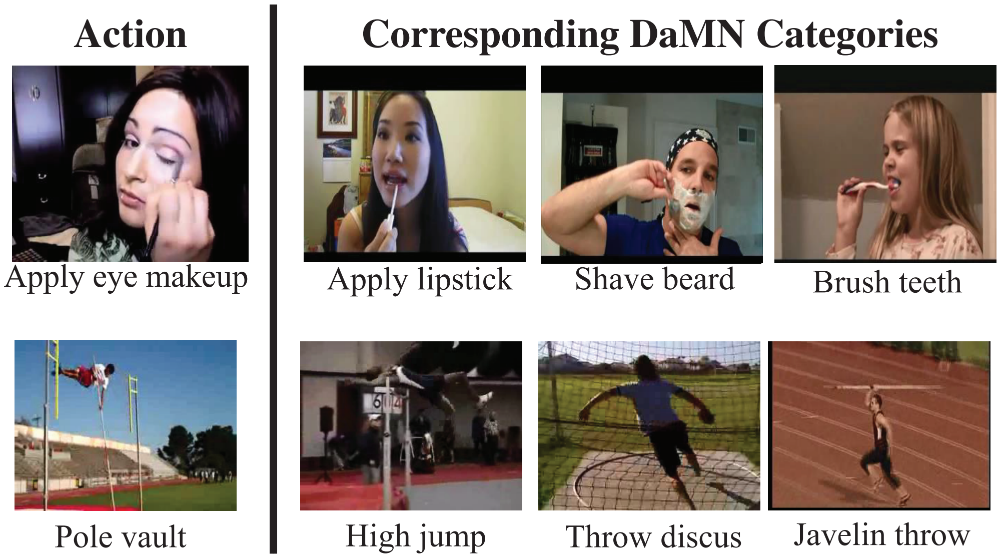

Figure 1.1: Examples of DaMN category pairs automatically discovered in UCF101. For the actions shown on the left, three sample mutually nearest categories are shown on the right. Explicitly representing such category-level information enables improved action recognition accuracy.

We propose DaMN, a method for automatically constructing category-level features for multi-class problems based on combining one-vs-one, one-vs-rest and two-vs-rest classifiers in a principled manner. Rather than requiring methods to accurately split such "Siamese Twin" categories, we choose to augment one-vs-rest classifiers with a *judicious selection* of two-vs-rest classifiers that keep the strongly related categories together. The challenge is how best to identify such categories and then how to combine information from the different classifiers in a principled manner. Figure 1.1 illustrates examples of category pairs identified as closely related by DaMN; the complete grouping of DaMN pairs extracted for UCF101 is shown in Figure 1.2. It is important to note that the DaMN category pairs are (by construction) similar in kernel space and generally well separated from the remaining categories. By contrast, the manually constructed category-attribute matrix for UCF101 is not as amenable to machine classification despite having human-interpretable names [36]. Such experiences drive us to explore the idea of data-driven category features as an alternative to human selected attributes, with the hope that such features can still offer the benefits (such as cross-dataset generalization and one-shot learning) of traditional attributes.

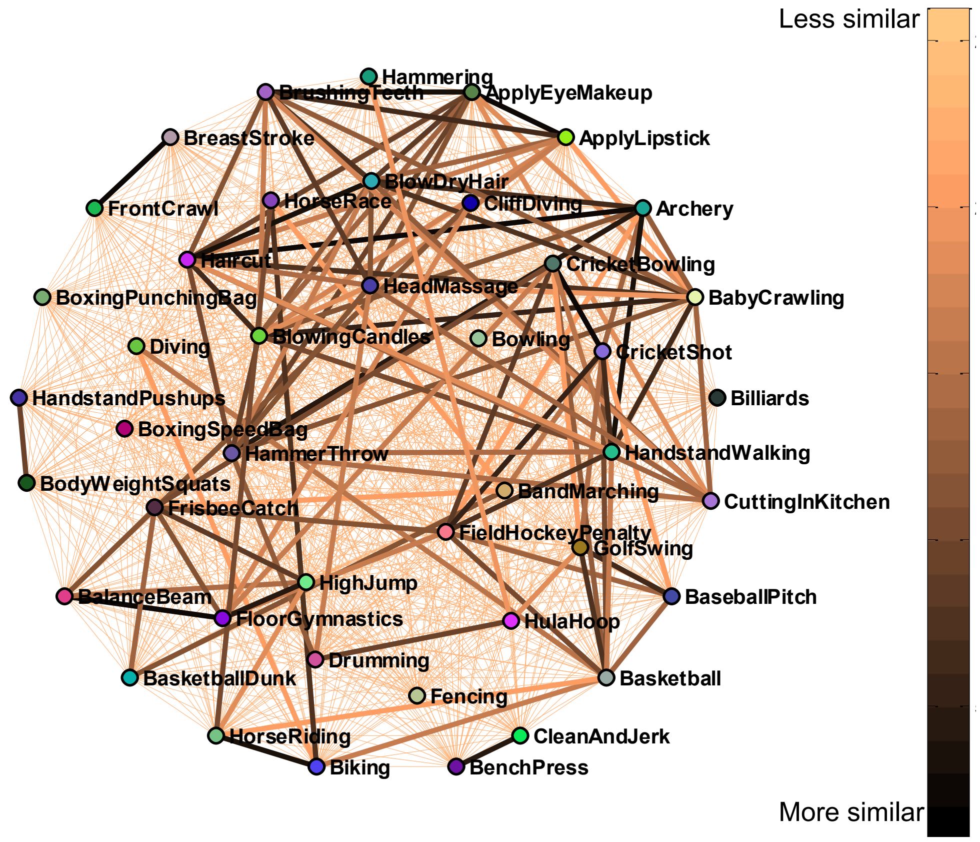

Figure 1.2: Visualization of the extracted pairwise category similarity for UCF101. To manage visual clutter, only the first 43 categories are shown, with edges shaded according to the pairwise similarity between the respective classes. We see that DaMN identifies meaningful semantic groups, such as *weightlifting*, *bat-and-ball sports*, and *face-centered actions*. Also note the presence of singleton categories.

1.2 Temporal Action Localization

Although DaMN has shown promising results on video action recognition, it is designed for clip-level classification in manually trimmed datasets. However, in the real world, the majority of videos are untrimmed, where an action of interest may occur only in a small part of a long video.

The challenge is to temporally localize an action of interest while ignoring other irrelevant actions and the background. In this context, most of the methods designed for trimmed videos will fail. In this section, we address the problem of real-time temporal action localization, which predicts the beginning and ending frames of an action in an untrimmed video, in a computationally efficient manner.

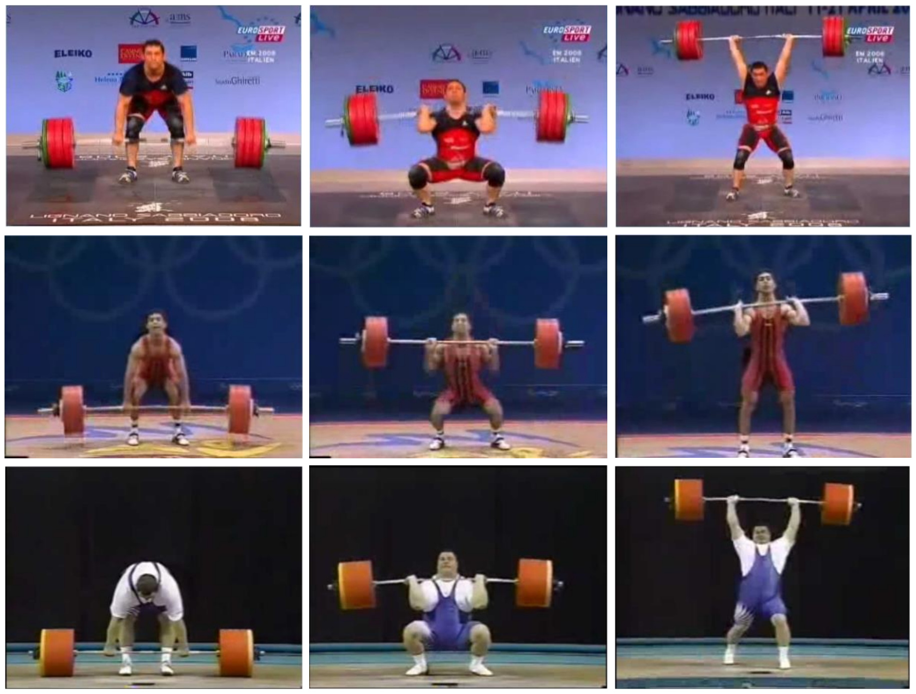

Figure 1.3: Automatically discovered sub-actions in diverse instances of two actions. The number of sub-actions is automatically determined and they are found to be semantically meaningful. Each row shows one example of "clean and jerk" and "long jump" actions from diverse videos.

In the proposed approach (Fig. 1.3), an action is modeled as a sequence of automatically discovered, semantically meaningful sub-actions, whose number and duration can vary from action to action. For instance, the "clean and jerk" action (left) is decomposed into three sub-actions, corresponding to "lift", "clean" and "jerk", while the "long jump" (right) has two sub-actions, that correspond to "approaching run" and "jump". Note that our discovered sub-actions are consistent across different instances that vary significantly in terms of viewpoint and visual appearance.

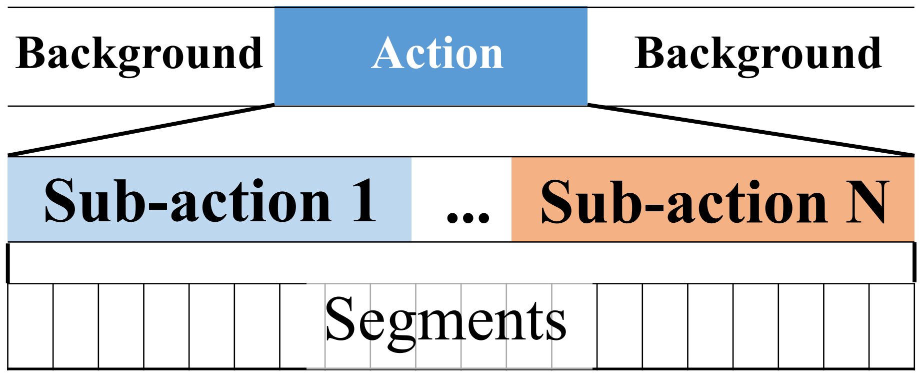

Figure 1.4: A typical untrimmed video consists of many background segments with one or more actions. We group short segments from untrimmed video into sub-actions whose temporal structure is exploited for temporal action localization.

Decomposing actions into characteristic sub-actions is a challenging task, due to significant intra-class variability, such as variations in viewpoints, human poses and the execution speed of the action. Therefore, most former approaches either rely on manual annotation of sub-actions or fix the number of sub-actions per action. Manually defining and annotating sub-actions is a subjective task and the quality of annotations depends on the judgment and preference of the annotators. Also, manual annotation of sub-actions requires heavy labor. In addition, actions contain different complexity levels; typically a few sub-actions are enough to represent simple actions, while complex actions would need to be represented with more sub-actions. Obviously, limiting the number of sub-actions to a fixed number for all actions would limit their representation power. In our approach, the number of sub-actions as well as the sub-actions themselves are learned from the training data automatically. Moreover, from the experiments, we observe that the automatically discovered sub-actions are semantically meaningful and they are consistent across different instances of an action.

Given the rate at which videos are generated, applications such as surveillance and video retrieval demand computationally efficient approaches for action detection in untrimmed video. Our proposed approach is designed with this in mind and can process 40 frames per second (on commodity hardware) including all steps, i.e., feature extraction, sub-action detection and action detection.

This chapter makes the following contributions.

1. We propose a novel, fully automated algorithm that discovers a discriminative sequence of sub-actions directly from data. The discovered sub-actions are semantically meaningful.
2. We exploit the temporal structure of actions by modeling an action as a sequence of sub-actions, which takes sub-action durations and time between sub-actions into consideration.
3. The proposed approach is computationally efficient and processes video in real-time.
4. We evaluate the proposed sub-action based sequential model on large-scale temporal action localization task and show that the proposed method achieves state-of-the-art results on THUMOS'14 and MEXaction2 datasets.

1.3 Video Action Detection

The above sub-action based approach is designed for temporally localizing the actions in long videos, but it is not good at spatially separating the actors from the background. The goal of action detection is to spatiotemporally localize the action by determing the position of the actor. In Chapter 5, we address the problem of action detection by leveraging the descriptiveness of Convolutional Neural Network. Our approach is the first end-to-end CNN based action detection pipeline.

Deep learning based approaches have significantly improved video action recognition performance. Compared to action recognition, action detection is a more challenging task due to flexible volume shape and large spatio-temporal search space.

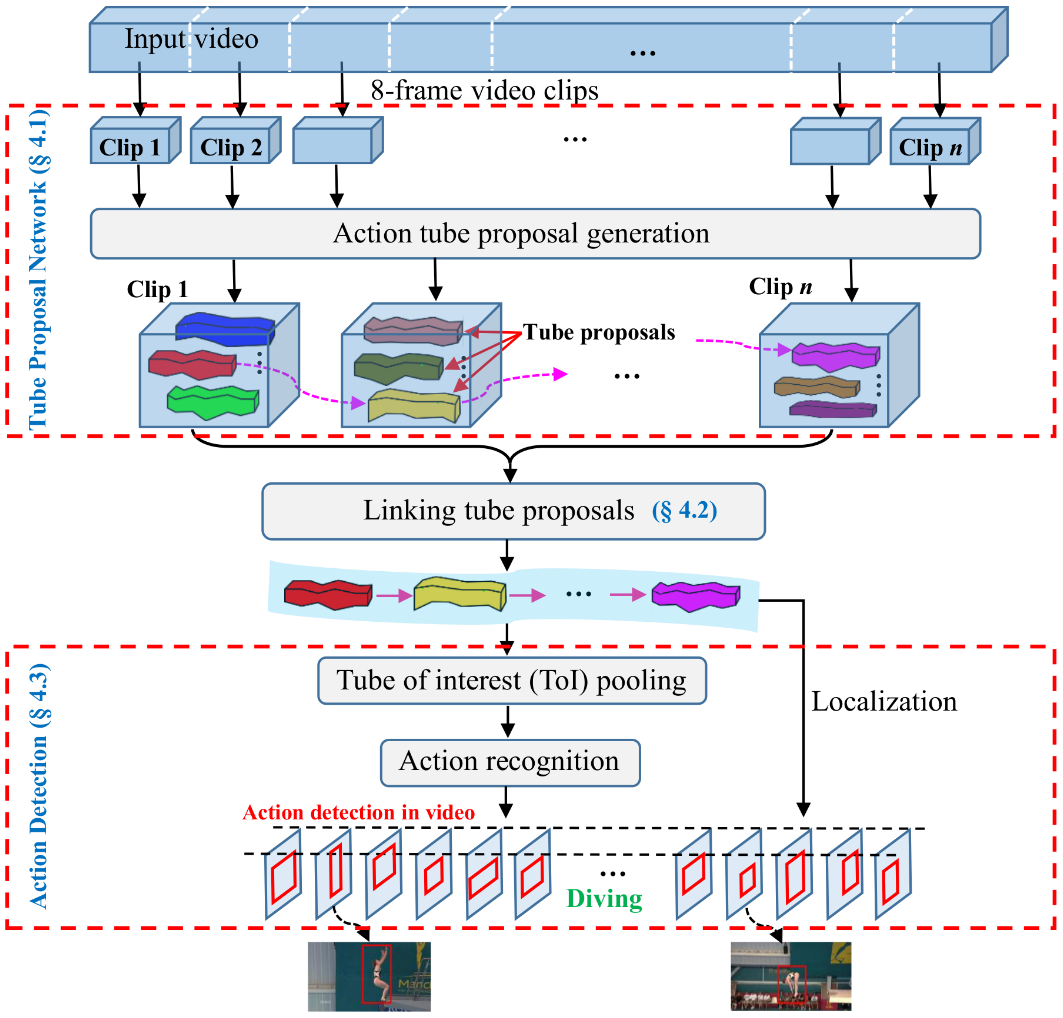
Input video
8-frame video clips
Tube Proposal Network (§ 4.1)
Clip 1
Clip 2
Clip n
Action tube proposal generation
Clip 1
Clip n
Tube proposals
Linking tube proposals (§ 4.2)
Action Detection (§ 4.3)
Tube of interest (ToI) pooling
Action recognition
Localization
Action detection in video
Diving

Figure 1.5: Overview of the proposed Tube Convolutional Neural Network (T-CNN).

Previous deep learning based action detection approaches first detect frame-level action proposals by popular proposal algorithms [23, 105] or by training proposal networks [72]. Then the frame-level action proposals are associated across frames to form final action detection through tracking based approaches. Moreover, in order to capture both spatial and temporal information of action, two-stream networks (a spatial CNN and a motion CNN) are used. In this manner, spatial and motion information are processed separately.

Region Convolution Neural Network (R-CNN) for object detection in images was proposed by Girshick *et al.*[22]. It was followed by a fast R-CNN proposed in [21], which includes the classifier as well. Later, faster R-CNN [75] was developed by introducing a region proposal network. It has been extensively used to produce excellent results for object detection in images. A natural generalization of the R-CNN from 2D images to 3D spatio-temporal volumes is to study their effectiveness for the problem of action detection in videos. A straightforward spatio-temporal generalization of the R-CNN approach would be to treat action detection in videos as a set of 2D image detections using faster R-CNN. However, unfortunately, this approach does not take the temporal information into account and is not sufficiently expressive to distinguish between actions.

Inspired by the pioneering work of faster R-CNN, we propose Tube Convolutional Neural Network (T-CNN) for action detection. To better capture the spatio-temporal information of video, we exploit 3D ConvNet for action detection, since it is able to capture motion characteristics in videos and shows promising result on video action recognition. We propose a novel framework by leveraging the descriptive power of 3D ConvNet. In our approach, an input video is divided into equal length clips first. Then, the clips are fed into Tube Proposal Network (TPN) and a set of tube proposals are obtained. Next, tube proposals from each video clip are linked according to their actionness scores and overlap between adjacent proposals to form a complete tube proposal for spatio-temporal action localization in the video. Finally, the Tube-of-Interest (ToI) pooling is applied to the linked action tube proposal to generate a fixed length feature vector for action label prediction.

Our work makes the following contributions:

1. We propose an end-to-end deep learning based approach for action detection in videos. It directly operates on the original videos and captures spatio-temporal information using a single 3D network to perform action localization and recognition based on 3D convolution

features. To the best of our knowledge, it is the first work to exploit 3D ConvNet for action detection.

2. We introduce a Tube Proposal Network, which leverages skip pooling in temporal domain to preserve temporal information for action localization in 3D volumes.

3. We propose a new pooling layer – **Tube-of-Interest** (ToI) pooling layer in T-CNN. The ToI pooling layer is a 3D generalization of Region-of-Interest (RoI) pooling layer of R-CNN. It effectively alleviates the problem with variable spatial and temporal sizes of tube proposals. We show that ToI pooling can greatly improve the recognition results.

4. We extensively evaluate our T-CNN for action detection in both trimmed videos from UCF-Sports, J-HMDB and UCF-101 datasets and untrimmed videos from THUMOS'14 dataset and achieve state-of-the-art performance.

1.4 Video Action and Object Segmentation

In the above section we introduce a 3D CNN based pipeline for video action detection. In this section a more challenging problem, video action segmentation, is solved. The challenge of video object segmentation (VOS) has two aspects. First, VOS is less robust to random noise since it aims to assign a foreground/background label for each pixel in a video frame. Second, the amount of data to be processed in video can be orders of magnitude greater than in image segmentation, placing greater constraints in terms of computational resources.

Video object segmentation approaches can be divided into two categories – semi-supervised and unsupervised. The semi-supervised approaches [61, 97] assume the foreground object in the first frame of test video is provided and the task is to segment the specified object in the following frames. On the other hand, the unsupervised approaches [44, 4, 83, 91] segment foreground objects without any prior knowledge, which is more suited for practical use.

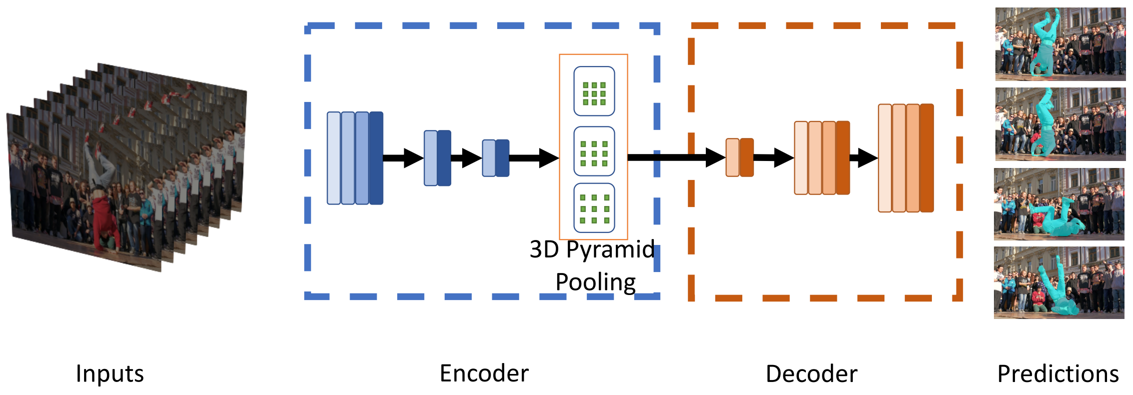

Figure 1.6: Encoder-decoder based video dense prediction pipeline. The encoder module is shown in blue, and decoder modules are in orange. The encoder module of reduces the spatial and temporal size layer by layer; the The 3D pyramid pooling module is able to capture information using multiple receptive fields. The decoder module recovers both spatial and temporal dimensions to the same size as the input clip.

In this thesis, we approach the video object segmentation in the **unsupervised setting** as shown in Figure 1.6. We propose an end-to-end encoder-decoder style 3D CNN based method to solve the video object segmentation problem efficiently. Its encoder is composed of an R2plus1D (R2P1D) network [96] and a 3D pyramid pooling module. Its decoder is designed to recover both spatial and temporal dimensions to generate an output of the same size as the input clip. Instead of the popular two-stream framework, we adopt 3D CNN to aggregate spatial and temporal information. To efficiently process video, we propose 3D separable convolution for the pyramid pooling module and decoder, which dramatically reduces the number of operations while maintaining the performance. Additionally, we also extend our framework to video action segmentation by adding an extra classifier to predict the action label for actors in videos. The main contributions of this section are three-fold:

1. We propose a simple yet efficient 3D CNN framework for action/object segmentation in

videos. To the best of our knowledge, this is the first time 3D CNN has been explored for video object segmentation to simultaneously model the spatial and temporal information in a video.

2. We evaluate our method on video object segmentation and action segmentation benchmarks and demonstrate state-of-the-art performance.

3. We conduct a detailed ablation study to identify the relative contributions of the individual components.

1.5 Organization of The Thesis

The rest of this thesis is organized as follows. In Chapter 2, we review existing literature on video action understanding – recognition, temproal localization and spatio-temporal detection. In Chapter 3, we present our proposed approach for action recognition using discriminative and mutual nearest pairs. In Chapter 4, we describe a novel approach for temoral action localization by discovering sub-actions. In Chapter 5, we present our proposed approach that leverages 3D CNN to generate tube proposals and detect actions. In Chapter 6, we extend 3D-CNN into a encoder-decoder based pipeline to segment the actions and objects in video. In Chapter 7, we draw some concluding remarks and summarize the most valuable results obtained in this thesis, as well as future perspectives.

CHAPTER 2: LITERATURE REVIEW

This chapter reviews a representative subset of the related work in video action understanding and details the key differences between the proposed approaches and existing efforts. First, We present early works on whole clip action video classification with human selected attributes and data-driven mid-level representations. We also discuss transfer learning with mid-level representation. Second, we present previous approaches for temporal action localization with hand-designed or data-driven sub-actions. Moreover, we review different data-driven sub-action discovery approaches and point out our major advantages over the other methods – automatically determining the number and types of sub-actions, most of which are semantically meaningful. Third, we review works on video action understanding with Convolutional Neural Network and R-CNN style detection approaches, in context of 3D tube proposal network and 3D-CNN based action detection pipeline proposed in this thesis. Finally, we present previous works on CNN based semantic segmentation approaches in both image and video.

2.1 Action Recognition

The majority of research on attributes focuses on how semantic attributes can better solve a diverse set of computer vision problems [48, 15, 57, 2, 10, 28] or enable new applications [50, 69]. Generally, specifying these semantic attributes and generating suitable datasets from which to learn attribute classifiers is a difficult task that requires considerable effort and domain expertise. For instance, the popular animals-with-attributes dataset provided by Lampert et al. [50] relied upon a category/attribute matrix that was originally created by domain experts in the cognitive science [68] and AI [42] communities.

Traditionally, semantic attributes are constructed by manually selecting a set of terms that characterize the concept [57, 51]. An important line of work has explored ways to alleviate this human burden. Berg *et al.* [2] propose to automatically discover attributes by mining text and images on the web. Ferrari and Zisserman [17] learn attributes such as "striped" and "spotted" in a weakly supervised scheme from unsegmented images. Parikh and Grauman [71] propose an interactive scheme that efficiently uses annotator feedback.

While all these methods generate human-interpretable semantic attributes, we are motivated to ask whether semantic attributes are necessarily superior to those learned directly from low-level features. Large-scale action recognition dataset, UCF101 [85] has been augmented by a set of category-level attributes, generated using human rater judgments, for the ICCV'13 THUMOS contest on Action Recognition [36]. Section 3.2 discusses that even though the THUMOS semantic attributes encode an additional source of information from human raters, they are significantly outperformed by the automatically learned DaMN features.

2.1.1 Data-driven Category-Level Features

Category-level features (sometimes termed *data-driven attributes*) are mid-level representations that are typically learned from low-level features without manual supervision. Their main drawbacks are that: 1) the distinctions found in the feature space may not correspond in any obvious way to a semantic difference that is visible to humans; 2) unlike the attributes described in last paragraph, which incorporate additional domain knowledge from human raters, it is unclear whether data-driven features should be expected to glean anything from the low-level features that a state-of-the-art classifier employing the same features would fail to extract.

Farhadi et al. [13] discover promising attributes by considering large numbers of random splits of the data. Wang & Mori [104] propose a discriminatively trained joint model for categories and their visual attributes in images. Liu *et al*. [58] combine a set of manually specified attributes with data-driven attributes for action recognition. Yang and Shah [108] propose the use of data-driven concepts for event detection in video. Mensink *et al*. [62] propose an efficient method for generalizing to new categories in image classification by extending the nearest class mean (NCM) classifier that is reminiscent of category-level features. Our work is related to that of Yu *et al*. [111], who independently proposed a principled approach to learning category-level attributes from data by formulating the similarity between two categories based on one-vs-one SVM margins. However, DaMN differs from [111] in several key respects. First, our features are real-valued while theirs (like most attributes in previous work) are binary. Second, our method centers on identifying pairs of strongly-connected categories (mutual near neighbors), while theirs is set up as a Rayleigh quotient optimization problem and solved using a greedy algorithm. Third, their application is in image classification while we are primarily interested in action recognition with a large number of classes. Nonetheless, we show in direct comparisons that DaMN outperforms [111] using their experimental methodology on image datasets.

At a high level, the work most closely related to DaMN is Bergamo & Torresani's Meta-Class features, which also seek to group categories in a data-driven manner in order to improve upon one-vs-rest classifiers. The fundamental difference is in how the two algorithms determine which categories to group: meta-class trains an auxiliary SVM classifier and treats the classification error (computed over a validation set) as a proxy for the similarity between two categories. In contrast, DaMN directly formulates principled measures for this category-level distance, such as the pairwise (one-vs-one) SVM margin. The latter is an improvement both in terms of theory and is experimentally validated in direct comparisons on several datasets.

2.1.2 Knowledge Transfer

Category-level features and attributes are well suited to vision tasks that require trained classifiers to generalize to novel categories for which there exists limited training data. This is because a classifier trained to recognize an attribute such as “furry” from cats and dogs is likely to generalize to recognizing other furry mammals. Attribute-driven knowledge transfer has been successfully demonstrated both in the one-shot [50] and zero-shot [69, 77] context. In the action recognition domain, Liu *et al*. [57] explore attribute-driven generalization for novel actions using manually specified action attributes.

Our approach for generalizing to novel categories is purely data-driven. At a high level, our approach to one-shot learning is related to the use of one-vs-rest classifiers or classemes [94] by which novel categories can be described; DaMN features can be viewed as augmenting one-vs-rest with the most useful subset of potential two-categories-vs-rest classifiers. Our experiments (Section 3.2) confirm that our DaMN category-level features are significantly superior to existing approaches in both cross-dataset and novel category generalization scenarios.

2.2 Temporal Action Localization

The goal of spatio-temporal action localization is to localize actions in space and time simultaneously. In the other hand the goal of temporal localization is to localize action only temporally. This problem is more meaningful in context of untrimmed videos.

2.2.1 Temporal Action Localization in Untrimmed Video

Different from the above approaches, we aim to solve temporal action localization in untrimmed and unconstrained videos. Almost all approaches for spatio-temporal action localization employ trimmed videos. However, untrimmed videos for temporal action localization are much longer and unconstrained, posing significant challenges. Therefore, solving this problem requires us not only to discriminate between actions but also to distinguish actions from the background (which is taken from the same scene).

When only video-level labels are available, weakly supervised learning based approach can be used. Lai *et al*. [49] use multiple instance learning to select the key concepts in untrimmed videos and localize actions temporally. Sun *et al*. [87] use web images as a prior to improve detection performance. However, as large scale action datasets with temporal annotation, such as THUMOS'14 [36], are introduced, researchers have explored approaches based on learning directly from temporally annotated datasets. Wang *et al*. [101] combine the improved dense trajectories (iDTF) [99, 113] as motion features and frame-level CNN as appearance features together. Karaman *et al*. [39] utilize Fisher encoded iDTF with saliency based pooling. Oneata *et al*. [66, 67] fuse motion, visual and audio features to train classifiers and use the classification scores as a contextual feature to help localization. Shou *et al*. [80] propose a loss function considering temporal overlap and learn segment-based 3D ConvNets for localization. Yeung *et al*. [109] train a recurrent neural network to predict the temporal bounds of actions. Heilbron *et al*. [3] propose temporal proposal to locate the actions temporally. However, none of the above mentioned approaches model the temporal structure of an action by a sequence of sub-actions. Gaidon *et al*. [18] propose to model action as a sequence of atomic action units (actoms). However, their approach relies on manually annotated actoms. In contrast, the sub-actions in our approach are automatically discovered. According to [27, 15], data-driven concepts or sub-actions or actoms mainly have two advantages over manually defined concepts. First, a large number of data-driven concepts can be determined as far as they are different in the corresponding feature space, since they do not necessarily have to be semantically different. Second, manually defined concepts need extra knowledge from sophisticated human raters and they must be carefully designed to maximize their usage.

2.2.2 Data-driven Sub-action Discovery

Several works have aimed to automatically discover mid-level representations. Tang *et al.* [89] propose to treat states of temporal segments as latent variables and model durations of states as well as transitions between states using variable-duration HMM. Lan *et al.* [52] automatically discover mid-level action elements by clustering spatio-temporal segments and represent videos by a hierarchy of mid-level action elements. However, our approach differs from them in several key respects. First, the sub-actions discovered by our approach are consistent across different instances of an action and they are semantically meaningful. By contrast, videos of an action may consist of different sets of sub-actions and the sub-actions may not have clear semantic meanings in [89, 52]. Second, the number of sub-actions in our method is automatically discovered, while [89] manually defines the number of states. Third, the distances between sub-actions are taken into consideration in our approach. Thus our model allows temporal overlap or gap between sub-actions, which is more flexible and robust in realistic videos. In contrast, no overlap or gap between states is allowed in [89]. Last, we address temporal action localization problem while [52] attacks the easier whole-clip action recognition problem.

2.3 Action Detection

Convolutional Neural Networks (CNN) have been demonstrated to achieve excellent results for action recognition [54, 64]. Karpathy *et al.*[40] explore various frame-level fusion methods over time. Ng *et al.*[114] use recurrent neural network employing the CNN feature. Since these approaches only use frame based CNN features, the temporal information is neglected. Simonyan *et al.*[82] propose the two-stream CNN approach for action recognition. Besides a classic CNN which takes images as an input, it has a separate network for optical flow. Moreover, Wang *et al.*fuse the trajectories and CNN features. Although these methods, which take hand-crafted temporal feature as a separate stream, show promising performance on action recognition, however, they do not employ end to end deep network and require separate computation of optical flow and optimization of the parameters. 3D CNN is a logical solution to this issue. Ji *et al.*[34] propose a 3D CNN based human detector and head tracker to segment human subjects in videos. Tran *et al.*[95] leverage 3D CNN for large scale action recognition problem. Sun *et al.*[88] propose a factorization of 3D CNN and exploit multiple ways to decompose convolutional kernels. However, to the best of our knowledge, we are the first ones to exploit 3D CNN for action *detection*.

Compared to action recognition, action detection is a more challenging problem [19, 30, 103], which has been an active area of research. Ke *et al.*[41] present an approach for event detection in crowded videos. Tian *et al.*[90] develop Spatio-temporal Deformable Parts Model [16] to detect actions in videos. Jain *et al.*[29] and Soomro *et al.*[84] use supervoxel and selective search to localize the action boundaries. Recently, researchers have leveraged the power of deep learning for action detection. Authors in [23] extract frame-level action proposals using selective search and link them using Viterbi algorithm. While in [105] frame-level action proposals are obtained by EdgeBox and linked by a tracking algorithm. Two-stream R-CNNs for action detection is proposed in [72], where a spatial Region Proposal Network (RPN) and a motion RPN are used to generate frame-level action proposals. However, these deep learning based approaches detect actions by linking frame-level action proposals and treat the spatial and temporal features of a video separately by training two-stream CNN. Therefore, the temporal consistency in videos is not well explored in the network. In contrast, we determine action tube proposals directly from input videos and extract compact and more effective spatio-temporal features using 3D CNN.

For object detection in images, Girshick *et al.*propose Region CNN (R-CNN) [22]. In their approach region proposals are extracted using selective search. Then the candidate regions are warped to a fixed size and fed into ConvNet to extract CNN features. Finally, SVM model is trained for object classification. A fast version of R-CNN, Fast R-CNN, is presented in [21]. Compared to the multi-stage pipeline of R-CNN, fast R-CNN incorporates object classifier in the network and trains object classifier and bounding box regressor simultaneously. Region of interest (RoI) pooling layer is introduced to extract fixed-length feature vectors for bounding boxes with different sizes. Recently, faster R-CNN is proposed in [75]. It introduces a RPN (Region Proposal Network) to replace selective search for proposal generation. RPN shares full image convolutional features with the detection network, thus the proposal generation is almost cost-free. Faster R-CNN achieves state-of-the-art object detection performance while being efficient during testing. Motivated by its high performance, in this paper we explore generalizing faster R-CNN from 2D image regions to 3D video volumes for action detection.

2.4 Video Segmentation

Convolutional neural networks have been demonstrated to achieve excellent results in video action understanding [54, 40, 114]. Video should not be treated as a set of independent frames, since the connection between frames provides extra temporal information for understanding. Simonyan *et al.*[82] propose the two-stream CNN approach for action recognition, which consists of two CNNs taking image and optical flow as input respectively. To avoid computing optical flow separately, Tran *et al.*[95] propose 3D CNN for large scale action recognition. Hara *et al.*[24] apply 3D convolution on ResNet structure. Carreira *et al.*[5] propose I3D by extending inception network from 2D to 3D and including an extra optical flow stream. Tran *et al.*[96] and Xie *et al.*[106] factorize 3D CNN to treat spatial and temporal information separately to reduce the computational cost while keeping the performance. However, to the best of our knowledge, we are the first ones to exploit 3D CNN for video object segmentation.

CNN-based segmentation. The success of CNN-based approaches for image classification [46, 25] have led to dramatic advances in image segmentation [59, 65]. Many of the segmentation approaches leverage recognition models trained on ImageNet and replace the fully-connected layers with 1×1 kernel convolutions to generate dense (pixel-wise) labels. Recently, the encoder-decoder style network architecture, such as SegNet [1] and U-Net [78], has been the mainstream design for semantic segmentation. Moreover, pyramid pooling [115, 9] and dilated convolution [110, 9, 8] are effective techniques to improve segmentation accuracy. Our 3D CNN also build upon the encoder-decoder structure for video object segmentation.

Video object segmentation. Video object segmentation [41, 107] aims to delineate the foreground object(s) from the background in each frame. Some semi-supervised segmentation pipelines [61, 97] assume that the segmentation mask of the first frame in the sequence is given and the object maps in video are obtained by a CNN model. Most of the semi-supervised video object segmentation algorithms exploit temporal consistency in video sequences to propagate the initial segmentation mask to subsequent frames.

In the more challenging unsupervised setting, as we address in this paper, no object mask is provided as initialization during the test phase. Unsupervised segmentation has been attacked by several variants on CNN-based models, such as the two-stream architecture [44, 4], recurrent neural networks [83, 91] and multi-scale feature fusion [92, 83]. These approaches generally perform much better than traditional clustering-based pipelines [7]. The core idea behind such approaches involves leveraging motion cues explicitly (via optical flow) using a two-steam network [31, 92, 91], and/or employing a memory module to capture the evolution of object appearance over time [83, 91].

Action segmentation. Action segmentation provides pixel-level localization for actions (*i.e.*action segmentation maps), which are more accurate than bounding boxes for action localization. Lu *et al.*[60] propose supervoxel hierarchy to enforce the consistency of the human segmentation in video. Gavrilyuk *et al.*[20] infer pixel-level segmentation of an actor and its action in video from a natural language input sentence.

While we take inspiration from these works, we are the first to present a 3D CNN based deep framework for video object segmentation in a fully automatic manner. Moreover, our proposed method is designed with computational efficiency in mind to enable practical applications for video segmentation.

2.5 Summary

In this chapter, we first discussed related works in the area of action recognition, including hand crafted attributes, data driven mid-level features and knowledge transferring. Next, we presented previous temporal action localization framework and various sub-action discovery approaches. Third we reviewed works on CNN based object detection and its generalization from 2D image to 3D video clips. Finally we discussed previous works on CNN based object segmentation in image and video. In the next four chapters, we present our proposed methods on action recognition, temporal action localization, action detection and segmentation.

CHAPTER 3: DAMN – DISCRIMINATIVE AND MUTUALLY NEAREST: EXPLOITING PAIRWISE CATEGORY PROXIMITY FOR VIDEO ACTION RECOGNITION

The objective of action recognition is to classify a video clip into one of the action categories. In this chapter, we propose DaMN, a method for automatically constructing category-level features for multi-class problems by combining one-vs-one, one-vs-rest and two-vs-rest classifiers in a principled manner to solve action recognition problem. The key intuition behind DaMN is that similar actions (e.g., jogging vs. running) are often poorly represented by one-vs-rest classifiers. Rather than requiring methods to accurately split such "Siamese Twin" categories, we choose to augment one-vs-rest classifiers with a *judicious selection* of two-vs-rest classifiers that keep the strongly related categories together. The challenge is how best to identify such categories and then how to combine information from the different classifiers in a principled manner.

3.1 Proposed Approach: DaMN

Figure 3.1 provides a visual overview of the DaMN process, detailed further in this section. Action recognition in video is typically formulated as a classification problem where the goal is to assign a category label $y \in \mathcal{Y}$ to each video clip $v \in \mathcal{V}$. Although our proposed method also works with more complex feature families, let us simplify the following discussion by assuming that the given video v is represented by some feature $\mathbf{x}_v \in \mathbb{R}^d$.

Let $n = |\mathcal{Y}|$ denote the number of categories and $\mathcal{F}$ the set of category-level features that describe how categories are related. The size of this set, $m = |\mathcal{F}|$; $m = n$ in the case of one-vs-rest classifiers, and $m > n$ for DaMN — with the number of two-vs-rest classifiers given by $m - n$. In general, the category-level relationships can be expressed by a binary-valued matrix $\mathbf{B} \in \mathbb{B}^{n \times m}$. The ith row of $\mathbf{B}$ contains the features for category y_i while the jth column of the matrix denotes the categories that share a given category-level feature. Each column is associated with a classifier, $f_j(\mathbf{x}_v)$ (such as a kernel SVM), trained either one-vs-rest or two-vs-rest, that operates on a given instance (video clip) v.

Our goal is to automatically identify the set of suitable features $\mathcal{F}$, train their associated classifiers $f(.)$ using the available training data and use the instance-level predictions to classify novel videos. In the additional task of one-shot learning, we use the same category-level features with an expanded set of categories, $\mathcal{Y}'$, for each of which we are given only a single exemplar.

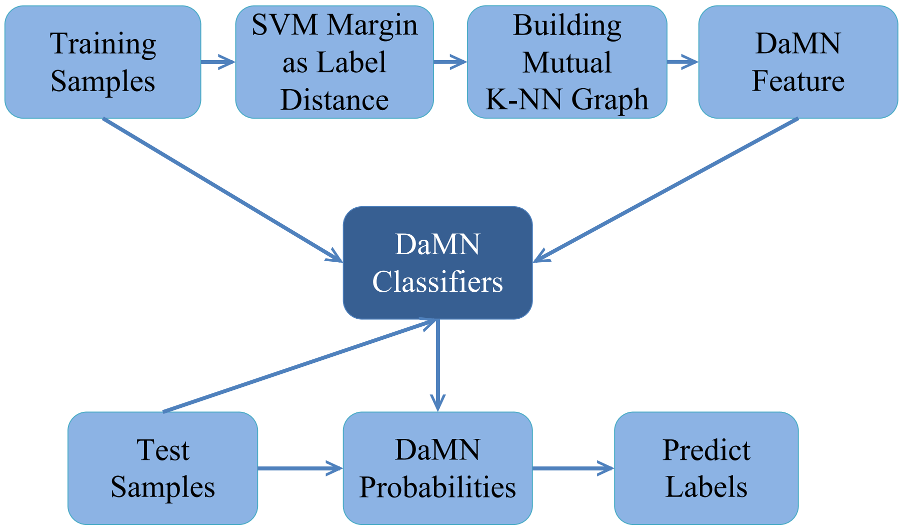

Figure 3.1: Overview of the DaMN Learning/Prediction Process.

3.1.1 Pairwise Category Similarity

We train one-vs-one SVM classifiers for every pair of categories to compute the category-level similarity. We propose a natural generalization from the instance-level similarity, typically expressed through some kernel function $K(.,.)$ that compares their respective low-level feature representation (e.g., using χ^2 for bag-of-words features) typically used in action recognition to category-level similarity as the margin of a one-vs-one SVM trained to discriminate the two categories, expressed in the dual form as

$$d_{y_i,y_j} = \sum_{\forall (p,q) \in y_i \cup y_j} \alpha_p \alpha_q (-1)^{I[c(p) \neq c(q)]} K(\mathbf{x}_p, \mathbf{x}_q) \tag{3.1}$$

where p and q are all pairs of instances from the union of the two categories, α their non-negative weights (> 0 only for support vectors), $c(.)$ is a function that returns the category of the given instance and $I[.]$ denotes the indicator function whose value is 1 when its argument is true and 0 otherwise. Figure 3.2 visualizes the similarity values computed using this metric for UCF101; to manage visual clutter, we show only the first 43 categories, with edges shaded according to the pairwise similarity between respective categories. Note that even though there are $\binom{n}{2}$ such classifiers, they are quick to train since each classifier uses only a tiny subset of the training data — a fact exploited by one-vs-one multi-class SVMs.

Given similarities between categories (at the feature level), we seek to identify pairs of categories that are close. For this, we construct a mutual k-nearest neighbor (kNN) graph over categories. A mutual kNN graph is similar to the popular kNN graph except that nodes p and q are connected if and only if p lists q as among its k closest neighbors and q also lists p among its k closest neighbors. Let $G = (V, E)$ be a graph with n nodes, V, corresponding to the category labels y and weights given by:

$$w_{pq} = \begin{cases} 0 & y_p \text{ and } y_q \text{ are not mutual } k\text{NN}, \\ d_{y_p,y_q} & \text{otherwise.} \end{cases} \tag{3.2}$$

Unlike a k-nearest neighbor graph, where every node has degree k, a mutual kNN graph exhibits much sparser connectivity.

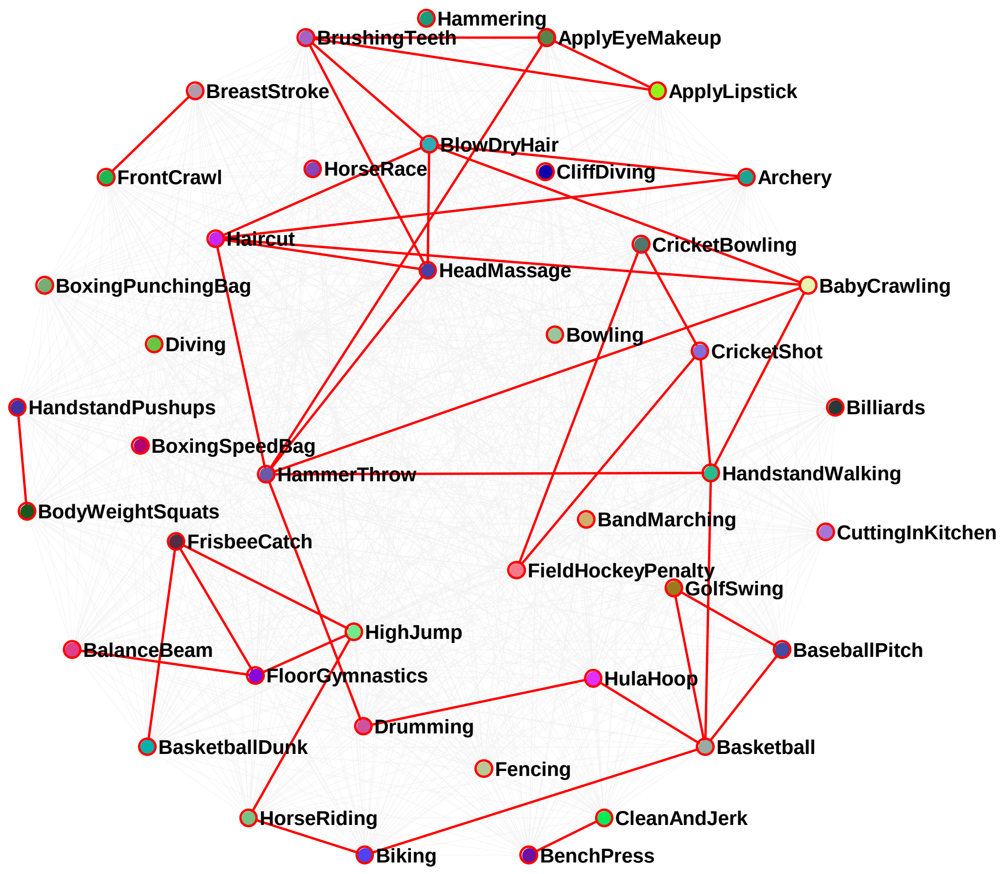

Figure 3.2: Examples of DaMN category pairs automatically discovered in UCF101.

3.1.2 Constructing the DaMN Category-Level Feature Matrix

The DaMN category-level feature matrix is composed of two parts: 1) features to separate each category individually (identical to one-vs-rest); 2) pair classifiers designed to separate mutually proximate pairs of categories from the remaining classes. Among the $\binom{n}{2}$ possible category pairs, DaMN selects only those that are mutually proximate ($w_{pq} > 0$) as additional features.

Thus, $\mathbf{B}$ is an $n \times m$ matrix composed of two portions: an $n \times n$ identity matrix corresponding to the one-vs-rest classifiers augmented by additional columns for the selected category pairs. Each of the additional columns c in the second block is associated with a category pair (p, q) and the entries of the new column are given by:

$$B_{ic} = \begin{cases} 1 & \text{if } i = p \text{ or } i = q \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

3.1.3 Training Category-Level Feature Classifiers

Each column in $\mathbf{B}$ defines a partition over labels that is used to split the training set into positives (those instances whose labels possess the given feature) and negatives (instances lacking the feature). We learn a χ^2 kernel SVM using this data split to predict the new feature.

Since the first n columns of $\mathbf{B}$ have a single non-zero entry, they correspond to training regular one-vs-rest SVMs. The remaining $m - n$ columns have two non-zero entries and require training a set of somewhat unusual two-category-vs-rest SVMs to separate similar category pairs from the other classes.

3.1.4 Predicting Categories

Given a video v from the test set, we extract its low-level features x_v and pass it through the trained bank of DaMN classifiers, both one-vs-rest and two-vs-rest. We also have a set of one-vs-one classifiers to distinguish between the two categories in one DaMN pair. For the given video v, let $P(y_i)$ denote the probabilistic score returned by the one-vs-rest classifier of category y_i, and $P(y_i \oplus y_j)$ represent the score computed by the two-vs-rest classifier of the DaMN pair (y_i, y_j). Also, let $P(y_i | y_i \oplus y_j)$ denote the score returned by the one-vs-one classifier which distinguishes between the categories y_i and y_j. All of the SVM classifier scores are Platt-scaled to provide probabilistic values.

We now compute a score that quantifies the match between v and each candidate category y_i by combining the traditional one-vs-rest score with those two-vs-rest scores that involve y_i, weighted by a one-vs-one classifier between y_i and the other class in the DaMN pair. All of these individual probabilities are then combined to obtain the final score for category y_i:

$$T_{y_i} = \frac{P(y_i) + \sum_{\{j:(y_i,y_j)\in \text{DaMN}\}} P(y_i \oplus y_j) \times P(y_i | y_i \oplus y_j)}{\sum_{c=1}^{m} B_{ic}}. \tag{3.4}$$

The argument of the summation in the numerator is equivalent to the following probabilistic rule for finding the probability of event a: $P(a) = P(a \cup b) \times P(a|a \cup b)$. The set $\{j : (y_i, y_j) \in \text{DaMN}\}$ represents the DaMN pairs that involve the category y_i. Since there are several of such pairs, the score T_{y_i} is defined as the mean of the scores acquired from all of those pairs as well as the score of the one-vs-rest classifier (i.e., $P(y_i)$). Therefore, the denominator of Eq. 3.4 is equal to the total number of DaMN pairs that involve category y_i plus one (to count for its one-vs-rest classifier).

Since, as seen in Figure 1.2 (right), some categories participate in many more DaMN pairs than others, the final score for such categories involves more terms. At the other extreme are categories that do not show up in any DaMN pair, for which only the one-vs-rest score is used; this situation corresponds to a highly distinctive category, which is easily discriminated from others (as evidenced by a high margin in kernel space). An intuitive illustration is that when computing a score for v with respect to the category *running*, we should focus on DaMN features generated by pairs like *jogging-running* or *walking-running* rather than those seeded by *makeup-shaving pairs*.

Finally, we assign the test video to the category with the highest match score:

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} (T_y). \tag{3.5}$$

3.1.5 Knowledge Transfer to Novel Categories

This section details how DaMN features enable generalization to novel categories through knowledge transfer. Consider the scenario where m features have been generated using abundant training data from n categories, resulting in an $m \times n$ binary matrix and an $n \times n$ adjacency matrix describing the weighted mutual kNN graph. We are now presented with n' novel categories, each containing only a few instances. The goal is to classify instances in the test set generated from all $n + n'$ categories (both existing and novel).

We proceed as follows. First, for each new label y', we determine its similarity to the known categories $\mathcal{Y}$ using the small amount of new data and Equation 3.1. Note that while the data from novel categories may be insufficient to enable accurate training of one-vs-rest classifiers, it is sufficient to provide a rough estimate of similarity between the new category and existing categories. From such rough estimate, we determine the k categories that are most similar to y' and synthesize a new row for the matrix $\mathbf{B}$ for the novel category from the rows of similar categories:

$$B_{y'j} = \sum_{y \in k\text{NN}(y')} B_{yj}, \tag{3.6}$$

where $\sum$ is treated as the OR operator since $\mathbf{B}$ is a binary-valued matrix and kNN$(.)$ returns the k categories most similar to the novel category.

At test time, we obtain scores from the bank of existing DaMN feature classifiers and determine the most likely category using Equations 3.4 and 3.5. Note that for the novel categories, Equation 3.4 employs the synthesized values in the matrix.

3.2 Experiments

Our experimental methodology focuses on clarity, thoroughness and reproducibility rather than tuning our system to squeeze the best possible results. Even under these conditions, as detailed below, DaMN generates the best classification results to date on both the UCF101 and HMDB51 datasets.

All of our experiments are conducted on the two latest publicly available action recognition datasets, UCF101 [85] and HMDB51 [47], which contain YouTube videos from 101 and 51 categories, respectively. For UCF101 experiments, we closely follow the protocol specified in the ICCV'13 THUMOS action recognition challenge [36], for which test/train splits, manually generated category-level semantic attributes and baseline results are provided.

We restrict ourselves to low-level features for which open source code is available, and select the Dense Trajectory Features (DTF) [98] and Improved Dense Trajectory Features (I-DTF) [99] based on their state-of-the-art results reported in recent competitions. DTF consists of four descriptors: histogram-of-gradients (HOG), histogram-of-flow (HOF), motion-based histograms (MBH) and trajectories. The descriptors are traditionally combined using early fusion (concatenation), but we also present results using the individual component descriptors and late fusion (using equally weighted combination of SVMs). For each descriptor, we generate a 4000-word codebook and build a standard bag-of-words representation for each video by aggregating bin counts over the entire clip.

For classification, we employ the popular LIBSVM [6] implementation of support vector machines with C=1 and the χ^2 kernel, since our features are histograms. Because DaMN already employs one-vs-one SVMs for determining category proximity, results from multi-class one-vs-one SVMs serve as a natural baseline. Following Yu et al. [111], we also train 101 one-vs-rest SVM classifiers to serve as a stronger baseline.[1]

Implementing DaMN is straightforward, but we provide open-source code to enable the research community to easily duplicate our experiments and to employ DaMN in other domains.

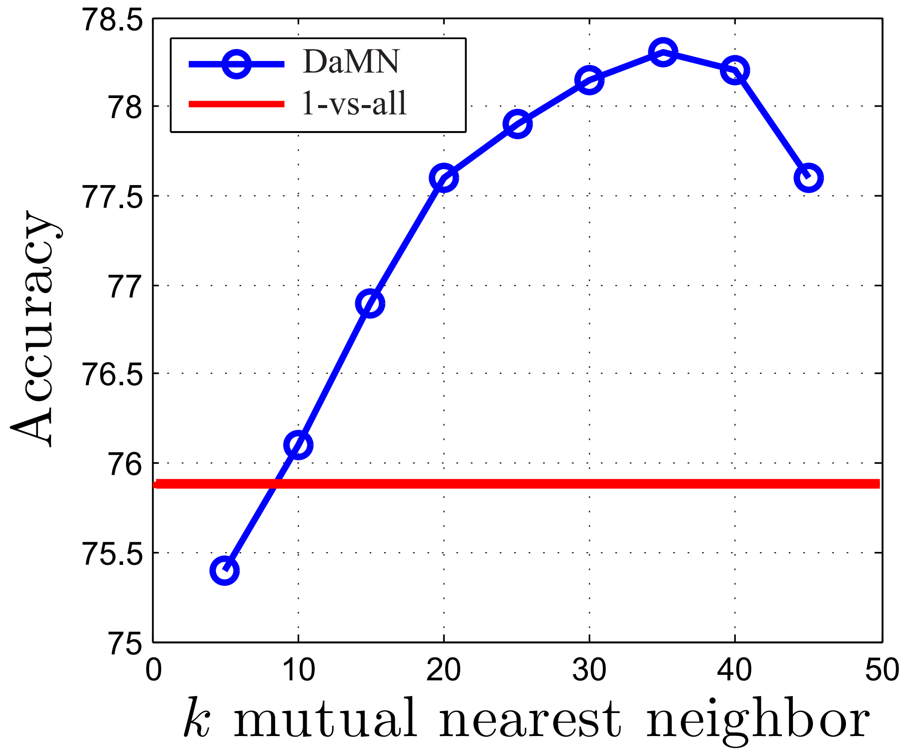

Figure 3.3: DaMN is tuned using a single hyper-parameter, k. Accuracy on UCF101 with DTF (late fusion) as k is varied. DaMN outperforms the strongest baseline for $k > 15$ and even at its worst, DaMN outperforms all remaining methods.

[1]Others (e.g., the majority of the submissions to ICCV'13 THUMOS Challenge [36]) report that one-vs-rest SVMs consistently outperform multi-class one-vs-one SVMs for reasons that are not clearly understood; an observation that merits further study.

3.2.1 Action Recognition on UCF101 and HMDB51

All DaMN category-level features for UCF101 were generated with the single hyper-parameter k=35 (see below for experiments showing sensitivity to choice of k). Following the ICCV'13 THUMOS [36] protocol, we present results averaged over the 3 provided folds. The small number next to the mean accuracy in the tables is the standard deviation of the accuracy over the 3 folds. Table 3.1 summarizes the action recognition results. We present direct comparisons against a variety of baselines as well as DaMN's competitors, one-vs-rest and meta-class. Meta-class requires a validation set and cannot train on all of the available data; to eliminate the possibility that DaMN performs better solely due to the additional data, we provide additional rows (denoted DaMN^{-} and one-vs-rest^{-}) where these algorithms were trained on a reduced dataset (instances in the validation set used by Meta-class are simply discarded). We also provide two baselines from the THUMOS contest: 1) the THUMOS contest baseline using STIP [53] + bag-of-words + χ^2 kernel SVM, and 2) a baseline computed using the manually generated THUMOS semantic attributes for UCF101. For the latter baseline, we employ the same methodology as DaMN to ensure a fair direct comparison. We make the following observations.

First, we note that switching from STIP to DTF features alone generates a large improvement over the THUMOS baseline and combining DTF components using late fusion (LF) is generally better than with the early fusion (EF) originally employed by the DTF authors. These are consistent with the results reported by many groups at the ICCV THUMOS workshop and are not claimed as a contribution. We simply note that STIP features are no longer a strong baseline for future work in this area.

Second, we observe that the manually generated THUMOS semantic features are outperformed by all of the methods. This drives a more detailed investigation (see Section 3.2.2).

Table 3.1: UCF101 Results. DaMN consistently boosts results across all features. DaMN achieves the best reported results on UCF101 using I-DTF.

	I-DTF [99]	DTF (LF)	DTF (EF)	MBH	HOG	HOF	Traj.
DaMN	**87.00**±1.1	**78.33**±1.7	**75.93**±1.8	**73.25**±1.3	**57.60**±0.6	**57.42**±2.1	**55.18**±1.6
1-vs-rest	85.90±1.2	75.88±2.4	74.60±2.5	71.32±2.9	56.94±2.3	56.08±3.1	53.72±1.6
1-vs-1	79.12±1.9	69.25±3.3	68.32±3.6	66.00±2.4	51.40±3.2	51.80±2.3	48.63±1.1
DaMN^{-}	**80.03**±0.4	**71.82**±1.4	**70.04**±1.5	**66.73**±1.1	**51.63**±0.7	**49.83**±1.9	**49.74**±1.5
Meta-class	78.65±0.6	69.71±1.8	68.32±1.4	60.07±2.3	44.15±2.6	44.98±0.8	43.91±1.0
1-vs-rest^{-}	78.54±0.6	67.84±1.9	66.91±2.1	62.34±2.2	43.71±1.6	44.93±1.4	43.71±1.3
Semantic	58.99	50.19	49.73	51.56	32.68	43.77	33.85
THUMOS [36] baseline (STIP + BOW + χ^2 SVM): 43.9%							

Table 3.2: HMDB51 Results. DaMN achieves the best reported results on this dataset.

	DaMN	1-vs-rest	Meta-class
I-DTF	**57.88**±0.46	57.01±1.44	57.36±0.24

Third, we note that the DaMN features in conjunction with *any* of the component features (either individual or fused) provide a consistent boost over both one-vs-rest or meta-class, regardless of experimental condition. In particular, DaMN outperforms meta-class even after discarding a portion of the data (which meta-class employs for estimating category-level similarity). Interestingly, meta-class outperforms one-vs-rest only when one-vs-rest is not given access to the full data (one-vs-rest^{-}); this demonstrates that, unlike DaMN, meta-class makes inefficient use of the available data and is not a recommended technique unless there is an abundance of training data. This additional training data boosts DaMN's accuracy by a further 6% in the late-fused DTF condition (DaMN vs. DaMN^{-}), which convincingly shows the benefits of the proposed approach over previous methods.

As discussed in Section 3.1, DaMN only employs a single hyper-parameter. Figure 3.3 shows how UCF101 classification accuracy varies with k using the DTF features (late fusion). We observe that DaMN is better than the strongest baseline after k=15 and peaks on UCF101 around k=35. Even the worst instance of DaMN (k=5) is better than all of the remaining methods.

Table 3.2 shows action recognition results on HMDB51 using its standard splits [47]. The selected parameters for DaMN and the baselines for this dataset were the same as for UCF101. DaMN achieves state-of-the-art results on HMDB51, outperforming the recent results [99] that employ one-vs-rest SVM on Improved DTF features.

3.2.2 DaMN vs. THUMOS semantic attributes

To be of practical use, attributes need to be semantic as well as machine-detectable. The THUMOS attributes clearly capture the first criterion, since they were developed by human raters. However, since they are category-level there is a danger that the attributes envisioned by the raters may not actually be visible (or reliably detectable) in a given instance from that category.

Table 3.3: Performance vs. THUMOS semantic attributes.

	mAP	StD	Min	Max
THUMOS	0.53	0.19	0.07	99.08
DaMN	**0.64**	0.18	0.12	95.74

The accuracy of an attribute classifier captures the reliability with which the given attribute can be recognized in data. Figure 3.4 plots histograms of accuracy for THUMOS and DaMN classifiers and Table 3.3 summarizes some key statistics.

Table 3.5 examines a natural question: how do different choices for the category-level distances illustrated in Figure 3.5 impact DaMN's performance. We briefly describe the choices. Given the

Table 3.4: Marginal benefits obtained by adding three-vs-rest classifiers to DaMN.

	DTF (LF)	DTF (EF)	MBH	HOG	HOF	Traj.
pairs	**78.33**	75.93	**73.25**	**57.60**	**57.42**	55.18
triples	77.88	**76.30**	73.04	57.51	57.20	**55.28**

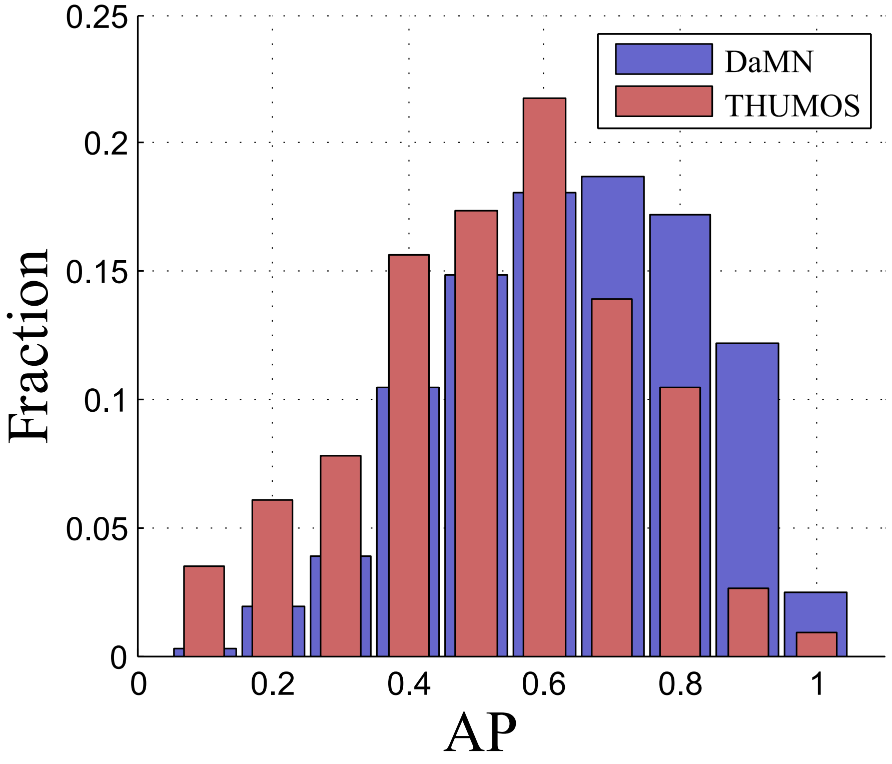

Figure 3.4: Histogram showing the fraction of classifiers that achieve a given AP on UCF101. DaMN features can be detected more reliably than the manually prescribed attributes provided by THUMOS.

set of distances $\{d_{ij} : i \in I, j \in J\}$ for kernel distances between instances in categories I and J, *linkage* is defined as the minimum distance; *median* is the median distance between pairs; *average* is the mean over the distances in this set; and *centroid* is the distance between the centroids of each category (in feature space). *SVM Margin* is the margin for an one-vs-one SVM trained on instances from each category. We see that the SVM Margin is better for almost all features, is robust to outliers and is thus the default choice for DaMN in this paper. We observe that the

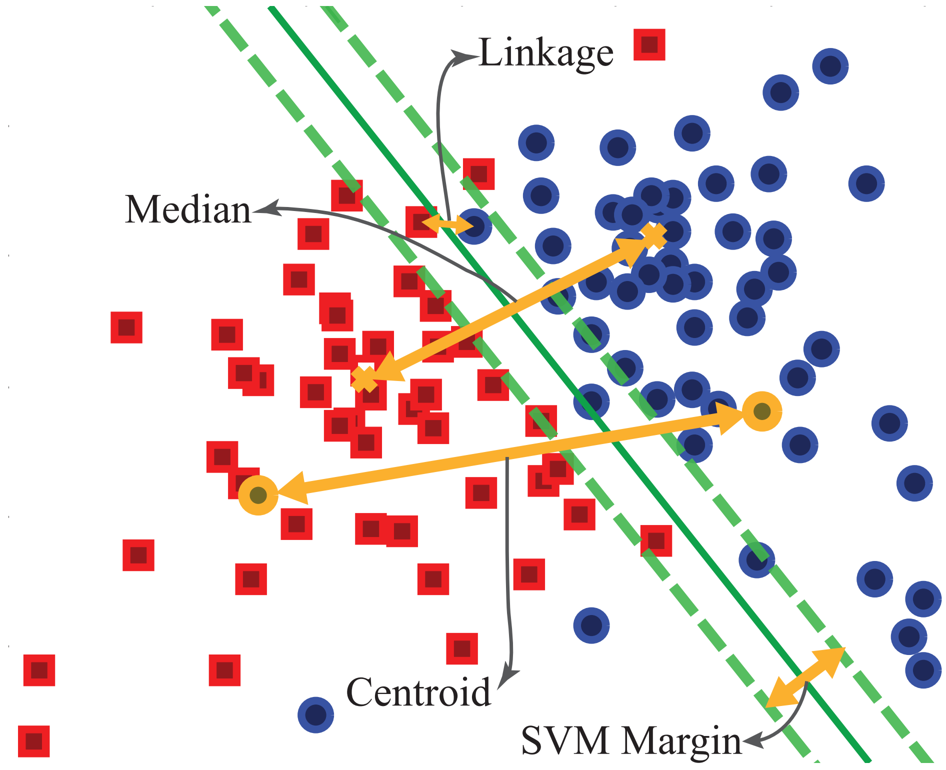

Figure 3.5: Illustration of different category-level distance metrics. The illustration uses a linear kernel only to simplify visualization; all of these, except for centroid, are actually performed in kernel space.

DaMN classifiers are more reliable than the THUMOS ones. The DaMN features were designed to identify pairs of categories that are mutually close (i.e., share an attribute) and to separate them from the remaining categories. As confirmed by these statistics, such a data-driven strategy enables very accurate recognition of attributes. Fortuitously, as shown in Figures 1.2 and 1.1, the selected pairs are also usually (but not always) meaningful to humans. Our results confirm that any price that we may pay in terms of human interpretability is more than offset by the gains we observe in recognition accuracy, both at the attribute and the category level.

Table 3.5: Empirical evaluation of different category-level distance metrics (in the feature space projected using χ^2 kernel).

	DTF (LF)	MBH	HOG	HOF	Traj.
SVM Margin	**78.33**±1.7	**73.25**±1.3	**57.60**±0.6	57.42±2.1	**55.18**±1.6
Average	77.59±1.6	72.84±1.7	57.41±0.8	57.20±1.6	53.82±0.7
Linkage (Min)	77.38±1.8	72.66±1.1	57.12±0.3	57.05±1.8	55.03±1.4
Median	77.79±1.7	72.97±1.5	57.49±0.8	**57.50**±1.7	54.81±1.7
Centroid	77.22±1.9	72.64±1.5	57.49±0.8	57.00±2.0	54.60±1.6

In the remaining experiments, we evaluate how well DaMN enables knowledge transfer, both within and across datasets.

3.2.3 Cross-Dataset Generalization to HMDB51

The focus of this experiment is to see how well the DaMN category-level features learned on UCF101 enable us to perform action recognition on a subset of 12 HMDB51 categories with no additional training. Since the category names are different and UCF101 categories more fine-grained, we roughly align them as shown in Table 3.6.

We follow the same experimental settings as in Section 3.2.1 but use just the MBH feature rather than late fusion for simplicity. We perform experiments using 3-fold cross-validation, with each run training on two folds of UCF101 and testing on the third fold of HMDB51. Table 3.6 shows results averaged over these three folds. We see that DaMN achieves an overall higher accuracy than both one-vs-rest and meta-class on cross-dataset generalization.

Table 3.6: Cross-Dataset Generalization: UCF101 $\rightarrow$ HMDB51

HMDB51 label	UCF101 IDs	DaMN	1-vs-rest	Meta-class
Brush hair	13	**57.78**	40.00	54.44
Climb	74 & 75	74.44	**83.33**	75.56
Dive	26	**77.78**	57.78	70.00
Golf	33	**66.67**	**66.67**	68.89
Handstand	30	43.33	**45.56**	41.11
Pullup	70	58.89	**68.89**	56.67
Punch	17 & 18	**81.11**	80.00	72.22
Pushup	72	**62.22**	56.67	54.44
Ride bike	11	72.22	**73.33**	67.78
Shoot ball	8	22.22	25.56	**30.00**
Shoot bow	3	**38.89**	36.67	**43.33**
Throw	7	**57.78**	**57.78**	54.44
Average		**59.44**$\pm$0.7	57.69$\pm$0.8	54.44$\pm$1.1

3.2.4 Generalization Performance with Limited Training Data

A popular use for attributes and category-level features is that they enable generalization to novel classes for which we have small amounts of training data; an extreme case of this is one-shot learning, where only a single exemplar is provided for each novel category [14].

For this experiment, we randomly select 10 categories to serve as "novel" and treat the remaining 91 as "known". Results are averaged on the three folds specified by THUMOS. We learn DaMN features (k=30) and train semantic and classeme (one-vs-rest) classifiers using the data from two folds of the known classes; we test on the entire third fold of the novel categories. As in cross-dataset scenario, all classifiers use the MBH feature rather than late fusion for simplicity.

We vary the number of training instances per novel category from 1 to 18, while ensuring that all three methods are given identical data. Figure 3.6 summarizes the results (averaged over three folds). As in the UCF101 action recognition experiments (Section 3.2.1), the semantic attributes perform very poorly and DaMN does the best, outperforming both meta-class and one-vs-rest in every trial.

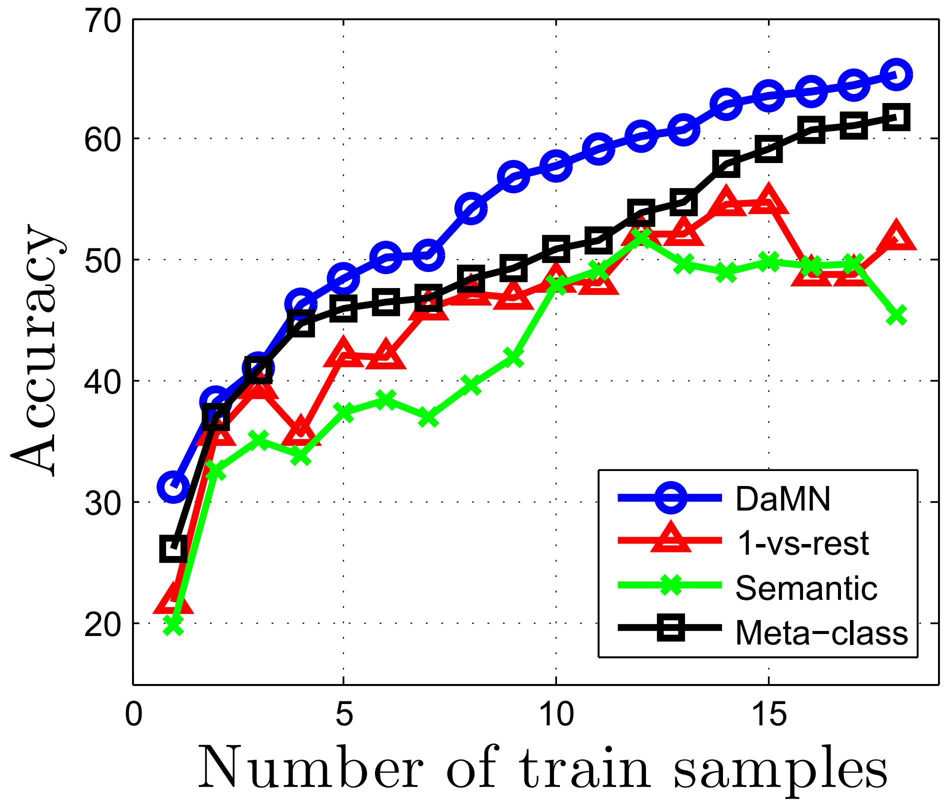

Figure 3.6: Knowledge transfer to novel categories with limited data (1 to 18 instances per category). DaMN consistently outperforms one-vs-rest, meta-class and THUMOS semantic attributes.

The difference between DaMN and the next best is much greater in this experiment, often more than 20%. This demonstrates that the information captured in DaMN generalizes much better across classes.

3.2.5 Extending DaMN to Image Datasets

Although DaMN was primarily developed for applications in action recognition, we recognize that the algorithm can be applied to any classification task. In order to demonstrate the generality of our method, we present a direct comparison against Yu et al.'s recently published method [111] for eliciting category-level attributes on the Animals with Attributes images dataset [50].

Figure 3.7 shows the accuracy of DaMN, one-vs-rest and Yu et al. [111] using the experimental methodology and data splits prescribed in [111]. We see that one-vs-rest and Yu et al. perform similarly, but both are consistently outperformed by DaMN.

3.2.6 Extending DaMN Beyond Pairs

Just as DaMN improves over one-vs-rest by judiciously mining suitable two-vs-rest classifiers, it is natural to explore whether one can obtain additional benefits by considering higher-level information, such as cliques consisting of 3 (or more) mutually near categories. Table 3.3 shows that the improvements are marginal. We believe that this is due to several reasons: 1) there are relatively few such higher-order category groupings; 2) many of the relationships are already captured by the DaMN features since a triplet of similar categories also generates three two-vs-rest pairs; 3) the additional complexity of discriminating within the higher-order grouping of categories may not be merited, whereas the one-vs-one classifiers used to differentiate within the pair were already trained during the DaMN pair selection process. For these reasons, we do not extend DaMN beyond pair categories.

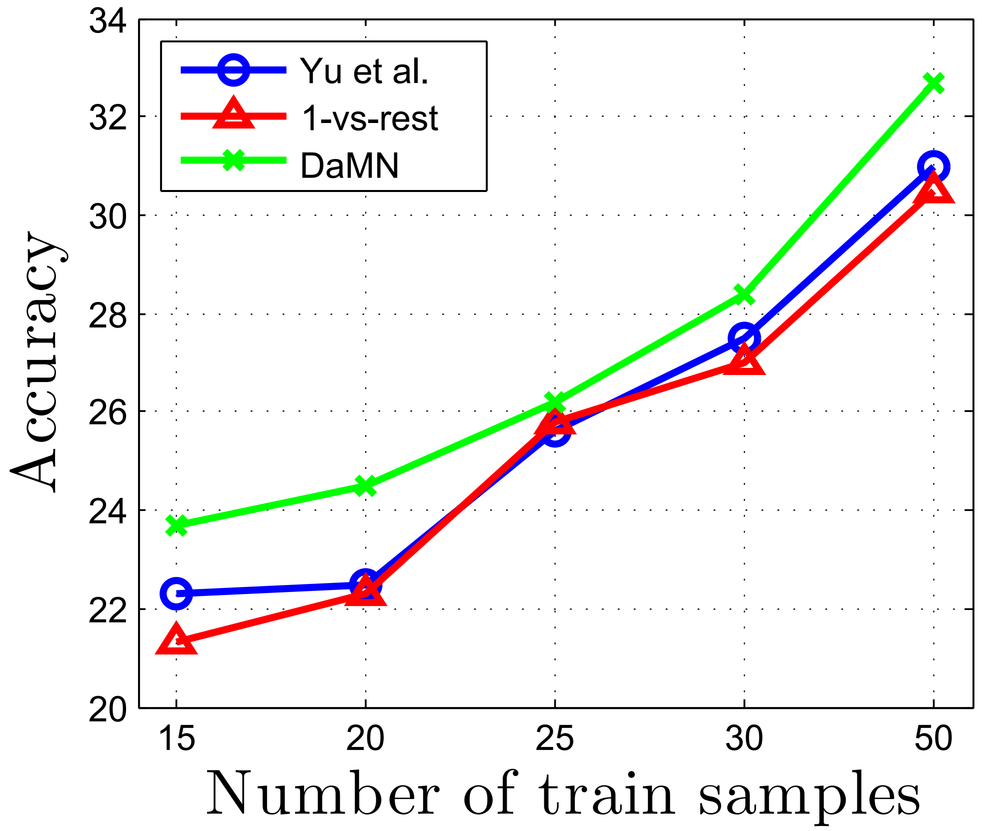

Figure 3.7: Generalization of DaMN to image datasets. DaMN outperforms one-vs-rest and Yu et al. [111] on the Animals with Attributes dataset.

3.2.7 Computational Complexity

While DaMN may seem to be prohibitively more expensive in terms of computation than the popular one-vs-rest classifiers employed in the action recognition community, we show that training these additional classifiers is both computationally manageable and worthwhile since they generate such consistent (if small) improvements in mAP and classification accuracy. In the interests of space, we briefly summarize wall-clock times for the UCF-101 experiments. Total time for one-vs-one SVM training: 10.14s; one-vs-one SVM testing: 1682.72s; total one-vs-rest and two-vs-rest SVM training: 2752.74s; one-vs-rest and two-vs-rest testing: 2546.51s. For each descriptor type in DTF, DaMN employs $\sim$800 one-vs-one SVMs, $\sim$800 two-vs-rest SVMs and 101 one-vs-rest SVMs. Training time for the two-vs-rest classifiers is similar to the one-vs-rest since they have the same number of training instances, but with different labels.

3.3 Summary

In this chapter, we present discriminative and mutually nearest (DaMN) features for action recognition. DaMN is a novel, general-purpose and fully automated algorithm that generates discriminative category-level features directly from data. It automatically identifies similar category pairs based on mutual pairwise proximity in the feature space.

DaMN is a strong choice for mid-level action representation, enabling us to obtain the highest-reported results on the UCF 101 and HMDB 51 dataset. Besides that, DaMN also outperforms the manually generated category-level attributes provided in the UCF 101 dataset. It demonstrates the advantages of data driven mid-level representations over hand designed attributes.

While our focus is on video action recognition, the proposed method can also be applied to other problems, such as knowledge transfer. It can work both across dataset and on novel categories with limited training data and we show DaMN's superiority on the Animals-with-Attributes dataset.

DaMN works well when input video is well trimmed. However, most of the real world videos are long and contain multiple scenes and actions. It is desirable to divide a long video into several clips and recognize each clip to decide whether it contains an action. In the next chapter, we present a novel approach for video temporal action localization.

CHAPTER 4: REAL-TIME TEMPORAL ACTION LOCALIZATION IN UNTRIMMED VIDEOS BY SUB-ACTION DISCOVERY

Compared to action recognition, temporal action localization in video is a more challenging task. Its objective is to recognize actions in untrimmed long videos in the wild instead of in manually trimmed short videos. Temporal action localization is an important topic because most of the real-world videos are long and untrimmed, in which there are multiple action instances as well as background scenes. Temporally localizing actions in long videos, such as in surveillance videos, could save extensive human labor and time.

4.1 Proposed Approach

Temporal action localization can be formulated as a classification problem where the goal is to assign action labels to each frame, and determine the beginning and ending frames of an action in untrimmed videos. Traditionally, an action has been represented by a single model, which is simple and efficient, nevertheless it is not robust to intra-class variation (in particular in context of untrimmed videos), leading to unsatisfactory performance.

An important fact about actions is that they are usually composed of multiple semantic sub-actions (Fig. 1.4). While the sub-actions may vary in appearance and duration (e.g., the length of the "approach" run in the "long jump" action), a given action nearly always consists of the same set of sub-actions in a consistent order. Thus, we choose to model an action as a series of sequential sub-actions and train a separate classifier for each sub-action.

An important issue, in context of modeling an action using sub-actions, is how to determine the number of sub-actions for each action. One obvious solution is to manually identify a set of sub-actions for each action and generate training sets by annotating each sub-action in every video; that would be a daunting task. Instead, we propose an automatic method to discover sub-actions for each action. Our approach for discovering sub-actions consists of three main steps. First, temporal segments of all training videos of an action are clustered into different parts. Second, similar parts are merged to obtain candidate sub-actions. Finally, boundaries between candidate sub-actions are adjusted to obtain final sub-actions. Sub-actions discovered in this way are consistent and semantically meaningful.

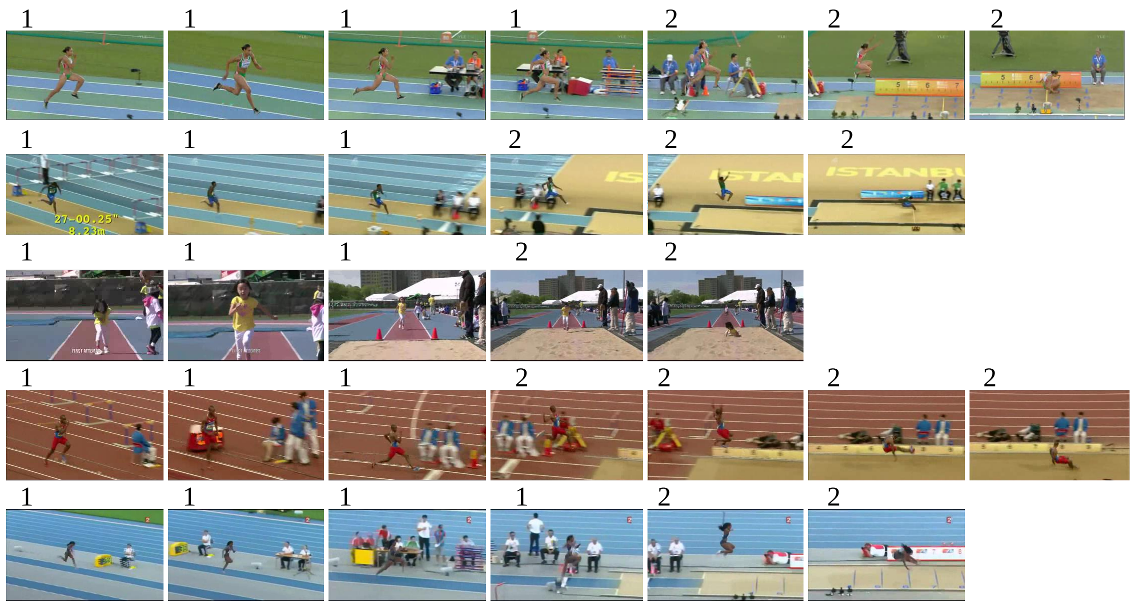

Figure 4.1: An example of segmenting "high jump" action into several sub-actions. Rows represent different length videos of the same action. Temporal segments within a video are represented by key frames. The number on the top of a frame represents the ground truth index of sub-action in the action. In this action there are two sub-actions.

Assume a video v is composed of temporal segments of fixed length. Each temporal segment $i \in v$, is represented by a Fisher vector $x_i \in \mathbb{R}^d$ computed over the features, which is used as an input to our model. Let n denotes the number of actions and m_l the number of sub-actions for action l $(l \in \{1, \ldots, n\})$. And let a n dimensional one-hot vector $g_i \in \mathbb{R}^n$ denotes the action labels for segment i. If segment i does not belong to action l then $g_i(l)$= 0. Otherwise, segment i contains sub-action $g_i(l)$ of action l.

4.2 Discovering Sub-actions

Our key assumption is that all the video clips of an action l share the same sequence of m_l sub-actions. The goal is to design an approach that can automatically find the appropriate number of sub-actions for each action in an unsupervised manner. Sub-actions should correspond to different semantic parts and be consistent in videos clips of the same action. Moreover, the sub-actions in an action should occur in a specific order.

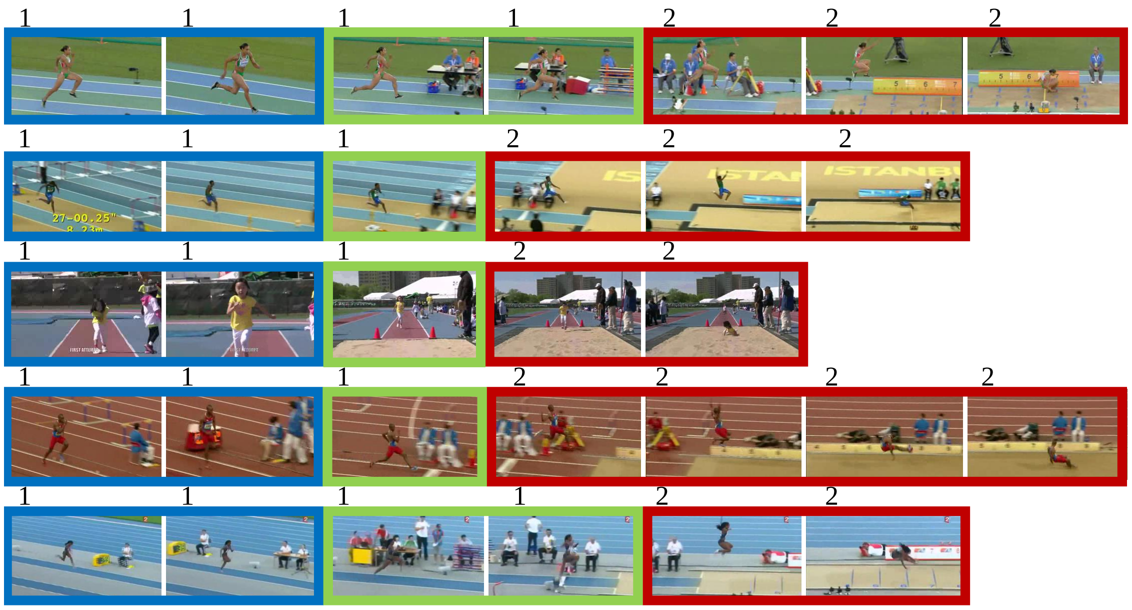

Figure 4.2: In the first step, all segments of each video of an action are clustered into k_l (in this case 3) sequential parts, which are shown by borders of different colors (blue, green and red). However, as can be seen that the first sub-action is broken into two parts.

4.2.1 Clustering Segments Into Parts

Since the number of sub-actions in an action is unknown, we first cluster segments in each video of an action l into k_l parts (we use k_l equal to half of the number of segments in the shortest training video of action l) to serve as candidate sub-actions. These candidate sub-actions are updated and adjusted later to select the most discriminative sub-actions. Below, we describe the procedure of clustering segments into parts.

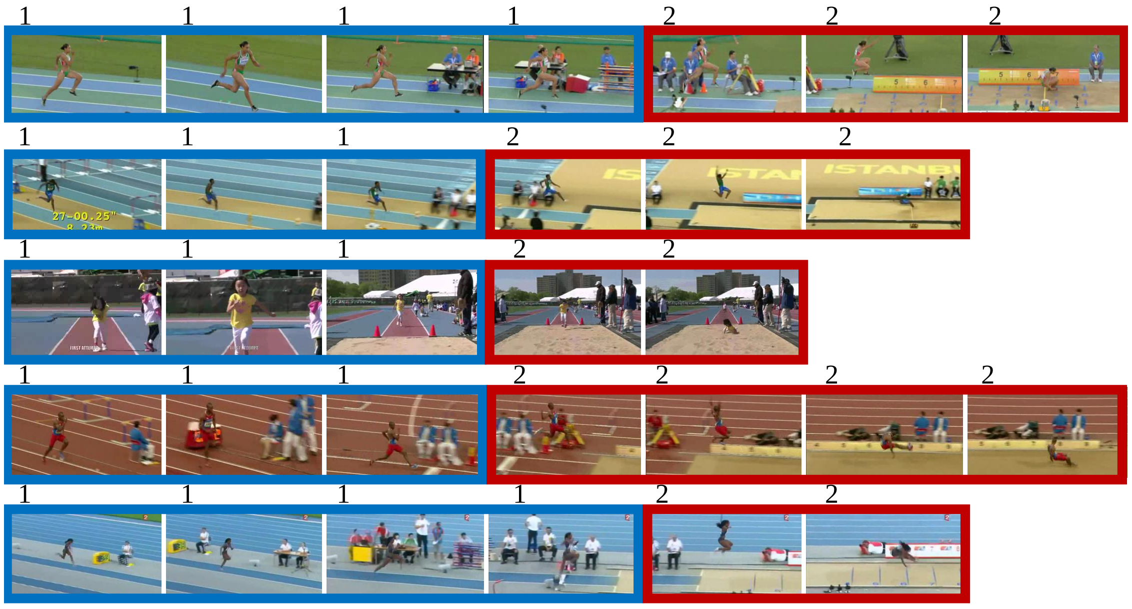

Figure 4.3: In the second step, we use hierarchical agglomerative clustering to merge similar parts. Then the first two parts in previous one are merged. However, in the first clip, one segment is incorrectly merged with the first part.

For each video, we want to find k_l tight clusters of segments such that the distance of each segment in a cluster from the cluster center is minimized. Let $c_p = \frac{1}{|p|}\sum_{i=0}^{|p|-1} x_{b_p+i}$ be the center of cluster p, where x_{b_p+i} is a feature vector of segment $b_p + i$, $b_p + i(i \in \{0, 1, 2, ..., |p| - 1\})$ represents all the segments in cluster p and b_p is the first segment which belongs to cluster p. Let $\gamma_{b_p+i,p}$ be a binary variable, which is 1 if segment $b_p + i$ is assigned to cluster p, otherwise it is 0. Since we cluster *temporal* segments, which have sequential order, we cannot arbitrarily cluster segments using a standard approach like K-means. Therefore, we define an objective function that imposes a sequential order on the cluster as follows:

$$L = \sum_{p} \sum_{i=0}^{|p|-1} \gamma_{b_p+i,p} ||x_{b_p+i} - c_p||^2, \quad \text{where } b_p + i \in p \text{ and } b_p + |p| = b_{p+1}. \tag{4.1}$$

The objective function represents the sum of the squares of the Euclidean distances of each segment to the center of its assigned cluster. Our goal is to find values for $\gamma_{b_p+i,p}$ so as to minimize L. We also need to determine cluster centers c_p. This problem is solved using an expectation maximization (EM) like algorithm. In the expectation step, we keep c_p fixed, and only update $\gamma_{b_p+i,p}$ for segments which are at the part boundaries. In the maximization step, we minimize L with respect to c_p, keeping $\gamma_{b_p+i,p}$ fixed. This two-stage optimization is then repeated until convergence. Note that exploiting the sequential ordering constraint enables the proposed method to be more computationally efficient than naive EM clustering.

4.2.2 Updating Sub-actions

After clustering, segments in a video are divided into sequential parts, which are candidate sub-actions. However, these candidates may not be sufficiently good to be directly used in our model. There are two main issues. First, sub-actions may be split into multiple parts (Fig. 4.1), requiring them to be merged. Second, the partitioning results may be inconsistent over different videos of an action (Fig. 4.2). To remove inconsistencies, sub-action labels for some of the segments should be adjusted.

Both of the above issues can be effectively solved by a set of linear SVM classifiers. An SVM classifier is trained between each two adjacent candidate sub-actions, where segments belonging to one candidate sub-action are taken as positive samples and segments belonging to the adjacent candidate sub-action are treated as negative samples. The SVMs try to distinguish adjacent candidate sub-actions. The discriminant function of an SVM classifier is $y = \mathrm{sign}(w \cdot x + b)$, where w, b are learned parameters, y is predicted sub-action label and x is the feature vector of a segment.

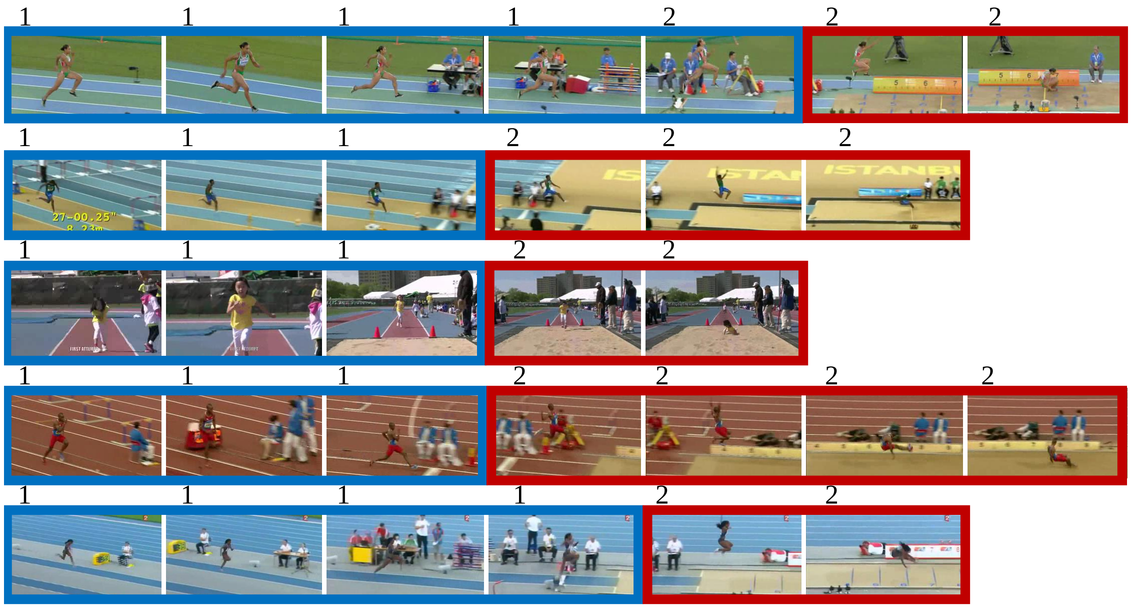

Figure 4.4: The partitioning results after adjustment. The partitions are updated iteratively.

Merging similar candidate sub-actions together. We use hierarchical agglomerative clustering to merge similar candidate sub-actions. The distance metric used in the clustering is the SVM margin obtained from the above learned SVM. The SVM margin is determined by its primal form $\frac{1}{\|w\|}$. This process is repeated iteratively by merging candidate sub-actions with the smallest distance until no distance is lower than the threshold (0.9). After merging, action l has m_l sub-actions.

Optimizing sub-actions. In order to ensure that sub-actions are consistent among all videos of an action, we use an iterative procedure to adjust sub-action partitions by alternating between the following two steps. 1) Fix sub-action labels y, train an SVM (w, b) for every pair of adjacent sub-actions. 2) Given the learned SVMs (w, b), update the sub-action labels y of the two segments at the boundary of each adjacent sub-action pairs.

The procedure converges when no sub-action labels are updated, which usually takes 3–4 iterations. In each iteration, only segments at the boundary of adjacent sub-actions pair are tested with the current SVMs and their sub-action labels may be updated. The sub-action labels for all the other segments are kept the same, making the procedure efficient.

4.3 Sub-Action Detectors

After obtaining the final sub-action partitions for all training videos, a set of sub-action classifiers $T_z(\cdot)(1 \leq z \leq m_l)$ is trained separately for each action l. In order to recognize and temporally localize actions in the untrimmed testing videos, we first detect sub-action candidates and then combine these sub-action detections to localize actions.

For a sub-action z of action l, we collect all the segments $\{i | g_i(l) = z\}$ as positive samples to train a SVM model. We perform Platt Scaling for the decision values to obtain the probability that the given segment is present in the sub-action z. The model $T_z(\cdot)$ gives us probability of the sub-action in a segment. However, localizing actions only based on this prediction value can be suboptimal. Some false negative segments may break an action instance into multiple instances, leading to inaccurate localization. To reduce the number of false alarms, we take the duration of actions into consideration as well. In most cases, the majority of instances of the same action have similar durations. For instance, the “jump” action usually finishes within 1 second, while “clean and jerk” always takes several seconds. We assume the duration d of a sub-action follows a Gaussian distribution $\Phi(d) \sim \mathcal{N}(\mu, \sigma)$.

Combining the prediction and duration scores together, sub-action z's confidence value with duration d can be expressed as in Eq. 4.2. The confidence score is computed as the product of classifier prediction and duration score. The prediction score of a duration is the average of SVM probability output from all the segments within that duration.

$$P_z(d) = \frac{1}{|d|} \sum_{i \in d} T_z(x_i) \cdot \Phi_z(d). \tag{4.2}$$

Assume, we have s segments in a video for which we already have computed sub-action scores. Our aim is to determine the starting and ending segment of a sub-action in this video. We compute an $s \times s$ upper triangular detection scores matrix. The column and row indices of the matrix represent the candidate starting and ending segment, respectively. For each starting and ending segment pair that represents a candidate duration, the sub-action score is computed using Eq. 4.2. Then, using dynamic programming we optimally determine the beginning and ending segments for the sub-action.

4.4 Detecting Actions

Our approach represents each action by a sequence of sub-actions. We enforce that the sub-actions of an action must occur in a sequential order, but two adjacent sub-actions may have some temporal overlap or gap between them. We assume the time between adjacent sub-actions follows a Gaussian probability density function $\Psi(\cdot) \sim \mathcal{N}(\mu, \sigma)$. This function penalizes sub-action combinations with a wrong order or those that have a greater gap than desired. The parameters μ and σ of $\Psi(\cdot)$ are learned from the validation set.

Given an unclipped video of unknown action, we first segment it into equal length segments. We then apply all sub-action detectors and corresponding to each sub-action detector we get a sequence of detection scores. Now the aim becomes selecting optimal combinations of sub-action detections to detect an action, by considering both sub-action scores and distance between sub-actions. This inference is formulated as a network flow problem. We build a flow graph as shown in Fig. 4.5. The start and end time of each sub-action detection d are represented as two vertical bars, which are connected by an observation edge with cost $-P(d)$ (Eq. 4.2). Let d and d' be two adjacent sub-action detections. They are connected by a transition edge with cost $-\Psi(d, d')$.

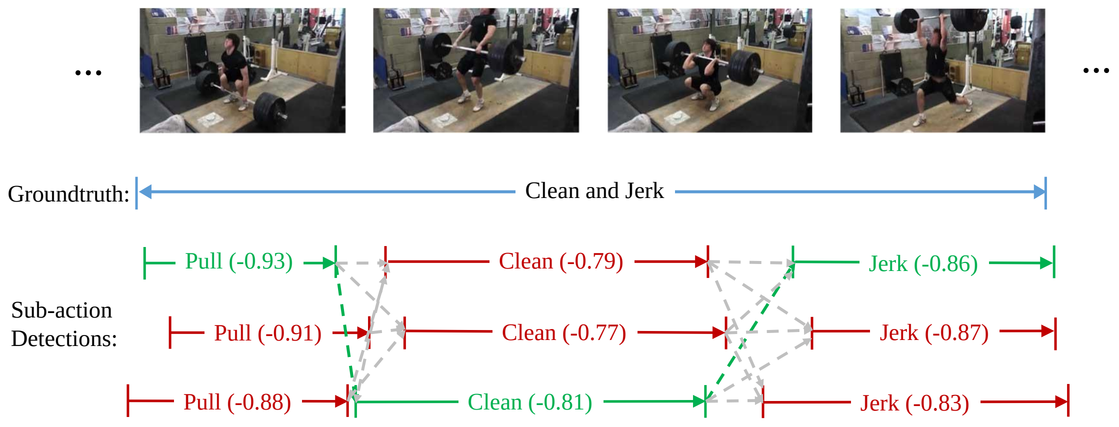

Figure 4.5: An example of network flow with three sub-actions. The groundtruth is shown by a blue line. The observation edges are denoted as solid lines, and the transition edges are expressed as dashed lines. The green lines represent the selected optimal path. Since the optimal path considers both sub-action scores and distance scores between sub-actions, the locally optimal detection of sub-action 'jerk' with the lowest cost is correctly ignored.

The objective function is

$$\min_f(-\sum_d P(d)f_d - \sum_{d,d'} \Psi(d,d')f_{d,d'}), \quad s.t. \quad f_d, f_{d,d'} \in \{0,1\} \text{ and } \sum_{d'} f_{d,d'} = f_d = \sum_{d'} f_{d',d}, \quad (4.3)$$

where $f_{dd'}$ represents the flow from detection d to d'. The first constraint enforces that flow is either 1 or 0. A sub-action detection d is selected if $f_d = 1$ and its adjacent sub-action detection d' is also selected if $f_{d,d'} = 1$. Each sub-action detection can be selected to obtain an action detection only once. The second constraint ensures that the incoming flow to a node is equal to its outgoing flow. The shortest path problem is solved efficiently using dynamic programming, similar as in [74] which finds high-quality approximate solutions to min-cost flow problem for multiple objects tracking. A selected shortest path represents a valid action detection, whose sub-action scores and distance scores between sub-actions are optimal. We keep solving the shortest path problem until the cost of the obtained path is higher than a threshold, which is learned from the validation set. Every time a shortest path is selected, its corresponding sub-action detections are removed from the graph.

4.5 Experimental Results

We evaluate our method on THUMOS'14 and Mexaction2 datasets. We use the Improved Dense Trajectory Features (iDTF) [99] as low-level features, which consist of four descriptors: histogram-of-gradients (HOG), histogram-of-flow (HOF), motion-based histograms (MBH) and trajectories. In the experiments, we use the first 3 descriptors with late fusion. We generate a 256-bin GMM and build a Fisher vector representation for both datasets.

THUMOS'14. The temporal action detection task of THUMOS'14 [36] consists of 20 classes of sports actions. We use both train and validation sets for training sub-action models, and fix the length of segments to 0.3s. The thresholds and parameters for Gaussian distributions are learned by a ten-fold cross-validation process. We follow the detection measurement protocol specified in the THUMOS'14 temporal action localization challenge.

We present direct comparisons against top performers on the THUMOS'14 challenge leader board [67, 100], which use the same low-level features. Moreover, we also compare our results with two recent deep learning based approaches [109, 80].

For the set of THUMOS'14 classes, the number of sub-actions discovered by our approach is as follows: two actions (Clean & Jerk and High Jump) consist of 3 sub-actions, 15 actions are divided into 2 sub-actions, and 3 actions only contain one sub-action (we call these singleton actions). The mAP (Mean Average Precision) is reported with different intersection over union (IOU) thresholds, α. The results are shown in Table 4.1. As is clear from these results, our approach significantly outperforms the other approaches.

Table 4.1: Temporal action localization results on THUMOS'14. mAP is reported for different intersection-over-union thresholds α.

	$\alpha = 0.5$	$\alpha = 0.3$	$\alpha = 0.1$
Wang et al. [101]	8.3	14.0	18.2
Oneata et al. [67]	14.4	27.0	36.6
Yeung et al. [109]	17.1	36.0	48.9
Shou et al. [80]	18.8	-	-
Yuan et al. [112]	18.8	30.8	40.9
Ours	**22.0**	**43.7**	**51.3**
Ours (w/o opt.)	20.4	36.1	49.5
Ours (Singleton)	19.3	33.3	48.2

We also conduct two ablation experiments. In the first experiment, we skip the optimization of part partitioning step (Section 4.2.2) and assign sub-actions labels for each segment based on the merging results (Section 4.2.1). The mAPs of this model, denoted as Ours (w/o opt), are consistently worse than those from our complete system, verifying the effectiveness of our part partition optimization step.

In the second experiment, we assume that all actions are composed of only a sub-action (singleton actions). We see that the ablative singleton model, denoted as Ours (Singleton), performs worse than our proposed approach. This validates our hypothesis that sub-action discovery improves action localization.

Mexaction2. This is a two actions dataset: "BullChargeCape" and "HorseRiding". This data set has three parts. The first part is INA videos, which were collected from TV shows. The videos in this part are untrimmed and split into train, validation and test sets. The second part is from Youtube videos, and the third part is from UCF101 Horse Riding clips. Moreover, the clips in the last two parts are trimmed and are only used for training. We use the train set for training sub-action detectors. The length of segments is fixed as 0.1s. The validation set is used for learning the parameters for Gaussian models and thresholds for action detectors.

We present the comparison against baseline, which uses Bag of Visual Words DTF features, and Shou *et al.*[80]'s reported results. We use the same metric as in [80], which report the mAP with IOU threshold $\alpha = 0.4$. Since both actions are hard to be divided into sub-actions, we get one sub-action for them (they are singleton actions). Our sub-action based approach achieves huge improvement for "BullCharge-Cape" and slightly better result for "HorseRiding" (Table 4.2). In this experiment, we did not use the distance term.

Table 4.2: Average precision on MEXaction2 dataset

	Baseline [63]	Shou *et al.*[80]	Ours
BullCharge	0.3	11.6	**26.2**
HorseRiding	3.1	3.1	**3.8**
mAP	1.7	7.4	**15.0**

4.5.1 Analysis of Our Results

Table 4.1 shows results from the ablation experiments, i.e. Ours (w/o opt) and Ours (Singleton), in order to verify the contributions of different components of our method. We observe:

1. When α changes from 0.5 to 0.3 in Table 4.1, our approach outperforms other methods at

all the overlap thresholds. The mAP increases greatly at high IOU and reaches 43.7% at $\alpha = 0.3$, demonstrating that we always have greater IOU with ground truth.

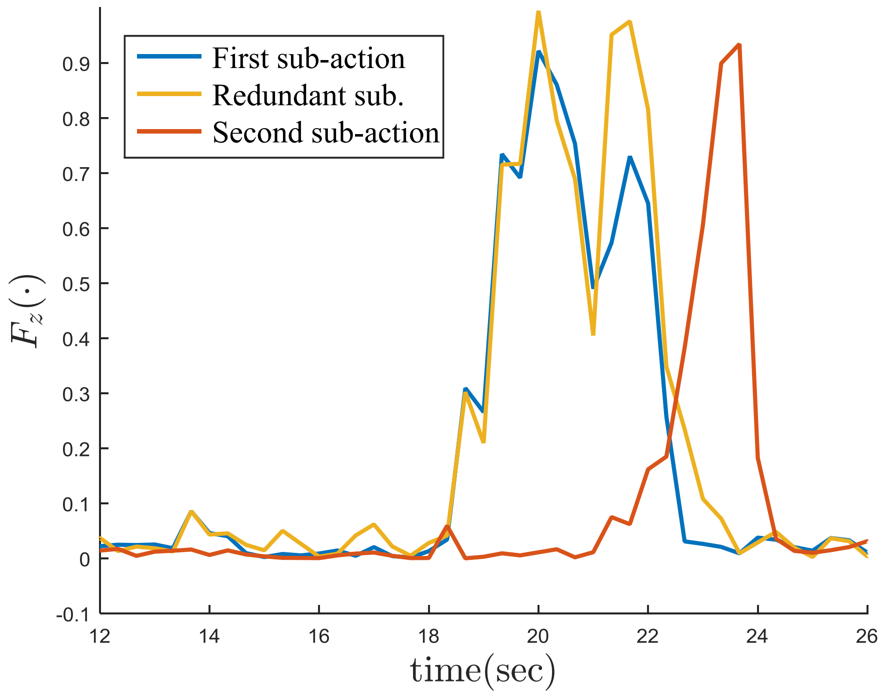

Figure 4.6: Results after clustering segments into parts (Section 4.2).

2. Our sub-action based model consistently outperforms the singleton model for temporal action detection (Tables 4.1, 4.2). However, when IOU threshold decreases to 0.1, singleton model shows less difference (48.2% vs. 51.3%), while when $\alpha = 0.3$ the difference is 10% (Table 4.1), showing the benefits of modeling sub-actions.
3. Without the merging of similar parts, the first sub-action is divided into two redundant parts. As shown in Fig. 4.6, the two-part model generates similar scores for the segments. Training separate models for these redundant sub-actions would be wasteful.
4. Comparing plots in Fig. 4.7 with plots in Fig. 4.8, we appreciate the importance of part optimization (Section 4.2.2), which adjusts boundaries to accentuate the difference between discovered sub-actions.

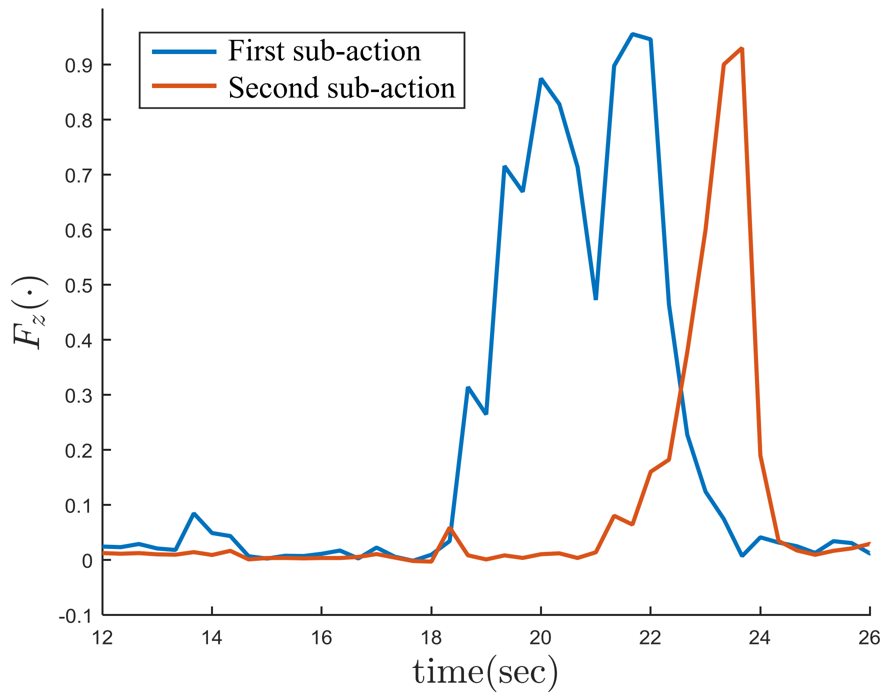

Figure 4.7: Results obtained after merging similar parts (Section 4.2.1).

5. Our approach outperforms other methods by a large margin when an action can be clearly divided into sub-actions (e.g., cliff diving, javelin throw and long jump). When the sub-actions are hard to determine, our algorithm is still competitive (e.g., billiards).
6. Finally, our approach performs less well on short duration actions (golf swing) or actions that resist decomposition into sub-actions (frisbee catch). When the duration of an action varies widely (horse riding), the duration term may hurt performance.

Runtime Analysis. The proposed approach is very efficient, particularly for long videos. Compared to sliding window based methods, for each segment we only compute the low-level features once. Compared to CNN-based methods, we train much faster. For the MEXaction2 experiments, during sub-action discovery, clustering segments takes 121s and updating sub-actions takes 87s. The total time for sub-action training is 1686s. The improved DTF feature extraction and Fisher

Vector computation in our implementation is optimized by avoiding unnecessary string-float conversions. Using dynamic programming and our optimized feature extraction and Fisher Vector computation, our temporal action detection system can process videos at 40 fps.

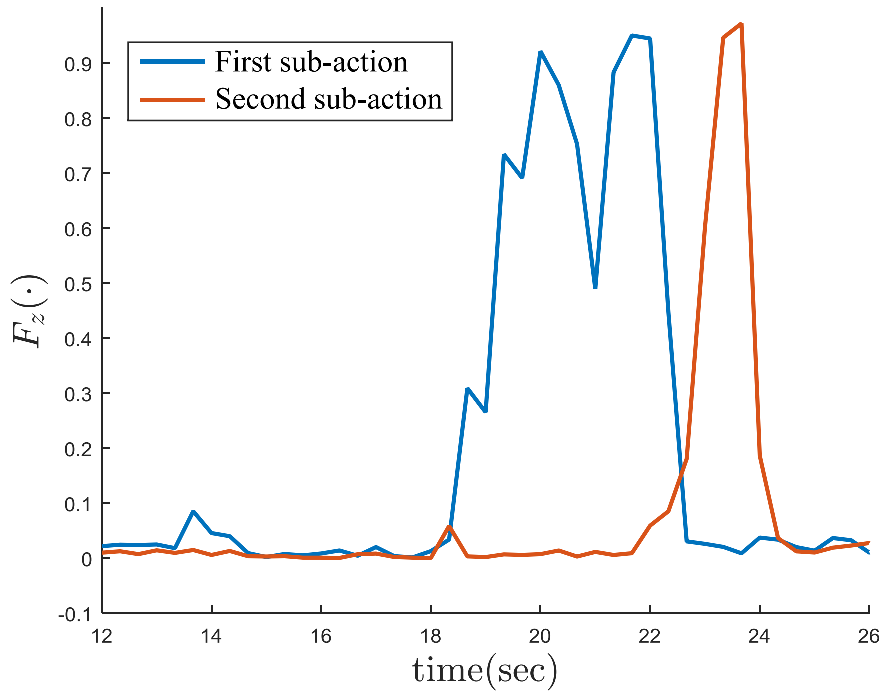

Figure 4.8: Results of our full approach using allsteps.

4.6 Summary

In this chapter, we present an approach to temporally localize actions in long untrimmed videos. Our approach decomposes each action into multiple sub-actions. The generated sub-actions are discovered automatically and consistent across different instances in the same class that vary significantly in terms of viewpoint and visual appearance. Our proposed sub-action based sequential model is evaluated on large-scale temporal action localization datasets and achieves state-of-the-art results on THUMOS'14 and MEXaction2 datasets.

The proposed approach in this chapter is good at dealing with temporal action localization problem. However, in applications such as invasion detection or sports analysis, spatial localization is desired as well. In the next chapter, we present a convolutional neural network based end-to-end video action detection approach, which is able to localize action spatial-temporally.

CHAPTER 5: TUBE CONVOLUTIONAL NEURAL NETWORK (T-CNN) FOR ACTION DETECTION IN VIDEOS

The temporal action localization has the capability to answer "when" the action happens, but is not able to answer "where" it happens. In this chapter, we address the problem of action detection, by detecting every occurrence of a given action within a long video, and localizing each detection both in space and time. In image domain, Region Convolution Neural Network (RCNN) for object detection was proposed by Girshick *et al.*[22]. It was followed by Fast R-CNN proposed in [21], which includes the classifier as well. Later, Faster R-CNN [75] was developed by introducing a region proposal network. It has been extensively used to produce excellent results for object detection in images. To consider both spatial and temporal information at the same time, we propose Tube Convolutional Neural Network (T-CNN) for action detection by leveraging the descriptive power of 3D CNN.

5.1 Generalizing R-CNN from 2D to 3D

Generalizing R-CNN from 2D image regions to 3D video tubes is challenging due to the asymmetry between space and time. Different from images which can be cropped and reshaped into a fixed size, videos vary widely in temporal dimension. Therefore, we divide input videos into fixed length (8 frames) clips, so that video clips can be processed with a fixed-size ConvNet architecture. Also, clip based processing mitigates the cost of GPU memory.

To better capture the spatio-temporal information in video, we exploit 3D CNN for action proposal generation and action recognition. One advantage of 3D CNN over 2D CNN is that it captures motion information by applying convolution in both time and space. Since 3D convolution and 3D

Table 5.1: T-CNN architecture. We refer kernel with shape $d \times h \times w$ where d is the kernel depth, h and w are height and width, and output matrix with shape $C \times D \times H \times W$ where C is the number of channels, D is the number of frames, H and W are the height and width. toi-pool2 exists in TPN. Divide this table into different parts of your approach Proposal network, TOI pooling etc.

name	kernel dims	output dims
	$(d \times h \times w)$	$(C \times D \times H \times W)$
conv1	$3 \times 3 \times 3$	$64 \times 8 \times 300 \times 400$
max-pool1	$1 \times 2 \times 2$	$64 \times 8 \times 150 \times 200$
conv2	$3 \times 3 \times 3$	$128 \times 8 \times 150 \times 200$
max-pool2	$2 \times 2 \times 2$	$128 \times 4 \times 75 \times 100$
conv3a	$3 \times 3 \times 3$	$256 \times 4 \times 75 \times 100$
conv3b	$3 \times 3 \times 3$	$256 \times 4 \times 75 \times 100$
max-pool3	$2 \times 2 \times 2$	$256 \times 2 \times 38 \times 50$
conv4a	$3 \times 3 \times 3$	$512 \times 2 \times 38 \times 50$
conv4b	$3 \times 3 \times 3$	$512 \times 2 \times 38 \times 50$
max-pool4	$2 \times 2 \times 2$	$512 \times 1 \times 19 \times 25$
conv5a	$3 \times 3 \times 3$	$512 \times 1 \times 19 \times 25$
conv5b	$3 \times 3 \times 3$	$512 \times 1 \times 19 \times 25$
toi-pool2*	–	$128 \times 8 \times 8 \times 8$
toi-pool5	–	$512 \times 1 \times 4 \times 4$
1x1 conv	–	8192
fc6	–	4096
fc7	–	4096

max pooling are utilized in our approach, not only in the spatial dimension but also in the temporal dimension, the size of video clip is reduced while distinguishable information is concentrated. As demonstrated in [95], the temporal pooling is important for recognition task since it better models the spatio-temporal information of video and reduces some background noise. However, the temporal order information is lost. That means if we arbitrarily change the order of the frames in a video clip, the resulting 3D max pooled feature cube will be the same. This is problematic in action *detection*, since it relies on the feature cube to get bounding boxes for the original frames. To this end, we incorporate temporal skip pooling to retain temporal order information residing in the original frames. More details are provided in the next section.

Since a video is processed clip by clip, action tube proposals with various spatial and temporal sizes are generated for different clips. These clip proposals need to be linked into a tube proposal sequence, which is used for action label prediction and localization. To produce a fixed length feature vector, we propose a new pooling layer – **Tube-of-Interest** (ToI) pooling layer. The ToI pooling layer is a 3D generalization of Region-of-Interest (RoI) pooling layer of R-CNN. The classic max pooling layer defines the kernel size, stride and padding which determines the shape of the output. In contrast, for RoI pooling layer, the output shape is fixed first, then the kernel size and stride are determined accordingly. Compared to RoI pooling, which takes 2D feature map and 2D regions as input, ToI pooling deals with feature cube and 3D tubes. Denote the size of a feature cube as $d \times h \times w$, where d, h and w respectively represent the depth, height and width of the feature cube. A ToI in the feature cube is defined by a d-by-4 matrix, which is composed of d boxes distributed in all the frames. The boxes are defined by a four-tuple $(x_1^i, y_1^i, x_2^i, y_2^i)$ that specifies the top-left and bottom-right corners in the i-th feature map. Since the d bounding boxes may have different sizes, aspect ratios and positions, in order to apply spatio-temporal pooling, pooling in spatial and temporal domains are performed separately. First, the $h \times w$ feature maps are divided into $H \times W$ bins, where each bin corresponds to a cell with size of approximately $h/H \times w/W$. In each cell, max pooling is applied to select the maximum value. Second, the spatially pooled d feature maps are temporally divided into D bins. Similar to the first step, d/D adjacent feature maps are grouped together to perform the standard temporal max pooling. As a result the fixed output size of ToI pooling layer is $D \times H \times W$. A graphical illustration of ToI pooling is presented in Figure 5.1.

Back-propagation of ToI pooling layer routes the derivatives from output back to the input. Assume x_i is the i-th activation to the ToI pooling layer, and y_j is the j-th output. Then the partial derivative of the loss function (L) with respect to each input variable x_i can be expressed as:

$$\frac{\partial L}{\partial x_i} = \frac{\partial y_j}{\partial x_i}\frac{\partial L}{\partial y_j} \rightarrow \frac{\partial L}{\partial x_i} = \sum_j [i = f(j)] \frac{\partial L}{\partial y_j}. \tag{5.1}$$

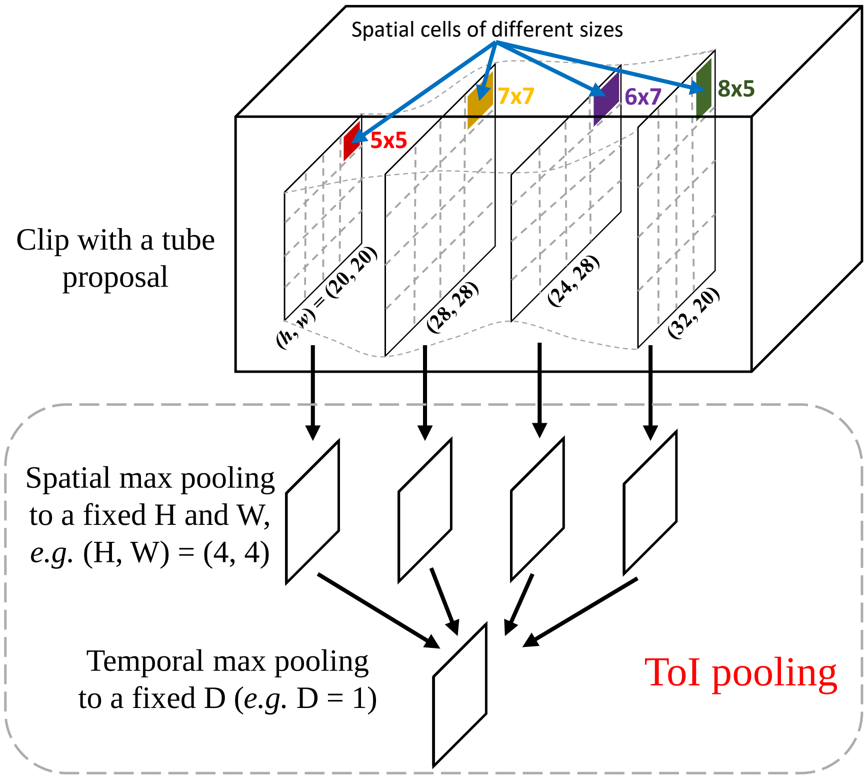
Spatial cells of different sizes
5x5
7x7
6x7
8x5
Clip with a tube proposal
(h, w) = (20, 20)
(28, 28)
(24, 28)
(32, 20)
Spatial max pooling to a fixed H and W, e.g. (H, W) = (4, 4)
Temporal max pooling to a fixed D (e.g. D = 1)
ToI pooling

Figure 5.1: Tube of interest pooling.

Each pooling output y_j has a corresponding input position i. We use a function $f(\cdot)$ to represent the `argmax` selection from the ToI. Thus, the gradient from the next layer $\partial L/\partial y_j$ is passed back to only that neuron, which achieved the max $\partial L/\partial x_i$. Since one input may correspond to multiple outputs, the partial derivatives are the accumulation of multiple sources.

5.2 T-CNN Pipeline

As shown in Figure 1.5, our T-CNN is an end-to-end deep learning framework that takes video clips as input. The core component is the Tube Proposal Network (TPN) (see Figure 5.2) to produce tube proposals for each clip. Linked tube proposal sequence represents spatio-temporal action detection in the video and is also used for action recognition.

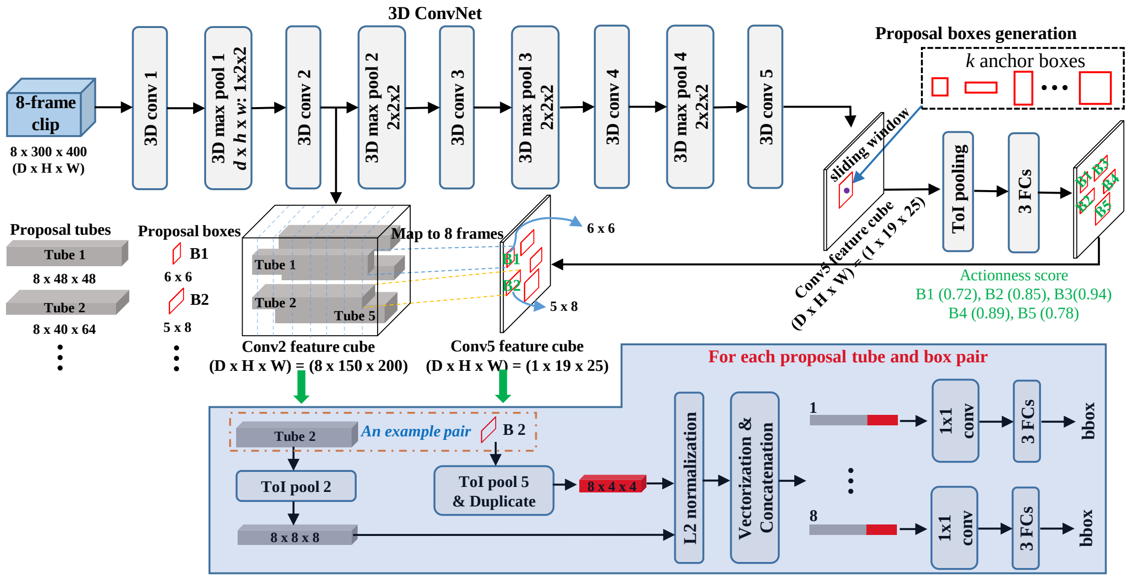

Figure 5.2: Tube proposal network.

5.2.1 Tube Proposal Network

For a 8-frame video clip, 3D convolution and 3D pooling are used to extract spatio-temporal feature cube. In 3D ConvNet, convolution and pooling are performed spatio-temporally. Therefore, the temporal information of the input video is preserved. Our 3D ConvNet consists of seven 3D convolution layers and four 3D max-pooling layers. We denote the kernel shape of 3D convolution/pooling by $d \times h \times w$, where d, h, w are depth, height and width, respectively. In all convolution layers, the kernel sizes are $3 \times 3 \times 3$, padding and stride remain as 1. The numbers of filters are 64, 128 and 256 respectively in the first 3 convolution layers and 512 in the remaining convolution layers. The kernel size is set to $1 \times 2 \times 2$ for the first 3D max-pooling layer, and $2 \times 2 \times 2$ for the remaining 3D max-pooling layers. The details of network architecture are presented in Table 5.1. We use the C3D model [95] as the pre-trained model and fine tune it on each dataset in our experiments.

After conv5, the temporal size is reduced to 1 frame (*i.e.*feature cube with depth $D = 1$). *In the feature tube, each frame/slice consists of a number of channels specified in Table 5.1. Here, we drop the number of channels for ease of explanation.* Following faster R-CNN, we generate bounding box proposals based on the conv5 feature cube.

Anchor bounding boxes selection. In faster R-CNN, the bounding box dimensions are hand picked, *i.e.*9 anchor boxes with 3 scales and 3 aspect ratios. We can directly adopt the same anchor boxes in our T-CNN framework. However, it has been shown in [37] that if we choose better priors as initializations for the network, it will help the network learn better for predicting good detections. Therefore, instead of choosing hand-picked anchor boxes, we apply k-means clustering on the training set bounding boxes to learn 12 anchor boxes (*i.e.*clustering centroids). This data driven anchor box selection approach is adaptive to different datasets.

Each bounding box is associated with an “actionness” score, which measures the probability that the content in the box corresponds to a valid action. We assign a binary class label (of being an action or not) to each bounding box. Bounding boxes with actionness scores smaller than a threshold are discarded. In the training phase, the bounding box which has an IoU overlap higher than 0.7 with any ground-truth box or has the highest Intersection-over-Union (IoU) overlap with a ground-truth box (the later condition is considered in case the former condition may find no positive sample) is considered as a positive bounding box proposal.

Temporal skip pooling. Bounding box proposals generated from conv5 feature tube can be used for frame-level action detection by bounding box regression. However, due to temporal concentration (8 frames to 1 frame) of temporal max pooling, the temporal order of the original 8 frames is lost. Therefore, we use temporal skip pooling to inject the temporal order for frame-level detection. Specifically, we map each positive bounding box generated from conv5 feature tube to conv2 feature tube which has 8 feature frames/slices. Since these 8 feature slices correspond to the original 8 frames in the video clip, the temporal order information is preserved. As a result, if there are 5 bounding boxes in conv5 feature tube for example, 5 scaled bounding boxes are mapped in each conv2 feature slice at the corresponding locations. This creates 5 tube proposals as illustrated in Figure 5.2, which are paired with the corresponding 5 bounding box proposals for frame-level action detection. To form a fixed feature shape, ToI pooling is applied to the variable size tube proposals as well as the bounding box proposals. Since a tube proposal covers 8 frames, the ToI pooled bounding box is duplicated 8 times to form a tube. We then L2 normalize the paired two tubes and perform vectorization. For each frame, features are concatenated. Since we use the C3D model [95] as the pre-trained model, we connect a 1x1 convolution to match the input dimension of fc6. Three fully-connected layers process each descriptor and produce the output: displacement of height, width and center coordinate of each bounding box ("bbox") in each frame. Finally, a set of refined tube proposals are generated as an output from the TPN representing spatio-temporal action localization of the input video clip.

5.2.2 Linking Tube Proposals

We obtain a set of tube proposals for each video clip after the TPN. We then link these tube proposals to form a proposal sequence for spatio-temporal action localization of the entire video. Each tube proposal from different clips can be linked in a tube proposal sequence (*i.e.*video tube proposal) for action detection. However, not all combinations of tube proposals can correctly capture the complete action. For example, a tube proposal in one clip may contain the action and a tube proposal in the following clip may only capture the background. Intuitively, the content within the selected tube proposals should capture an action and connected tube proposals in any two consecutive clips should have a large temporal overlap. Therefore, two criteria are considered when linking tube proposals: actionness and overlap scores. Each video proposal is then assigned a score defined as follows:

$$S = \frac{1}{m}\sum_{i=1}^{m} Actionness_i + \frac{1}{m-1}\sum_{j=1}^{m-1} Overlap_{j,j+1} \tag{5.2}$$

where $Actionness_i$ denotes the actionness score of the tube proposal from the i-th clip, $Overlap_{j,j+1}$ measures the overlap between the linked two proposals respectively from the j-th and $(j + 1)$-th clips, and m is the total number of video clips. As shown in Figure 5.2, each bounding box proposal from conv5 feature tube is associated with an actionness score. The actionness scores are inherited by the corresponding tube proposals. The overlap between two tube proposals is calculated based on the IoU (Intersection Over Union) of the last frame of the j-th tube proposal and the first frame of the $(j+1)$-th tube proposal. The first term of S computes the average actionness score of all tube proposals in a video proposal and the second term computes the average overlap between the tube proposals in every two consecutive video clips. Therefore, we ensure the linked tube proposals can encapsulate the action and at the same time have temporal consistency. An example of linking tube proposals and computing scores is illustrated in Figure 5.3. We choose a number of linked proposal sequences with the highest scores in a video (see more details in Sec. 5.3.1).

5.2.3 Action Detection

After linking tube proposals, we get a set of linked tube proposal sequences, which represent potential action instances. The next step is to classify these linked tube proposal sequences. The tube proposals in the linked sequences may have different sizes. In order to extract a fixed length feature vector from each of the linked proposal sequences, our proposed ToI pooling is utilized. Then the ToI pooling layer is followed by two fully-connected layers and a drop-out layer. The dimension of the last fully-connected layer is $N + 1$ (N action classes and 1 background class).

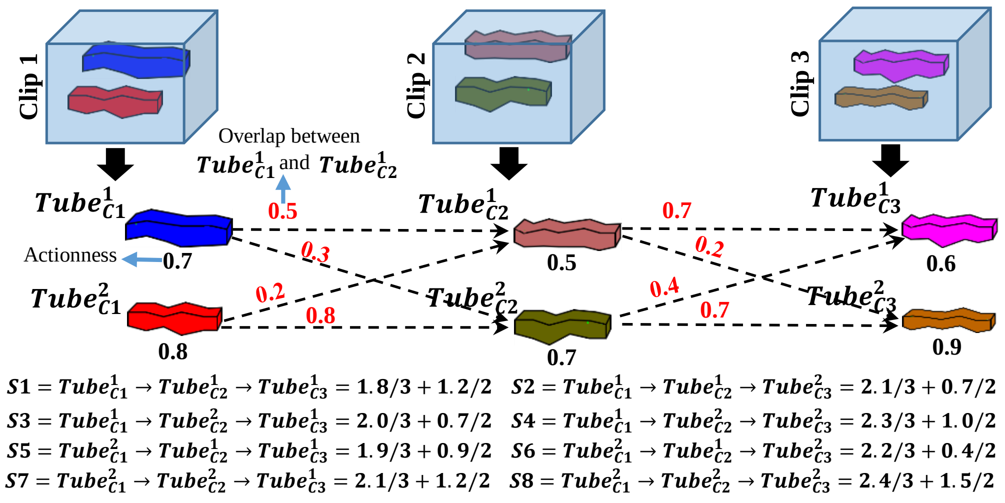
Clip 1
Clip 2
Clip 3
Overlap between
Tube$^1_{C1}$ and Tube$^1_{C2}$
Tube$^1_{C1}$
Tube$^2_{C1}$
Tube$^1_{C2}$
Tube$^2_{C2}$
Tube$^1_{C3}$
Tube$^2_{C3}$
Actionness
0.7
0.8
0.5
0.7
0.6
0.9
0.5
0.3
0.2
0.8
0.7
0.2
0.4
0.7
$S1 = Tube^1_{C1} \rightarrow Tube^1_{C2} \rightarrow Tube^1_{C3} = 1.8/3 + 1.2/2$
$S2 = Tube^1_{C1} \rightarrow Tube^1_{C2} \rightarrow Tube^2_{C3} = 2.1/3 + 0.7/2$
$S3 = Tube^1_{C1} \rightarrow Tube^2_{C2} \rightarrow Tube^1_{C3} = 2.0/3 + 0.7/2$
$S4 = Tube^1_{C1} \rightarrow Tube^2_{C2} \rightarrow Tube^2_{C3} = 2.3/3 + 1.0/2$
$S5 = Tube^2_{C1} \rightarrow Tube^1_{C2} \rightarrow Tube^1_{C3} = 1.9/3 + 0.9/2$
$S6 = Tube^2_{C1} \rightarrow Tube^1_{C2} \rightarrow Tube^2_{C3} = 2.2/3 + 0.4/2$
$S7 = Tube^2_{C1} \rightarrow Tube^2_{C2} \rightarrow Tube^1_{C3} = 2.1/3 + 1.2/2$
$S8 = Tube^2_{C1} \rightarrow Tube^2_{C2} \rightarrow Tube^2_{C3} = 2.4/3 + 1.5/2$

Figure 5.3: An example of linking tube proposals in each video clips using network flow. In this example, there are three video clips and each has two tube proposals, resulting in 8 video proposals. Each video proposal has a score, *e.g.*$S1, S2, ..., S8$, which is computed according to Eq. (2).

5.3 Experiments

To verify the effectiveness of the proposed T-CNN for action detection, we evaluate T-CNN on three trimmed video datasets including UCF-Sports [76], J-HMDB [32], UCF-101 [85] and one un-trimmed video dataset – THUMOS'14 [36].

5.3.1 Implementation Details

We implement our method based on the Caffe toolbox [35]. The TPN and recognition network share weights in their common layers. Due to memory limitation, in training phase, each video is divided into overlapping 8-frame clips with resolution 300×400 and temporal stride 1. When training the TPN network, each anchor box is assigned a binary label. Either the anchor box which has the highest IoU overlap with a ground-truth box, or an anchor box that has an IoU overlap higher than 0.7 with any ground-truth box is assigned a positive label, the rest are assigned negative label. In each iteration, 4 clips are fed into the network. Since the number of background boxes is much more than that of action boxes, to well model the action, we randomly select some of the negative boxes to balance the number of positive and negative samples in a batch. For recognition network training, we choose 40 linked proposal sequences with the highest scores in a video as Tubes of Interest.

Our model is trained in an alternative manner. First, **Initialize TPN** based on the pre-trained model in [95], then using the generated proposals to **initialize recognition networks**. Next, the weights tuned by recognition network are used to **update TPN**. Finally, the tuned weights and proposals from TPN are used for **finalizing recognition network**. For all the networks for UCF-Sports and J-HMDB, the learning rate is initialized as 10^{-3} and decreased to 10^{-4} after 30k batches. Training terminates after 50k batches. For UCF-101 and THUMOS'14, the learning rate is initialized as 10^{-3} and decreased to 10^{-4} after 60k batches. Training terminates after 100k batches.

During testing, each video is divided into non-overlapping 8-frame clips. If the number of frames in video cannot be divided by 8, we pad zeros after the last frame to make it dividable. 40 tube proposals with the highest actionness confidence through TPN are chosen for the linking process. Non-maximum suppression (NMS) is applied to linked proposals to get the final action detection results.

5.3.2 Datasets and Experimental Results

UCF-Sports. This dataset contains 150 short videos of 10 different sport classes. Videos are trimmed and bounding boxes annotations are provided for all frames. We follow the training and test split defined in [51]. We use the IoU criterion and generate ROC curve in Figure 5.6(a) when overlap criterion equals to $\alpha = 0.2$. Figure 5.6(b) illustrates AUC (Area-Under-Curve) measured with different overlap criterion. In direct comparison, our T-CNN clearly outperforms all the competing methods shown in the plot. We are unable to directly compare the detection accuracy against Peng *et al*.[72] in the plot, since they do not provide the ROC and AUC curves. As shown in Table 5.2, the frame level mAP of our approach outperforms theirs in 8 actions out of 10. Moreover, by using the same metric, the video mAP of our approach reaches 95.2 ($\alpha = 0.2$ and 0.5), while they report 94.8 ($\alpha = 0.2$) and 94.7 ($\alpha = 0.5$).

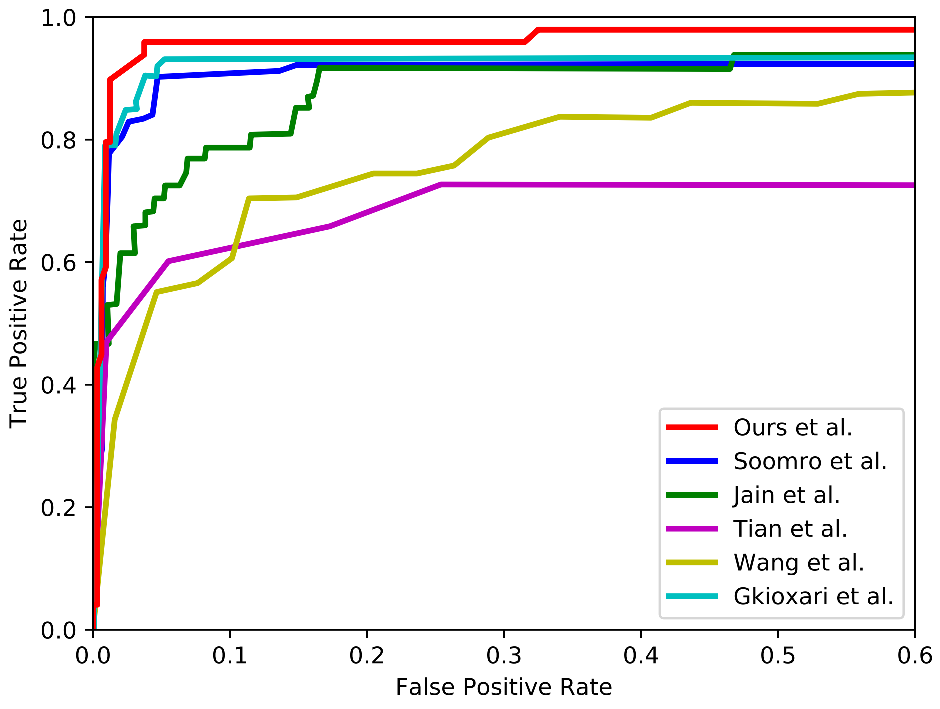

Figure 5.4: The ROC curves for UCF-Sports Dataset [76]. The results are shown for Jain *et al*.[29] (green), Tian *et al*.[90] (purple), Soomro *et al*.[84] (blue), Wang *et al*.[102] (yellow), Gkioxari *et al*.[23] (cyan) and Proposed Method (red).

J-HMDB. This dataset consists of 928 videos with 21 different actions. All the video clips are well trimmed. There are three train-test splits and the evaluation is done on the average results over the three splits. The experiment results comparison is shown in Table 5.3. We report our results with 3 metrics: frame-mAP, the average precision of detection at frame level as in [23].

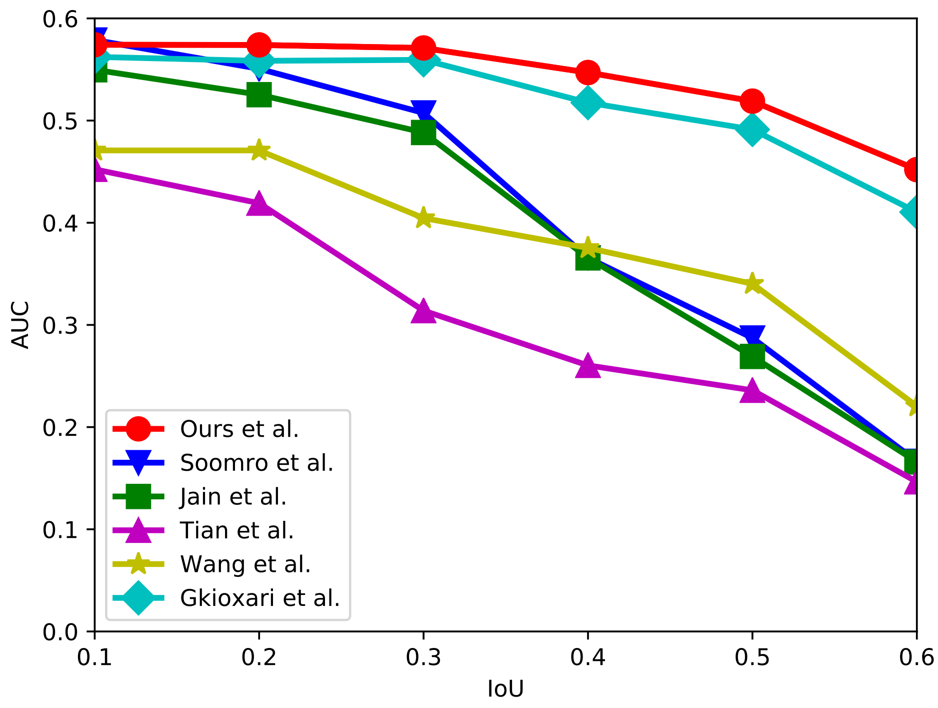

Figure 5.5: The AUC curves for UCF-Sports Dataset [76]. The results are shown for Jain *et al.*[29] (green), Tian *et al.*[90] (purple), Soomro *et al.*[84] (blue), Wang *et al.*[102] (yellow), Gkioxari *et al.*[23] (cyan) and Proposed Method (red).

Table 5.2: mAP for each class of UCF-Sports. The IoU threshold α for frame m-AP is fixed to 0.5.

	Diving	Golf	Kicking	Lifting	Riding	Run	SkateB.	Swing	mAP
Gkioxari *et al.*[23]	75.8	69.3	54.6	99.1	89.6	54.9	29.8	88.7	68.1
Weinzaepfel *et al.*[105]	60.71	77.55	65.26	**100.00**	99.53	52.60	47.14	**88.88**	71.9
Peng *et al.*[72]	**96.12**	80.47	73.78	99.17	97.56	82.37	57.43	83.64	84.51
Ours	84.38	**90.79**	**86.48**	99.77	**100.00**	**83.65**	**68.72**	65.75	**86.7**

The average precision at video level as in [23] with IoU threshold $\alpha = 0.2$ and $\alpha = 0.5$. It is evident that our T-CNN consistently outperforms the state-of-the-art approaches in terms of all three evaluation metrics.

Table 5.3: Comparison to the state-of-the-art on J-HMDB. The IoU threshold α for frame m-AP is fixed to 0.5.

	f.-mAP ($\alpha = 0.5$)	v.-mAP ($\alpha = 0.2$)	v.-mAP ($\alpha = 0.5$)
Gkioxari *et al.*[23]	36.2	–	53.3
Weinzaepfel *et al.*[105]	45.8	63.1	60.7
Peng *et al.*[72]	58.5	74.3	73.1
Ours w/o skip pooling	47.9	66.9	58.6
Ours	**61.3**	**78.4**	**76.9**

UCF101. This dataset has 101 actions. For action detection task, a subset of 24 action classes and $3,207$ videos have spatio-temporal annotations. Similar to other methods, we perform the experiments on the first train/test split only. We report our results in Table 5.4 with 3 metrics: frame-mAP, video-mAP ($\alpha = 0.2$) and video-mAP ($\alpha = 0.5$). Our approach again yields the best performance. Moreover, we also report the action recognition results of T-CNN on the above three datasets in Table 5.5.

Table 5.4: Comparison to the state-of-the-art on UCF-101 (24 actions). The IoU threshold α for frame m-AP is fixed to 0.5.

	f.-mAP	video-mAP			
IoU th.		0.05	0.1	0.2	0.3
Weinzaepfel *et al.*[105]	35.84	54.3	**51.7**	46.8	37.8
Peng *et al.*[72]	39.63	54.5	50.4	42.3	32.7
Ours	**41.37**	**54.7**	51.3	**47.1**	**39.2**

THUMOS'14. To further validate the effectiveness of our proposed T-CNN approach for action detection, we evaluate it using the untrimmed videos from the THUMOS'14 dataset [36]. The THUMOS'14 spatio-temporal localization task consists of 4 classes of sports actions: BaseballPitch, golfSwing, TennisSwing and ThrowDiscus. There are about 20 videos per class and each video contains 500 to $3,000$ frames. The videos are divided into validation set and test set, but only video in the test set have spatial annotations provided by [86]. Therefore, we use samples corresponding to those 4 actions in UCF-101 with spatial annotations to train our model.

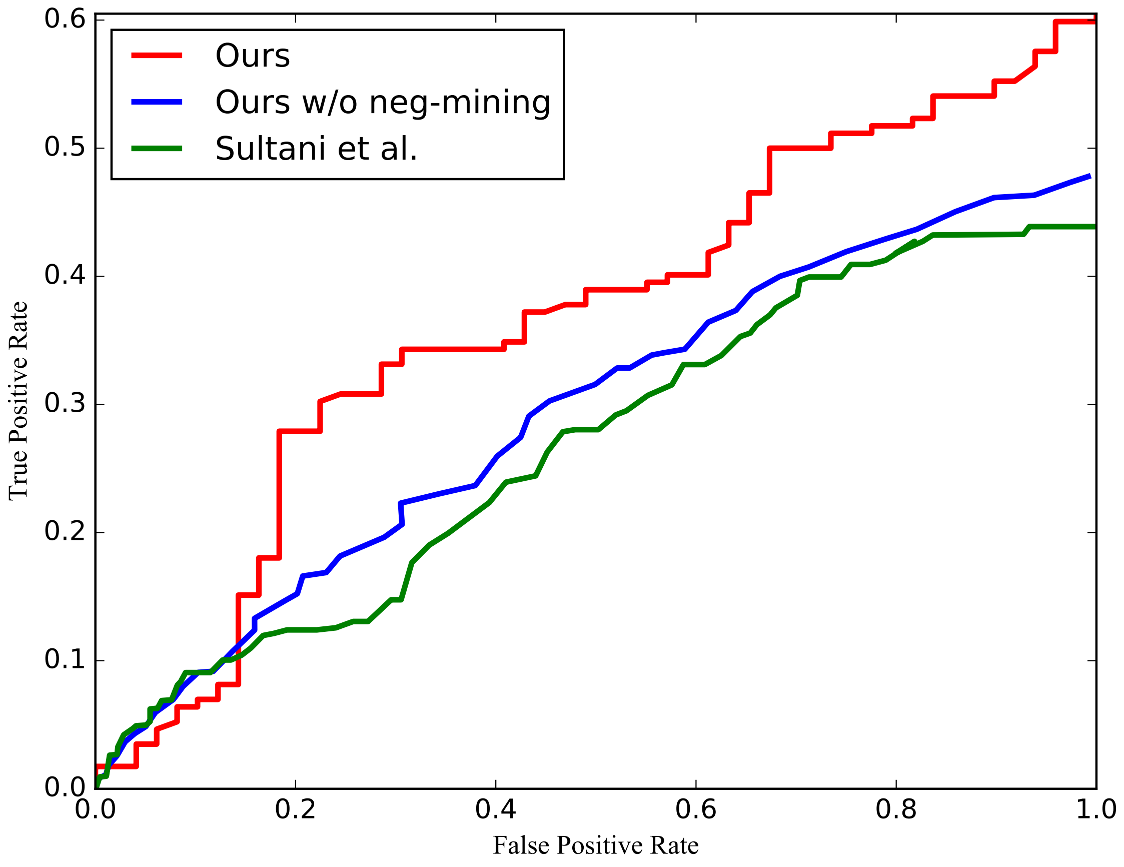

Figure 5.6: The mean ROC curves for four actions of THUMOS'14. The results are shown for Sultani *et al.*[86] (green), proposed method (red) and proposed method without negative mining (blue).

In untrimmed videos, there often exist other unrelated actions besides the action of interests. For example, "walking" and "picking up a golf ball" are considered as unrelated actions when detecting "GolfSwing" in video. We denote clips which have positive ground truth annotation as positive clips, and the other clips as negative clips (*i.e.*clips contain only unrelated actions). If we randomly select negative samples for training, the number of boxes on unrelated actions is much smaller than that of background boxes (*i.e.*boxes capturing only image background). Thus the trained model will have no capability to distinguish action of interest and unrelated actions.

To this end, we introduce a so called **negative sample mining** process. Specifically, when initializing the TPN, we only use positive clips. Then we apply the model on the whole training video (both positive clips and negative clips). Most false positives in negative clips should include unrelated actions to help our model learn the correlation between action of interest and unrelated actions. Therefore we select boxes in negative clips with the highest scores as **hard negatives** because low scores probably infer image background. In updating TPN procedure, we choose 32 boxes which have IoU with any ground truth greater than 0.7 as positive samples and randomly pick another 16 samples as negative. We also select 16 samples from hard negative pool as negative. Therefore, we efficiently train a model, which is able to distinguish not only action of interest from background, but also action of interest from unrelated actions.

The mean ROC curves of different methods on THUMOS'14 action detection are plotted in Figure 5.6(c). Our method without negative mining performs better than the baseline method Sultani *et al.*[86]. Additionally, with negative mining, the performance is further boosted.

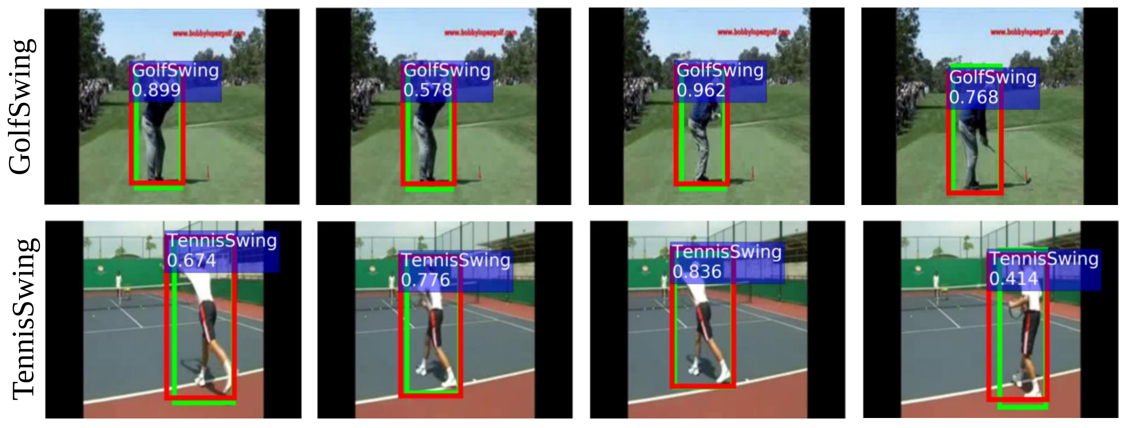

Figure 5.7: Action detection results by THUMOS'14. Red boxes indicate the detections in the corresponding frames, and green boxes denote ground truth. The predicted label is overlaid.

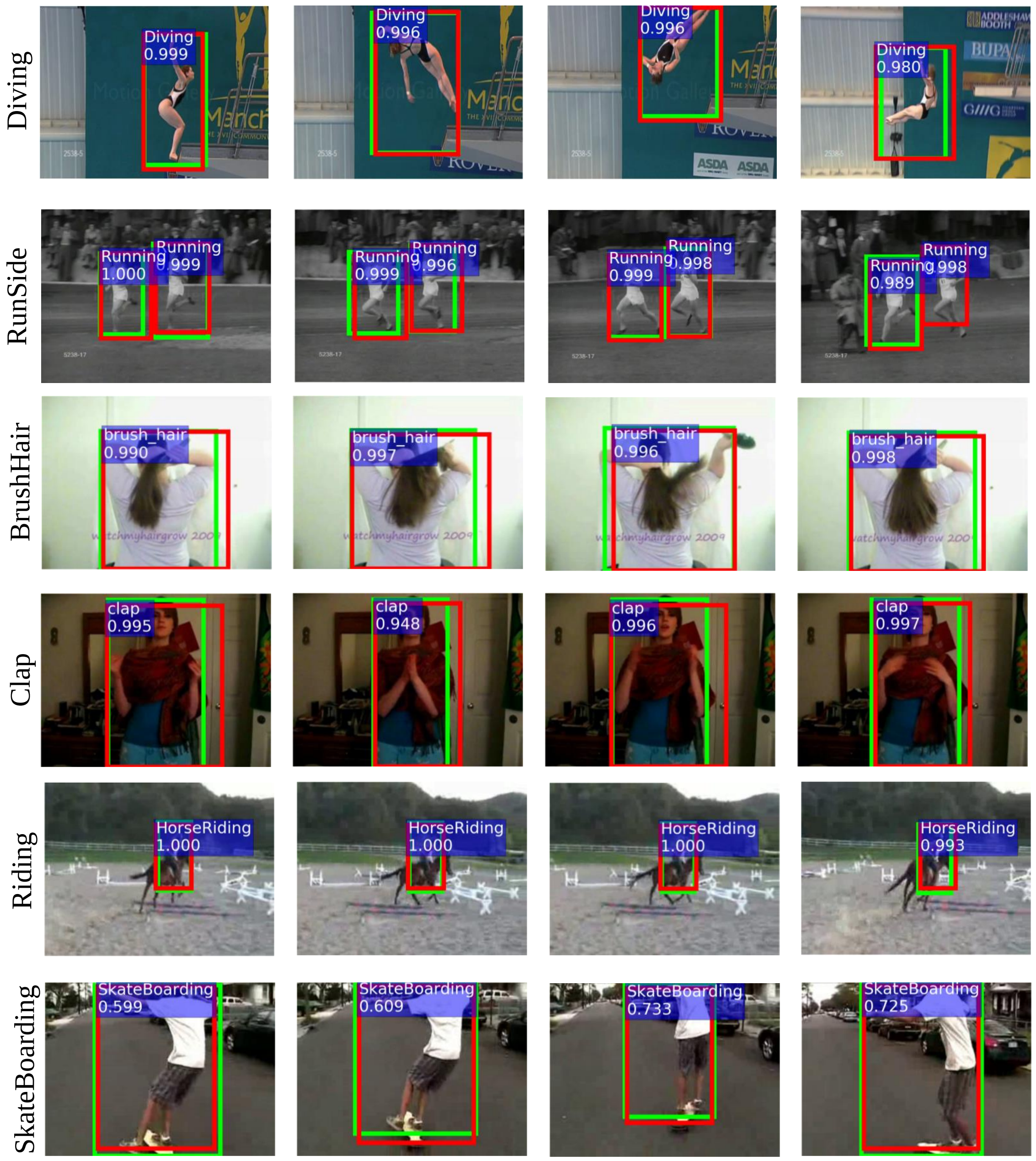

Figure 5.8: Action detection results by T-CNN on UCF-Sports, J-HMDB and UCF-101. Red boxes indicate the detections in the corresponding frames, and green boxes denote ground truth. The predicted label is overlaid.

For qualitative results, we show examples of detected action tubes in videos from UCF-Sports, JH-MDB, UCF-101 (24 actions) and THUMOS'14 datasets (see Figure 5.8). Each block corresponds to a different video that is selected from the test set. We show the highest scoring action tube for each video.

5.4 Discussion

ToI Pooling. To evaluate the effectiveness of ToI pooling, we compare action recognition performance on UCF-101 dataset (101 actions) using C3D [95] and our approach. For the C3D network, we use C3D pre-train model from [95] and fine tune the weights on UCF-101 dataset. In the C3D fine tuning process, a video is divided into 16 frames clips first. Then the C3D network is fed by clips and outputs a feature vector for each clip. Finally, a SVM classifier is trained to predict the labels of all clips, and the video label is determined by the predictions of all clips belonging to the video. Compared to the original C3D network, we use the ToI pooling layer to replace the 5-th 3d-max-pooling layer in C3D pipeline. Similar to C3D network, our approach takes clips from a video as input. The ToI pooling layer takes the whole clip as tube of interest and the pooled depth is set to 1. As a result, each video will output one feature vector. Therefore, it is an end-to-end deep learning based video recognition approach. Video level accuracy is used as the metric. The results are shown in Table 5.6. For a direct comparison, we only use the result from deep network without fusion with other features. Our approach shows a 5.2% accuracy improvement compared to the original C3D. Our ToI pooling based pipeline optimizes the weights for the whole video directly, while C3D performs clip-based optimization. Therefore, our approach can better capture the spatio-temporal information of the entire video. Furthermore, our ToI pooling can be combined with other deep learning based pipelines, such as two-stream CNN [82].

Temporal skip connection. Since we use overlapping clips with temporal stride of 1 in training, a particular frame is included in multiple training clips at different temporal positions. The actual temporal information of that particular frame is lost if we only use the conv5 feature cube to

Table 5.5: Action recognition results of T-CNN on various datasets.

Dataset	Accuracy (%)
UCF-Sports	95.7
J-HMDB	67.2
UCF-101 (24 actions)	94.4

Table 5.6: Video action recognition results on UCF-101.

	C3D [95]	Ours
Accuracy (%)	82.3	87.5

infer action bounding boxes. Especially when the action happens periodically (*i.e.*SwingBench), it always fails to locate a phase of spinning. On the contrary, by combining conv5 with conv2 through temporal skip pooling, temporal order is preserved to localize actions more accurately. To verify the effectiveness of temporal skip pooling in our proposed TPN, we conduct an experiment using our method without skip connection. In other words, we perform bounding box regression to estimate bounding boxes in 8 frames simultaneously using only the conv5 feature cube. As shown in Table 5.3, without skip connection, the performance decreases a lot, demonstrating the advantage of skip connection for extracting temporal order information and detailed motion in original frames.

Computational cost. We conduct experiments on a workstation with one GPU (Nvidia GTX Titan X). For a 40-frames video, it takes 1.1 seconds to generate tube proposals, 0.03 seconds to link tube proposals and 0.9 seconds to predict the action label.

5.5 Summary

In this chapter, we present T-CNN for action detection by extending faster R-CNN from 2D images to 3D spatial-temporal volumes. Previous approaches leverage two streams pipeline to process temporal and spacial streams separately, while our T-CNN, an end-to-end deep learning based pipeline, only takes video clips as input and considers spatial and temporal information simultaneously. We propose Tube Proposal Network (TPN), which is able to detect tube regions where interested actions are performed. We formulate the TPN as a frame-wise prediction problem and leverage skip pooling to preserve temporal information for action localization in 3D volumes. We generalize 2D image Region of Interest (RoI) pooling layer to 3D Tube of Interest (ToI) pooling, which takes tube clip feature map as input and normalizes it into a fixed size 3D cube. It effectively alleviates the problem caused by variable spatial and temporal size of tube proposals. We demonstrate that T-CNN is able to achieve state-of-the-art results on UCF Sports, J-HMDB and UCF Sports datasets.

T-CNN shows promising performance in video action detection. However, there are several applications which require more accurate action localization, such as virtual make-up, augmented reality. In these cases, it is desirable to locate the actions in pixel level. To solve the problem, we will present a new framework to segment video actions, which outputs action bounding boxes in each frame as well as action labels for each pixel.

CHAPTER 6: AN EFFICIENT 3D CNN FOR ACTION/OBJECT SEGMENTATION IN VIDEO

The proposed T-CNN approach in previous chapter is able to detect actions in videos and outputs bounding boxes for localization in each frame. However, in some real world applications, finer human silhouettes are desired since bounding boxes may also include background pixels. The goal of action segmentation is to assign an action label for each input pixel. Compared to action detection, video action segmentation is a more challenging task. Instead of predicting a label for each tube proposal, action segmentation generates labels for each input pixel, which requires much more computational resources and is less robust to random noise.

To precisely segment actions, we develop an encoder-decoder based approach to automatically localize and segment the silhouette of actions in video. The encoder network takes raw pixels as input and generates high-level contextual information through convolutions. The decoder combines the contextual information with local details through another group of convolutions. Finally, a softmax and cross entropy loss are used for pixel-wise prediction for each frame in a clip.

6.1 Generalizing CNN for Dense Prediction from 2D to 3D

Generalizing CNN framework from images (2D) to videos (spatio-temporal 3D) involves more work than simply adding one more dimension. A key challenge is due to the asymmetry between space and time. To address the change in apparent size of an object due to perspective, image pipelines crop and reshape images to a fixed size. One cannot do the same with videos since input videos and the duration of an object track or action in an untrimmed video can vary widely in the temporal dimension. Since an entire video cannot be rescaled to a fixed size, approaches typically process video as sequences of short (*e.g.*8-frame) clips. In this section, we delve into the details of our network design to address these challenges.

6.1.1 3D separable convolution with dilation

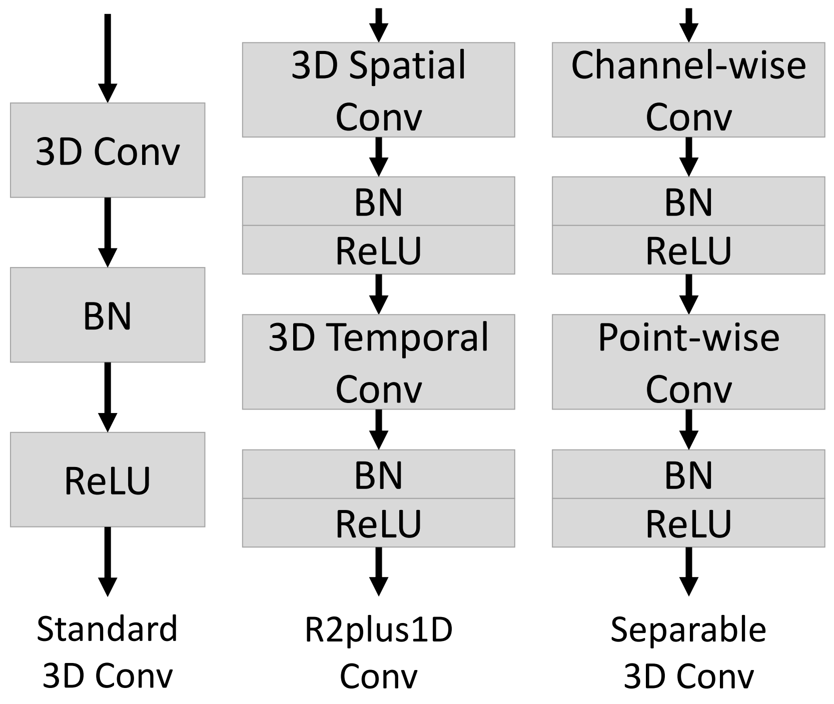

Figure 6.1: Comparison between standard 3D convolution, R2plus1D and 3D separable convolution. R2plus1D factorizes 3D convolution into spatial and temporal convolutions. 3D separable convolution is composed of two convolution modules – channel-wise convolution and point-wise convolution.

3D CNN training is much less efficient than its 2D counterpart, since 3D kernels and feature maps have more parameters. To address pixel-level prediction in video more efficiently, we propose to use 3D separable convolution in lieu of the standard 3D convolution. 3D separable convolution factorizes a standard 3D convolution into a channel-wise 3D convolution and a point-wise 3D convolution with $1 \times 1 \times 1$ kernel. The channel-wise convolution applies a specific filter to each input channel, then point-wise convolution combines the outputs of the channel-wise convolution via a $1 \times 1 \times 1$ convolution. This factorization drastically reduces computation and model size.

A standard 3D convolutional operator takes a $H \times W \times T \times M$ feature map F as input and produces a $H \times W \times T \times N$ feature map G, where H, W and T are the height, width and number of frames (duration) in a clip, respectively. M and N are the number of input/output channels. 3D convolution kernel has shape $K_h \times K_w \times K_t$, where K_h, K_w and K_t are the spatial and temporal dimensions of the kernel. The computation complexity of a standard 3D convolution is:

$$H \times W \times T \times M \times N \times K_h \times K_w \times K_t. \tag{6.1}$$

On the other hand, 3D separable convolution is a two-step process. First channel-wise convolution generates the intermediate feature map by applying a specific filter to each input channel (M channels in total). The computational complexity of channel-wise convolution is:

$$H \times W \times T \times M \times K_h \times K_w \times K_t. \tag{6.2}$$

where the kernel of channel-wise convolution has size $K_h \times K_w \times K_t$. Then point-wise one-by-one convolution projects intermediate feature map to the final output of N channels with computational complexity:

$$H \times W \times T \times M \times N. \tag{6.3}$$

The computational cost of the standard 3D convolution is the multiplication of the feature map dimension (H, W, T), input channel dimension M, output channel dimension N and kernel dimension (K_h, K_w, K_t); while in 3D separable convolution, channel-wise convolution is irrelevant to output filters and point-wise convolution is isolated from kernel dimension. Therefore, the computational cost reduction of 3D separable convolution is:

$$\frac{H \times W \times T \times M \times K_h \times K_w \times K_t + H \times W \times T \times M \times N}{H \times W \times T \times M \times N \times K_h \times K_w \times K_t} = \frac{1}{N} + \frac{1}{K_h \times K_w \times K_t}. \tag{6.4}$$

For example, a 3D separable convolution with $3 \times 3 \times 3$ kernel dimension and 512 input/output size only needs about a quarter of computational resource compared to that of a standard 3D convolution, leading to significant inference computation reduction.

Meanwhile, in R2plus1D convolution, the standard 3D convolution is factorized into spatial and temporal convolutions. The input feature map with M channels passes through spatial convolution with kernel shape $K_h \times K_w \times 1$ and generates a M'-channel intermediate feature map. Then, the intermediate result goes through the temporal convolution with kernel shape $1 \times 1 \times K_t$ and generates the final output. Compared to standard 3D convolution, the computational cost reduction of R2plus1D convolution is:

$$\frac{H \times W \times T \times M \times M' \times K_h \times K_w + H \times W \times T \times M' \times N \times K_t}{H \times W \times T \times M \times N \times K_h \times K_w \times K_t} = \frac{M'}{N \times K_t} + \frac{M'}{M \times K_h \times K_w} \tag{6.5}$$

The computational reduction depends on the number of intermediate filters M'.

Dilated convolution. Dilation rate γ_a is an attribute of convolution operation. It has the ability to increase the receptive field size, while maintaining the computational cost by skipping $\gamma_a - 1$ entities per valid one in one of dimensions (x, y, t). Adding dilation rates helps capture multi-scale feature representations. Specifically in 3D separable convolution, the dilation rate is only applied to channel-wise convolution, since kernel size of point-wise convolution is fixed at $1 \times 1 \times 1$. Moreover, due to the asymmetry between space and time in video, dilation rate in 3D convolution can be divided into two parts – spatial rate γ_s and temporal rate γ_t. By using the dilation rate, the channel-wise convolution can be expressed as follows:

$$\hat{G}_{h,w,t,m} = \sum_{i,j,k} \hat{K}_{i,j,k,m} \cdot F_{h+\gamma_s \cdot i, w+\gamma_s \cdot j, t+\gamma_t \cdot k, m}. \tag{6.6}$$

As shown in Eq. 6.6, the computational cost does not change by adding the dilation rate.

6.1.2 Network architecture

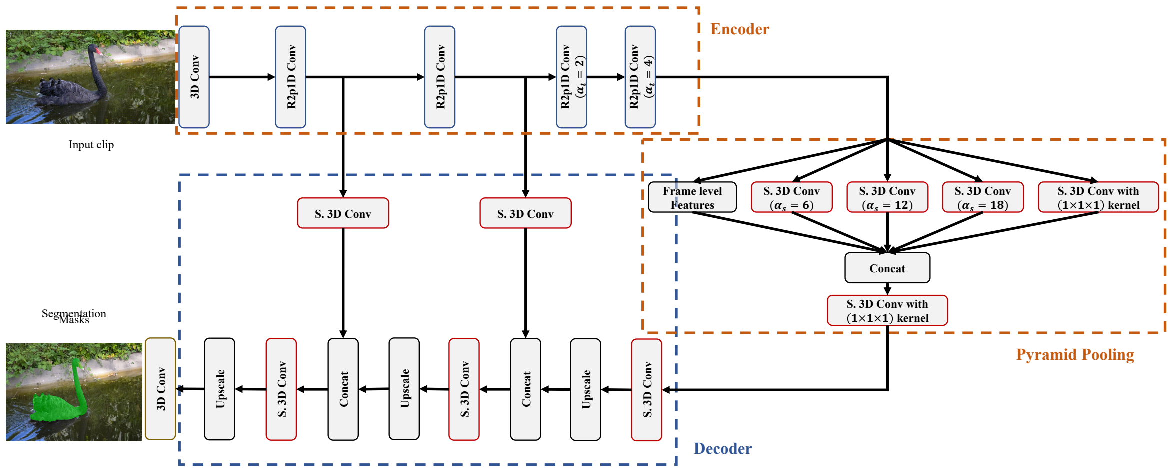

Figure 6.2: The network architecture of our method for video object segmentation. It has three components: an encoder (feature extractor), a pyramid pooling module and a decoder to recover the spatial-temporal details gradually. The encoder, shown in the top-left, takes pixel values as input, extracts features layer by layer, and generates rich contextual feature as the final output. Moreover, its intermediate results are merged with the decoder module. The pyramid pooling module connects the encoder and decoder modules. It varies the receptive field size by modifying the spatial stride. "S. 3D Conv" indicates the 3D separable convolution. "Frame level features" are composed by a reduced average pooling and duplication to generate the frame level features with the same dimension as the other branches.

The proposed 3D CNN architecture for action/object segmentation in video is illustrated in Figure 6.2. The network builds upon an encoder-decoder structure for image semantic segmentation. A video is divided into 8-frame clips as input to the network. In the encoder module, 3D convolution are performed. To capture higher level information, the spatial and temporal sizes are reduced. In order to generate the pixel-wise segmentation map for each frame in the original size, 3D upsampling is used in the decoder module to increase the resolution of feature maps. To capture spatial and temporal information at different scales, a concatenation with the corresponding feature maps from the encoder module is employed after each 3D upsampling layer. Finally, a segmentation branch is used for pixel-wise prediction (*i.e*. background or object foreground) for each frame in a clip.

In the encoder, we adopt R2plus1D [96] as feature extractor to leverage its pre-trained model on large scale action recognition dataset. We modify the last three groups of convolutional layers by adjusting the dilation rate to keep temporal stride as 4. We also insert a 3D pyramid pooling before the decoder. In the decoder, the up-sampling layer group is composed of a tri-linear interpolation layer, a 3D separable convolution layer as well as a feature concatenation layer incorporating encoder feature.

3D Pyramid Pooling. Compared to the 2D counterpart [8], pyramid pooling in 3D is more challenging. In spatial domain, multi-scale information should be captured. In temporal domain, detailed motion information should be preserved as well. Therefore, our 3D pyramid pooling has 5 branches as shown in Figure 6.2. The details are listed as follows:

1. 3D separable convolutions with kernel size $3 \times 3 \times 3$ and different spatial dilation rates $\gamma_s = 6, 12, 18$. The convolutions with multiple spatial dilation rates are able to capture multi-scale information.
2. 3D separable convolution with kernel size $1 \times 1 \times 1$ – a default layer for dense prediction.
3. There are two steps to get the frame feature. First, input is average pooled with kernel $H \times W \times 1$. For the average pooling output, its spatial dimension is 1, while its temporal

length and filter size are the same as those of the input. Then the average pooling output is upscaled to the input size.

By concatenating the 5 branches together, and applying a $1 \times 1 \times 1$ convolution, the final output is obtained and fed into the decoder module.

6.2 Experiments

Implementation details. To verify the effectiveness of the proposed 3D CNN framework for action/object segmentation in videos, we evaluate our approach on two video object segmentation datasets – DAVIS'16 [73] and Segtrack-v2 [56], and a video action segmentation dataset – J-HMDB [33]. We adopt the Adam optimizer with initial learning rate 10^{-4}, exponential decay rate 0.95 and decay step 1 epoch. The model is trained for 100 epochs with exponential decay. In 3D spatial pyramid pooling module, a large crop size is required to ensure the convolution kernel with spatial dilation rate is effective; otherwise, the filter weights with large dilation rate are mostly applied to the padded zero region. Therefore, We employ a spatial crop size of 384 during both training and testing. During training, we randomly scale the resolution of input clips with ratio $[0.5, 2]$ and apply a random horizontal flip.

Experiments on DAVIS'16. Densely Annotated Video Segmentation 2016 (DAVIS'16) dataset is a benchmark dataset for video object segmentation. It consists of 50 videos with 3455 annotated frames. Consistent with most prior work, we conduct experiments on the 480p videos with a resolution of 854×480 pixels. 30 videos are used for training and 20 for validation. We adopt the same evaluation setting in [73]. There are three parts. **Region Similarity** $\mathcal{J}$, which is obtained by IoU (Intersection over Union) between the prediction and the ground-truth segmentation map. **Contour Accuracy** $\mathcal{F}$ measures the contours accuracy. **Temporal Stability** $\mathcal{T}$ tracks the temporal consistency in a video. For the first two evaluation, we report the mean, recall and decay. For the third one, we report the average.

Table 6.1: Overall results of region similarity ($\mathcal{J}$), contour accuracy ($\mathcal{F}$) and temporal stability ($\mathcal{T}$) for different approaches. $\uparrow$ means the higher the better and $\downarrow$ means the lower the better.

Measure		MotAdapt [81]	LSMO [93]	PDB [83]	ARP [45]	FSEG [31]	LMP [92]	FST [43]	CUT [12]	NLC [12]	Ours
$\mathcal{J}$	Mean $\uparrow$	77.2	78.2	77.2	76.2	70.7	70.0	55.8	55.2	55.1	78.3
	Recall $\uparrow$	87.8	89.1	90.1	91.1	83.5	85.0	64.9	57.5	55.8	91.1
	Decay $\downarrow$	5.0	4.1	0.9	7.0	1.5	1.3	0.0	2.2	12.6	2.3
$\mathcal{F}$	Mean $\uparrow$	77.4	75.9	74.5	70.6	65.3	65.9	51.1	55.2	52.3	77.2
	Recall $\uparrow$	84.4	84.7	84.4	83.5	73.8	79.2	51.6	61.0	51.9	84.7
	Decay $\downarrow$	3.3	3.5	-0.2	7.9	1.8	2.5	2.9	3.4	11.4	4.9
$\mathcal{T}$	Mean $\downarrow$	27.9	21.2	29.1	39.3	32.8	57.2	36.6	27.7	42.5	22.0

We compare our results with several unsupervised approaches, since our approach does not require any manual annotation or prior information about the object to be segmented. We cannot compare directly to semi-supervised approaches that require the ground truth segmentation map in the first frame of each test video to be given. Table 6.1 summarizes the performance of our method against the state-of-the-art unsupervised approaches on DAVIS'16. Our method achieves the best performance in all performance metrics. Compared to ARP [45], the previous state-of-the-art unsupervised approach, our method achieves 5% gain in contour accuracy ($\mathcal{F}$) and 15% gain in temporal stability ($\mathcal{T}$), demonstrating that 3D CNN can effectively take advantage of the temporal information in video frames to achieve temporal segmentation consistency. We also present the qualitative results on four video sequences in Figure 6.3.

Experiments on Segtrack-v2. SegTrack-v2 [56] contains 14 video clips with 24 objects and 947 frames. Pixel-level object mask on each frame is provided. We adopt the same evaluation setting in [44] reports the mean Intersection over Union. For videos with multiple instances with individual ground-truth segmentation mask, we group them as a single foreground for evaluation. Compared with other unsupervised video object segmentation methods, our proposed approach outperforms all of them.

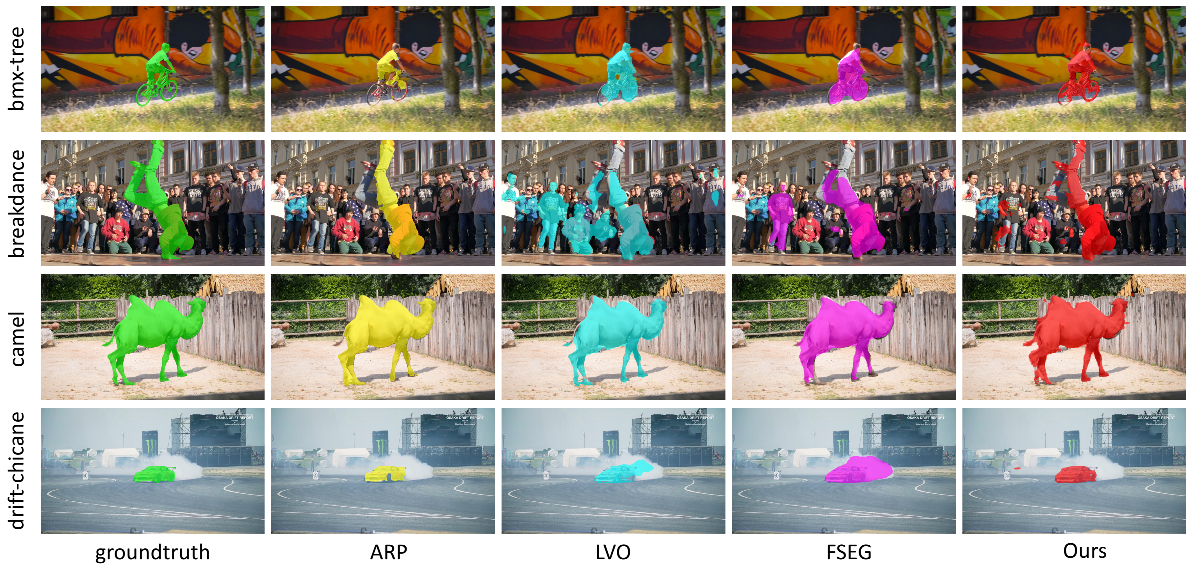

Figure 6.3: Qualitative results of the proposed approach (red), ARP (yellow), LVO (cyan) and FSEG (magenta) on selected frames from DAVIS dataset.

Table 6.2: Video object segmentation results on Segtrack-v2 dataset. We compare the performance of our approach with other state-of-the-art unsupervised approaches.

Method	FST [70]	KEY [55]	LSMO [91]	FSEG [31]	Ours
mean IoU	53.5	57.3	53.7	61.4	**62.1**

Experiments on J-HMDB. The J-HMDB dataset consists of 928 videos with 21 different actions. There are three train-test splits and the evaluation is done on the average results over the three splits. We leverage the mask annotation provided from J-HMDB dataset and train our semantic segmentation pipeline to segment the foreground action. Then the region with maximum area is selected through connected component [26]. On the final feature map of each frame, we crop a box to tightly surround the selected foreground region and resize the box into a fixed shape. Finally, the cropped feature map goes through a classifier to predict action classes. For evaluation, we followed the metrics in [20]. Mean IoU is computed as the average over the IoU of each test sample. In addition, frame-level mean Average Precision (mAP) is evaluated as well. Since bounding boxes detection can be obtained by selecting the tightest rectangle region enclosing the segmentation mask, we are also able to compare with the state-of-the-art action detection approaches.

Table 6.3: Comparison with the state-of-the-art action segmentation and detection approaches on J-HMDB. The first part of the table shows the mean Intersection-over-Union (IoU) of action segmentation approaches and the second part shows the frame level mean Average Precision (mAP) of CNN based action detection approaches.

	mean IoU	frame-mAP ($\alpha = 0.5$)
Lu *et al.* [60]	48.8	–
Gavrilyuk *et al.* [20]	54.2	–
Kalogeiton *et al.*[38]	–	65.7
Duarte *et al.*[11]	–	64.6
Hou *et al.*[79]	–	61.3
Ours	**68.1**	**68.9**

Table 6.3 reports the action segmentation/detection results of our method and the state-of-the-art approaches. It is evident that our method outperforms these methods considerably in evaluation metrics. Figure 6.4 presents both action segmentation (in red) and bounding boxes detection (in yellow) results on several video sequences from J-HMDB.

6.2.1 Ablation Study

To better understand the contribution of each component in our proposed approach, we conduct video object segmentation on DAVIS'16 with different settings, summarized in Table 6.4 and Table 6.5.

Dilation rate. We adopt R2Plus1D pre-trained model in our approach, and increase the feature map's temporal size of last two convolution layer group by specifying the dilation rate. As shown in first section of Table 6.4, the performance is boosted by increasing the number of frames in the final feature map. However, too large feature map (8 frames) will cause out of memory error, that is the reason we stop at 4 frames.

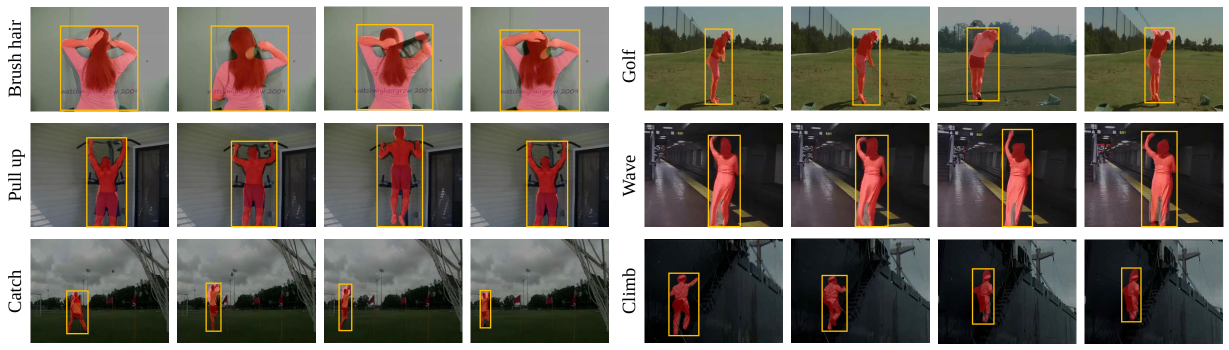

Figure 6.4: Action segmentation and detection results obtained by our method on the J-HMDB dataset. Red pixel-wise segmentation maps show the predictions, and the yellow boxes show the bounding boxes generated from the segmentation maps.

Table 6.4: Ablation study of our method for video object segmentation on DAVIS-16. In the first section of the Table, we investigate various temporal size of the final feature map by increasing the temporal dilation rate in the last two convolutional layer groups. In the second section of the Table, we explore different settings in 3D pyramid pooling module, which includes various receptive fields and whether frame-level features (FF) are included or not. Specifically, the numbers in the parentheses () indicate spatial dilation rates of branches. We compare performance with different sizes of feature maps, with or without 3D pyramid pooling layer. Mean IoU is used as evaluation metric.

Final feature map dim.	3D Pyramid Pooling	$\mathcal{J}$-Mean
$1 \times 20 \times 20$	–	64.5
$2 \times 20 \times 20$	–	69.1
$4 \times 20 \times 20$	–	71.3
$4 \times 20 \times 20$	(6, 12, 18)	78.1
$4 \times 20 \times 20$	(6, 12, 18) + FF	78.3
$4 \times 20 \times 20$	(6, 12, 18, 24)	77.9
$4 \times 20 \times 20$	(8, 16, 24)	74.9

3D pyramid pooling. As shown in the second section of Table 6.4, inserting 3D pyramid pooling module improves the segmentation accuracy by over 5%. We experiment multiple branches with different spatial dilation rates (as noted in the bracket). According to the experimental results, including more branches with larger receptive field (4 branches) or larger temporal stride is not helpful. When the receptive field size is close to or larger than the feature map size, most of the filters cannot capture any useful information, since they only cover padded zeros instead of valid area. Segmentation accuracy further improves when frame-level features are added.

3D separable convolution. The proposed pipeline leverages 3D separable convolution in pyramid pooling and decoder. We perform an experiment by replacing all the 3D separable convolutions with R2plus1D convolutions and standard 3D convolutions. The comparison of performance and computational cost is shown in Table 6.5. All the experiments are carried out on a workstation with a NVIDIA Titan XP GPU and PyTorch. We only take 3D pyramid pooling and decoder part during inference into measurement, since all of them share the same pre-trained model. “Operations” counts the total number of additions and multiplications. “GPU Mem.” is the size of allocated GPU memory. With the same input (a 8 frames clip with resolution 320×320), 3D separable convolution greatly reduces the computational cost without sacrificing the performance.

Table 6.5: The comparison of performance and computational cost of different 3D convolutions.

Conv. Type	Operations	GPU Mem.	$\mathcal{J}$-Mean
Standard 3D Conv.	136 Billion	255 MB	77.1
R2plus1D Conv.	136 Billion	256 MB	77.6
3D Separable Conv.	6 Billion	11 MB	77.4

6.3 Conclusion

This chapter proposes a fully convolutional 3D CNN pipeline for action/object segmentation in video. The approach leverages a model pre-trained on large-scale action recognition task as an encoder to enable us to perform unsupervised video object segmentation (*i.e.*generate pixel-level object masks without initialization). We also use separable filters to significantly reduce the computational burden of the standard 3D convolutions. Extensive experiments on several benchmark datasets demonstrate the strength of our approach for spatio-temporal action segmentation as well as video object segmentation compared with the state-of-the-art approaches.

CHAPTER 7: CONCLUSION

In this dissertation, we address four fundamental and related problems in video action understanding - action recognition, temporal action localization, spatio-temporal action detection and pixel-wise action segmentation. They have a variety of applications such as sports video analysis, video surveillance and video retrieval.

7.1 Summary

For action recognition, we present a novel method for learning mid-level features in a discriminative, data-driven manner and evaluate it on large-scale action recognition datasets. The DaMN features are selected on the basis of category-level mutual pairwise similarity and are shown to convincingly outperform existing approaches, both semantic as well as data-driven on a broad set of tasks.

For temporal action localization, we present a real-time system for temporal action localization in untrimmed videos that learns discriminative and semantically meaningful sub-actions. Both the number of sub-actions for each action and the sub-actions themselves are discovered automatically. We demonstrate state-of-the-art localization performance on standard action datasets.

For spatio-temporal action detection, we propose an end-to-end Tube Convolutional Neural Network (T-CNN) . It exploits 3D convolutional network to extract effective spatio-temporal features and perform action localization and recognition in a unified framework. Coarse proposal boxes are densely sampled based on the 3D convolutional feature cube and linked for action recognition and localization. Extensive experiments on several benchmark datasets demonstrate the strength of T-CNN for spatio-temporal localizing actions, even in untrimmed videos.

For video object and action segmentation, we propose an encoder-decoder based end-to-end 3D CNN pipeline. The approach leverages a model pre-trained on large-scale action recognition task as an encoder to enable us to perform unsupervised video object segmentation (*i.e.*generate pixel-level object masks without initialization). We also use separable convolution to significantly reduce the computational burden of the standard 3D convolutions. Extensive experiments on several benchmark datasets demonstrate the strength of our approach for spatio-temporal action segmentation as well as video object segmentation compared with the state-of-the-art approaches.

7.2 Future Work

For action recognition with DaMN pair, one natural direction for future work is to explore how our category-level features can be applied to problems outside action recognition (and possibly beyond computer vision). We believe that the image classification experiments where DaMN outperforms existing approaches are a promising sign in this direction.

For temporal action localization with automatic sub-action discovery, one potential direction is to discover sub-actions through deep neural network. CNN feature provides better descriptiveness and Long Short Term Memory (LSTM) module could be utilized to better find the start/end timestamp of actions.

For action detection with 3D CNN, there are three potential directions for future work. First, to get better performance, more advanced CNN structure could be incorporated in the T-CNN pipeline. Second, while pursuing the state-of-the-art performance, a light-weighted version is always desired. Especially for mobile or IoT devices, the computational power of which is limited, current network structure is still too heavy for them. Last, we could extend T-CNN to solve instance segmentation problem by adding one more branch to predict masks for tube proposals.

For video object and action segmentation with 3D CNN, one potential direction is to train a separable 3D CNN encoder for video recognition. Another possibility is to combine with T-CNN to achieve video panoptic segmentation. It is a two branches pipeline, the semantic segmentation branch is able to assign a semantic label for all pixels in frames, while the T-CNN branch is able to localize the foreground instances.

LIST OF REFERENCES

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[2] T. L. Berg, A. C. Berg, and J. Shih. Automatic attribute discovery and characterization from noisy web data. In *European Conference on Computer Vision*, pages 663–676. Springer, 2010.

[3] F. Caba Heilbron, J. Carlos 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*, pages 1914–1923, 2016.

[4] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. One-shot video object segmentation. In *CVPR*, 2017.

[5] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733. IEEE, 2017.

[6] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

[7] J. Chang, D. Wei, and J. W. F. Iii. A video representation using temporal superpixels. In *IEEE Conference on Computer Vision Pattern Recognition*, 2013.

[8] 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. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2018.

[9] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[10] K. Duan, D. Parikh, D. Crandall, and K. Grauman. Discovering localized attributes for fine-grained recognition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3474–3481. IEEE, 2012.

[11] K. Duarte, Y. S. Rawat, and M. Shah. Videocapsulenet: A simplified network for action detection. *arXiv preprint arXiv:1805.08162*, 2018.

[12] A. Faktor and M. Irani. Video segmentation by non-local consensus voting. In *BMVC*, volume 2, 2014.

[13] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1778–1785. IEEE, 2009.

[14] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.

[15] X. Y. Felix, R. Ji, M.-H. Tsai, G. Ye, and S.-F. Chang. Weak attributes for large-scale image retrieval. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2949–2956. IEEE, 2012.

[16] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[17] V. Ferrari and A. Zisserman. Learning visual attributes. In *Advances in Neural Information Processing Systems*, pages 433–440, 2008.

[18] A. Gaidon, Z. Harchaoui, and C. Schmid. Actom sequence models for efficient action detection. In *Computer Vision and Pattern Recognition*, 2011.

[19] A. Gaidon, Z. Harchaoui, and C. Schmid. Temporal localization of actions with actoms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2782–2795, 2013.

[20] K. Gavrilyuk, A. Ghodrati, Z. Li, and C. G. Snoek. Actor and action video segmentation from a sentence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5958–5966, 2018.

[21] R. Girshick. Fast r-cnn. In *IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[22] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[23] G. Gkioxari and J. Malik. Finding action tubes. In *Computer Vision and Pattern Recognition*, 2015.

[24] K. Hara, H. Kataoka, and Y. Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6546–6555, 2018.

[25] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[26] L. He, Y. Chao, K. Suzuki, and K. Wu. Fast connected-component labeling. *Pattern Recognition*, 42(9):1977–1987, 2009.

[27] R. Hou, A. R. Zamir, R. Sukthankar, and M. Shah. DaMN–discriminative and mutually nearest: Exploiting pairwise category proximity for video action recognition. In *European Conference on Computer Vision*, 2014.

[28] A. Jain, A. Gupta, M. Rodriguez, and L. S. Davis. Representing videos using mid-level discriminative patches. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2571–2578, 2013.

[29] M. Jain, J. van Gemert, H. Jégou, P. Bouthemy, and C. Snoek. Action localization with tubelets from motion. In *Computer Vision and Pattern Recognition*, 2014.

[30] M. Jain, J. C. van Gemert, T. Mensink, and C. Snoek. Objects2action: Classifying and localizing actions without any video example. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, 2015.

[31] S. D. Jain, B. Xiong, and K. Grauman. FusionSeg: Learning to combine motion and appearance for fully automatic segmentation of generic objects in videos. *arXiv preprint arXiv:1701.05384*, 2017.

[32] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black. Towards understanding action recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 3192–3199, 2013.

[33] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black. Towards understanding action recognition. In *ICCV*, 2013.

[34] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, 2013.

[35] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[36] Y. 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, 2014.

[37] R. Joseph and F. Ali. Yolo9000: Better, faster, stronger. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[38] V. Kalogeiton, P. Weinzaepfel, V. Ferrari, and C. Schmid. Action tubelet detector for spatio-temporal action localization. In *ICCV*, 2017.

[39] S. Karaman, L. Seidenari, and A. Del Bimbo. Fast saliency based pooling of fisher encoded dense trajectories. In *ECCV THUMOS Workshop*, volume 1, page 5, 2014.

[40] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732, 2014.

[41] Y. Ke, R. Sukthankar, and M. Hebert. Event detection in crowded videos. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.

[42] C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda. Learning systems of concepts with an infinite relational model. In *AAAI*, volume 3, page 5, 2006.

[43] M. Keuper, B. Andres, and T. Brox. Motion trajectory segmentation via minimum cost multicuts. In *ICCV*, 2015.

[44] A. Khoreva, F. Perazzi, R. Benenson, B. Schiele, and A. Sorkine-Hornung. Learning video object segmentation from static images. *arXiv preprint arXiv:1612.02646*, 2016.

[45] Y. J. Koh and C.-S. Kim. Primary object segmentation in videos based on region augmentation and reduction. In *CVPR*, 2017.

[46] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[47] 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*, pages 2556–2563. IEEE, 2011.

[48] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and simile classifiers for face verification. In *International Conference on Computer Vision*, 2009.

[49] K.-T. Lai, F. X. Yu, M.-S. Chen, and S.-F. Chang. Video event detection by inferring temporal instance labels. In *CVPR*, 2014.

[50] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 951–958. IEEE, 2009.

[51] T. Lan, Y. Wang, and G. Mori. Discriminative figure-centric models for joint action localization and recognition. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2003–2010, 2011.

[52] T. Lan, Y. Zhu, A. Roshan Zamir, and S. Savarese. Action recognition by hierarchical mid-level action elements. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, 2015.

[53] 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*, pages 1–8. IEEE, 2008.

[54] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[55] Y. J. Lee, J. Kim, and K. Grauman. Key-segments for video object segmentation. In *ICCV*, 2011.

[56] F. Li, T. Kim, A. Humayun, D. Tsai, and J. M. Rehg. Video segmentation by tracking many figure-ground segments. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2192–2199, 2013.

[57] J. Liu, B. Kuipers, and S. Savarese. Recognizing human actions by attributes. In *CVPR 2011*, pages 3337–3344. IEEE, 2011.

[58] J. Liu, J. Luo, and M. Shah. Recognizing realistic actions from videos in the wild. In *Computer vision and pattern recognition, 2009. CVPR 2009. IEEE conference on*, pages 1996–2003. IEEE, 2009.

[59] 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*, pages 3431–3440, 2015.

[60] J. Lu, J. J. Corso, et al. Human action segmentation with hierarchical supervoxel consistency. In *CVPR*, 2015.

[61] K. K. Maninis, S. Caelles, Y. Chen, J. Pont-Tuset, and L. V. Gool. Video object segmentation without temporal information. *IEEE TPAMI*, PP(99):1–1, 2018.

[62] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2624–2637, 2013.

[63] MEXaction2. `http://mexculture.cnam.fr/xwiki/bin/view/Datasets/Mex+action+dataset`, 2015.

[64] F. Negin and F. Bremond. Human action recognition in videos: A survey. *INRIA Technical Report*, 2016.

[65] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

[66] D. Oneata, J. Verbeek, and C. Schmid. Efficient action localization with approximately normalized fisher vectors. In *Computer Vision and Pattern Recognition*, 2014.

[67] D. Oneata, J. Verbeek, and C. Schmid. The LEAR submission at thumos 2014, 2014.

[68] D. N. Osherson, J. Stern, O. Wilkie, M. Stob, and E. E. Smith. Default probability. *Cognitive Science*, 15(2):251–269, 1991.

[69] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell. Zero-shot learning with semantic output codes. In *Advances in neural information processing systems*, pages 1410–1418, 2009.

[70] A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *ICCV*, 2013.

[71] D. Parikh and K. Grauman. Relative attributes. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 503–510. IEEE, 2011.

[72] X. Peng and C. Schmid. Multi-region two-stream r-cnn for action detection. In *European Conference on Computer Vision (ECCV)*, pages 744–759, 2016.

[73] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016.

[74] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *Computer Vision and Pattern Recognition*, 2011.

[75] 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 (NIPS)*, pages 91–99, 2015.

[76] M. Rodriguez, A. Javed, and M. Shah. Action mach: a spatio-temporal maximum average correlation height filter for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[77] M. Rohrbach, M. Stark, and B. Schiele. Zero-shot learning in a large-scale setting. In *Computer Vision and Pattern Recognition*, 2011.

[78] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[79] H. Rui, C. Chen, and M. Shah. Tube convolutional neural network (t-cnn) for action detection in videos. In *ICCV*, 2017.

[80] Z. Shou, D. Wang, and S.-F. Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *Computer Vision and Pattern Recognition*, 2016.

[81] M. Siam, C. Jiang, S. Lu, L. Petrich, M. Gamal, M. Elhoseiny, and M. Jagersand. Video segmentation using teacher-student adaptation in a human robot interaction (hri) setting. *arXiv preprint arXiv:1810.07733*, 2018.

[82] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems (NIPS)*, pages 568–576, 2014.

[83] H. Song, W. Wang, S. Zhao, J. Shen, and K.-M. Lam. Pyramid dilated deeper convlstm for video salient object detection. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[84] K. Soomro, H. Idrees, and M. Shah. Action localization in videos through context walk. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, 2015.

[85] 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.

[86] W. Sultani and M. Shah. What if we do not have multiple videos of the same action? – video action localization using web images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[87] C. Sun, S. Shetty, R. Sukthankar, and R. Nevatia. Temporal localization of fine-grained actions in videos by domain transfer from web images. In *ACM Multimedia*, 2015.

[88] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4597–4605, 2015.

[89] K. Tang, L. Fei-Fei, and D. Koller. Learning latent temporal structure for complex event detection. In *Computer Vision and Pattern Recognition*, 2012.

[90] Y. Tian, R. Sukthankar, and M. Shah. Spatiotemporal deformable part models for action detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2642–2649, 2013.

[91] P. Tokmakov and K. Alahari. Learning Video Object Segmentation with Visual Memory. In *ICCV*, 2017.

[92] P. Tokmakov, K. Alahari, and C. Schmid. Learning motion patterns in videos. *CVPR*, 2017.

[93] P. Tokmakov, C. Schmid, and K. Alahari. Learning to segment moving objects. *International Journal of Computer Vision*, 127(3):282–301, 2019.

[94] L. Torresani, M. Szummer, and A. Fitzgibbon. Efficient object category recognition using classemes. *European Conference on Computer Vision*, pages 776–789, 2010.

[95] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, 2015.

[96] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 2018.

[97] P. Voigtlaender and B. Leibe. Online adaptation of convolutional neural networks for video object segmentation. *arXiv preprint arXiv:1706.09364*, 2017.

[98] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3169–3176. IEEE, 2011.

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

[100] H. Wang and C. Schmid. LEAR-INRIA submission for the THUMOS workshop, 2013.

[101] L. Wang, Y. Qiao, and X. Tang. Action recognition and detection by combining motion and appearance features. In *THUMOS'14 Action Recognition Challenge*, 2014.

[102] L. Wang, Y. Qiao, and X. Tang. Video action detection with relational dynamic-poselets. In *European Conference on Computer Vision (ECCV)*, pages 565–580, 2014.

[103] L. Wang, Y. Qiao, X. Tang, and L. V. Gool. Actionness estimation using hybrid fully convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2708–2717, June 2016.

[104] Y. Wang and G. Mori. A discriminative latent model of object classes and attributes. *European Conference on Computer Vision*, pages 155–168, 2010.

[105] P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Learning to track for spatio-temporal action localization. In *Computer Vision and Pattern Recognition*, 2015.

[106] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 305–321, 2018.

[107] P. Yan, S. M. Khan, and M. Shah. Learning 4d action feature models for arbitrary view action recognition. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7. IEEE, 2008.

[108] Y. Yang and M. Shah. Complex events detection using data-driven concepts. *European Conference on Computer Vision*, pages 722–735, 2012.

[109] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *Computer Vision and Pattern Recognition*, 2016.

[110] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[111] F. X. Yu, L. Cao, R. S. Feris, J. R. Smith, and S.-F. Chang. Designing category-level attributes for discriminative visual recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 771–778, 2013.

[112] J. Yuan, B. Ni, X. Yang, and A. A. Kassim. Temporal action localization with pyramid of score distribution features. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[113] J. Yuan, Y. Pei, B. Ni, P. Moulin, and A. Kassim. Adsc submission at thumos challenge 2015. In *CVPR THUMOS Workshop*, volume 1, page 2, 2015.

[114] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4694–4702, 2015.

[115] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.