2016

Learning space-time structures for action recognition and localization

BOSTON UNIVERSITY

GRADUATE SCHOOL OF ARTS AND SCIENCES

Dissertation

LEARNING SPACE-TIME STRUCTURES FOR ACTION RECOGNITION AND LOCALIZATION

by

SHUGAO MA

B.E., Fudan University, China, 2006
M.E., Chinese Academy of Sciences, China, 2009

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2016

Approved by

First Reader

Stan Sclaroff, PhD
Professor of Computer Science

Second Reader

Margrit Betke, PhD
Professor of Computer Science

Third Reader

Leonid Sigal, PhD
Senior Research Scientist of Disney Research

Acknowledgments

I would like to thank my advisor Professor Stan Sclaroff for guiding me to my field of research for which I found my passion and enjoyed my work. I also want to thank him for the valuable discussions, the freedom in research and the excellent environment he has offered in the past several years. He has had great influence on me which I will keep as precious asset in my future career: it is not only about ways of doing research, but also on managing works, collaborating with other people and expressing ideas with clear writing.

I would want to thank members my thesis committee Prof. Margrit Betke, Dr. Leonid Sigal, Prof. George Kollios and Prof. Mark Crovella for their helpful comments on the thesis.

I would like to thank my two time internship advisor Dr. Leonid Sigal in Disney Research for his excellent guidance and collaborations in my research.

I would like to thank past and present members of the IVC group for the happy and productive times I have spent with them. Nazli, Jianming, Sarah and Mehrnoosh, we have closely collaborated in research. Diane, Danna, Qinxun, Kun, Jianming and Mikhail, we took many classes together and also helped each other in our works. And I learned a lot from Tai-Peng, Ashwin, Zheng and John during my first several years in PhD study. Lastly, I also want to thank Wenxin and Ajjen for our interesting discussions of research and life.

I would like my beautiful wife Xiaomin for her love and support for the years of my PhD study: it is her company that makes this long journey truly enjoyable and memorable.

LEARNING SPACE-TIME STRUCTURES FOR ACTION RECOGNITION AND LOCALIZATION

SHUGAO MA

Boston University, Graduate School of Arts and Sciences, 2016

Major Professor: Stan Sclaroff, Professor of Computer Science

ABSTRACT

In this thesis the problem of automatic human action recognition and localization in videos is studied. In this problem, our goal is to recognize the category of the human action that is happening in the video, and also to localize the action in space and/or time. This problem is challenging due to the complexity of the human actions, the large intra-class variations and the distraction of backgrounds. Human actions are inherently structured patterns of body movements. However, past works are inadequate in learning the space-time structures in human actions and exploring them for better recognition and localization. In this thesis new methods are proposed that exploit such space-time structures for effective human action recognition and localization in videos, including sports videos, YouTube videos, TV programs and movies. A new local space-time video representation, the hierarchical Space-Time Segments, is first proposed. Using this new video representation, ensembles of hierarchical spatio-temporal trees, discovered directly from the training videos, are constructed to model the hierarchical, spatial and temporal structures of human actions. This proposed approach achieves promising performances in action recognition and localization on challenging benchmark datasets. Moreover, the discovered trees show good cross-dataset generalizability: trees learned on one dataset can be used to

recognize and localize similar actions in another dataset. To handle large scale data, a deep model is explored that learns temporal progression of the actions using Long Short Term Memory (LSTM), which is a type of Recurrent Neural Network (RNN). Two novel ranking losses are proposed to train the model to better capture the temporal structures of actions for accurate action recognition and temporal localization. This model achieves state-of-art performance on a large scale video dataset. A deep model usually employs a Convolutional Neural Network (CNN) to learn visual features from video frames. The problem of utilizing web action images for training a Convolutional Neural Network (CNN) is also studied: training CNN typically requires a large number of training videos, but the findings of this study show that web action images can be utilized as additional training data to significantly reduce the burden of video training data collection.

Contents

List of Tables

List of Figures

List of Abbreviations

BoW	.....	Bag-of-Words
CNN	.....	Convolutional Neural Network
CRF	.....	Conditional Random Field
DP	.....	Dynamic Programming
DSC	.....	Discriminative Sub-Categorization
FV	.....	Fisher Vector encoding
GASTON	.....	GrAph / Sequence / Tree extractiON algorithm
HMDB	.....	Human Motion Database
HMM	.....	Hidden Markov Model
HSTS	.....	Hierarchical Space-Time Segment
IDT	.....	Improved Dense Trajectories
IOU	.....	Intersection Over Union
JHMDB	.....	Joint-annotated Human Motion Database
LSTM	.....	Long Short Term Memory
mAP	.....	mean Average Precision
MRF	.....	Markov Random Field
RGB	.....	Red Green Blue
RNN	.....	Recurrent Neural Network
STIP	.....	Space-Time Interest Point

STS	.....	Space-Time Segments
SVM	.....	Support Vector Machine
UCF	.....	University of Central Florida
UCM	.....	Ultrametic Contour Map
VGG	.....	Visual Geometry Group

Chapter 1

Introduction

Human action recognition is an important topic of interest, due to its wide ranging application in automatic video analysis, video retrieval and more. In this thesis we study the problem of action recognition and localization in realistic video clips. In our experiments we consider video clips collected from sports videos [65], TV programs [59], movies [26] or YouTube [44, 28]. In each video clip, multiple instances of the action that we want to recognize may exist and these actions could be interactions among multiple persons. Irrelevant actions could also be present in the video clip. In the following sections, we first define the problems that we study in this thesis and point out the existing challenges. Subsequently, we describe our proposed approaches to these problems, which are the contributions of this thesis. Finally we give a roadmap of this thesis and list the related publications this thesis is based on.

1.1 Problem Definitions

Two major problems are studied in this thesis: 1) action recognition and spatial localization in temporally trimmed video clips, and 2) action recognition and temporal localization in untrimmed videos.

In the first problem, we assume that the starting and ending frames of a video clip correspond to the start and end of an action respectively. This is a conventional

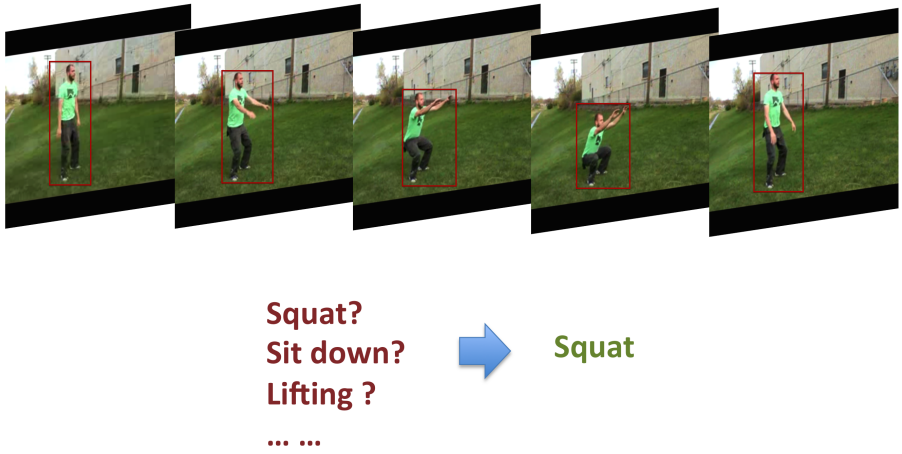

Figure 1.1: Action recognition in temporally trimmed video clips. The starting and ending frames of a video clip are the starting and ending points of the action respectively (*Squat* in this example). We want to classify the action as one of a predefined list of action classes, and also spatially localize the action (*e.g.* the red bounding boxes in this example).

setting in past works, for instance in [50, 77, 79, 41]. In training we assume that only the action class labels of the training video clips are available. In testing we predict both the action label as one of a predefined list of action classes, as well as the spatial location of the action performer(s). Fig. 1.1 illustrates an example of this problem.

In the second problem, we no longer assume that the videos are temporally trimmed. We want to both recognize the class of the action and its temporal duration. We consider two possible scenarios in this problem: 1) the action is fully contained in the video sequence and we observe the complete action, and 2) the action is still ongoing, and we observe the action from its start until some intermediate time point. In testing, in the first scenario, we want to predict the action class and the starting and ending time point of the action in the video, which we refer to as *action detection* in this thesis. In the second scenario, we want to predict the action class of the ongoing action and its starting time point, which we refer to as *action*

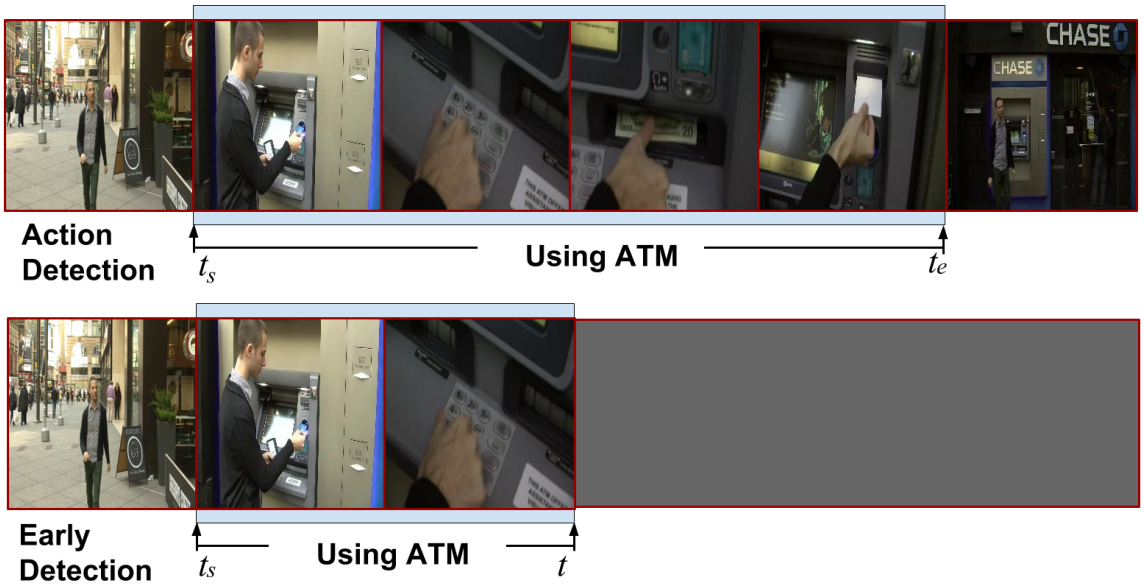

Figure 1.2: We study two problems: activity detection and early detection. For activity detection, we detect the category of the activity and its start and end point. For early detection, we need to detect the category and the start point of an activity after observing only a fraction of the activity. This example sequence contains the activity *using ATM*.

early detection. Early detection is useful in many applications, *e.g.* in human-robot interaction it is desirable to detect the action as early as possible [29, 67] to make the interaction more natural, such as deploying a robot to help an elderly patient stand up before he/she is upright and is risking a fall. Fig 1.2 illustrates both problems. In training, we only need the action labels and starting and ending time points of the actions in each training video.

1.2 Existing Challenges

Action recognition in temporally trimmed videos is an extensively studied problem [50, 68, 57, 54, 94, 54, 100, 77, 79, 41]. However, it remains a challenging problem largely due to the complexity of the human actions and the large intra-class variations

from different action executions and video capturing processes. When the video is not trimmed, recognizing the action and temporally localizing the action becomes even more challenging, especially for relatively long and complex actions.

We highlight four challenges. *First*, the widely used local space-time representations space-time interest points (STIPs) [50] and dense trajectories [76] focus on non-static parts of the video and discard static parts, but non-static *and* relevant static parts in the video are also important for action recognition and localization. *Second*, human actions are inherently structured patterns of body movements and the space-time structures in human actions are crucial for recognizing the action, but past works are inadequate in exploring such structures for action recognition. *Third*, when the action is long and complex, it is difficult to model the temporal structure of the action for accurate action detection / early detection in untrimmed videos. *Fourth*, Convolutional Neural Network (CNN) models are shown to be very effective [69, 22, 21, 8] in learning visual features from videos for action recognition and detection, but the supervised training of a deep CNN requires huge amount of annotated training videos, which are expensive to collect, annotate and process. In the following text we elaborate on these challenges and point out our solutions.

1.2.1 Extracting Local Space-Time Representation of Video

One successful major approach for action recognition and localization builds action models on local space-time representations that are extracted from videos [50, 100, 77, 79, 41]. Among the proposed local space-time representations, space-time interest points (STIPs) [50] and dense trajectories [76] are perhaps the most widely used. However, both STIPs and dense trajectories focus on non-static parts of the video, while the static parts are largely discarded. We argue that both non-static

and relevant static parts in the video are important for action recognition and localization. Firstly, some static parts of the space-time video volume may contain pose information that can be helpful in recognizing human actions. Secondly, extracting both static and non-static body parts is necessary for accurate localization of the whole visible body of the action performer.

To solve such problems, we propose a new local space-time representation that preserves both static and non-static relevant parts of the video for better action recognition and localization. This new representation is organized in a two-level hierarchy. The first level comprises local space-time video segments that may contain the whole human body. The second level comprises local space-time video segments that contain parts of the root. The algorithm we propose to extract this local space-time representation from a video is unsupervised: it does not need any pre-trained body or body part detectors that may be constrained by strong priors of common poses present in the related training set. The representation of parts is multi-grained in that the parts are allowed to overlap: some parts are actually parts of larger parts, e.g. lower leg and whole leg.

1.2.2 Modeling Space-Time Structures of Actions

We argue that as human actions are inherently structured patterns of body movements, the space-time structures in human actions are crucial for recognizing the action. Past works are inadequate in exploring such space-time structures for action recognition, by either discarding structural information [50, 77, 79], encoding only weak structural information [50, 68, 57, 54, 94] or using manually pre-defined structures [62, 87, 32, 85, 86, 63, 64, 82].

In this thesis, we study an approach that learns such space-time structures from

trimmed videos and leverage them for effective action recognition. Intuitively, human actions can be modeled as spatio-temporal graphs, where the graph vertices encode movements of whole body or body parts, and the graph edges encode spatio-temporal relationships between pairs of movement elements, for instance temporal progression, *e.g.*, one movement followed by another movement, spatial composition, *e.g.*, movement of upper body coupled with movement of lower extremities, or even hierarchical relationships of elements, *e.g.*, movement of the body as a whole can decompose into local movements of the limbs. Fig. 1.3 illustrates the example spatial, temporal and hierarchical structures in the action *squat*. A single spatio-temporal structure, however, is unlikely to be sufficient to represent a class of action in all but the simplest scenarios. First, the execution of the action may differ from subject to subject, involving different body parts or different space-time progressions of body part movements. Second, the video capture process introduces intra-class variations due to occlusions or variations in camera viewpoint. Thus, the resulting space-time and appearance variations necessitate using a *collection* of spatio-temporal structures that can best represent the action at large. We propose an approach that automatically discovers a set of space-time tree structures of human actions and builds an ensemble using these trees for action classification and spatial localization.

1.2.3 Modeling Action Progression in Untrimmed Videos

When the video is not trimmed, recognizing the action and temporally localizing the action becomes even more challenging, especially for relatively long and complex actions [1]. For example, the action "making pasta" typically entails *cutting vegetables*, *setting a pot on the fire*, *making boiling water*, *boiling pasta noodles*, *cooking pasta*

[1]Some works refer such long and complex actions as *activities*, *e.g.* as in [28]. We do not make such distinctions here.

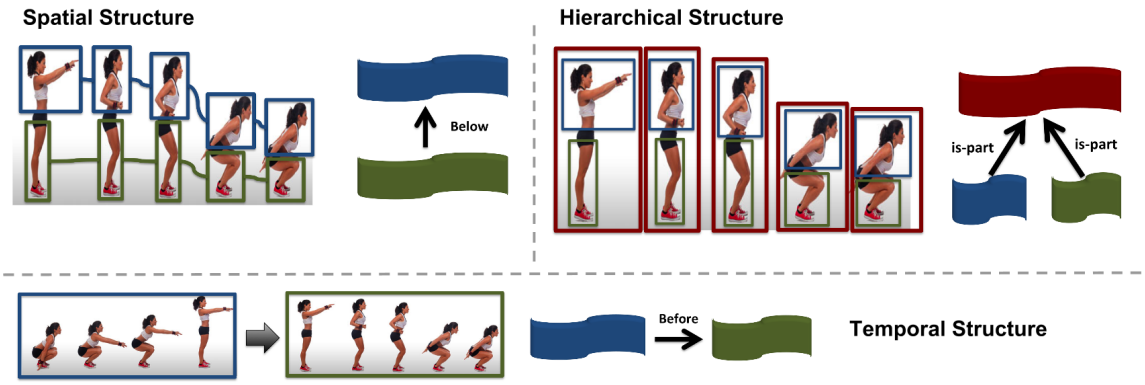

Figure 1.3: Example spatial, temporal and hierarchical structures in the action *squat*.

sauce, and *combining pasta with sauce*. To better detect, *i.e.*, recognize and temporally localize such activities, we argue that it is critically important for the learned detector to model the activities' temporal progression. Recurrent Neural Network (RNN) models can be quite useful in this situation: at each time instant, the prediction is based on both the current observations and the previous model hidden states that provide temporal context for the progression of the action. More specifically, in the Long Short Term Memory (LSTM) [31], a type of RNN, *memory* is used to capture useful patterns of previous observations, and is used in addition to the previous hidden states to provide longer-range context (*e.g.*, as compared to HMMs) for the current prediction. This has been explored in previous work for action detection, *e.g.* in [98] , and is shown to achieve promising results.

However, While RNN models are powerful, using only classification loss in training such models typically fails to properly penalize incorrect predictions. *i.e.*The prediction error is penalized the same no matter how much context the model has already processed. For example, given a video of the action *making pasta*, to output the action class label *preparing coffee* after the detector sees the action up to *combining pasta with sauce* should be penalized more than the same error when the

detector only sees up to *making boiling water*. This defect in training RNN models is especially critical for action detection. Unlike conventional applications of RNNs in machine translation and speech recognition, in which specific output such as words or phonemes continue for a relatively short time, human activities such as *making pasta* may continue for a relatively long period, *e.g.*, several minutes or thousands of video frames. It is thus very important for the model to learn the progression patterns of the activities in training. We propose two novel ranking losses that consider temporal progression of actions, and, when used together with classification loss, significantly improve the action detection and early detection performance.

1.2.4 Training Deep CNN With Limited Video Training Data

For building a good model for action recognition, it is crucial that the input to the model, *i.e.*the features extracted from the video, are representative and discriminative for the actions contained in the videos. Recent works [69, 22, 21, 8] show that deep Convolutional Neural Network models can learn good features from training videos for action recognition and detection. Our LSTM model, mentioned in Sec. 1.2.3, is also built on top of a deep CNN. However, CNN models typically have millions of parameters [7, 47, 70], and usually large amounts of training data are needed to avoid overfitting. For this purpose, work is underway to construct datasets consisting of millions of videos [42]. However, the collection, pre-processing, and annotation of such datasets can require a lot of human effort. Moreover, storing and training on such large amounts of data can consume substantial computational resources.

In contrast, collecting and processing images from the Web is much easier. Annotating the actions in an image is much simpler than annotating actions in a whole

video sequence. Moreover, videos often contain many redundant and uninformative frames, *e.g.*, standing postures, whereas action images tend to focus on discriminative portions of the action (Fig. 6.1). In this thesis, we collect large web action image datasets and study the utilization of such images in training CNN models for action recognition in videos. By extensive experimental evaluation, we show that web action images can be used to significantly reduce the burden of collecting and annotating video training data.

1.3 Contributions

In this thesis, we propose novel approaches that are designed to address each of the challenges described above. The main contributions of this thesis include the following:

- **Hierarchical Space-Time Segments.** We propose a new representation, hierarchical space-time segments, for both action recognition and localization that incorporates multi-grained representation of the parts and the whole body in a hierarchical way. An algorithm is designed to extract the proposed hierarchical space-time segments that preserves both static *and* non-static relevant space time segments as well as their hierarchical and temporal relationships. Such relationships serve for better recognition and localization.

- **Ensemble of Space-Time Trees.** We propose an approach that enables discovery of high-level tree structures that capture space, time and hierarchical relationships among the hierarchical space-time segments. We then further propose a discriminative action model that utilizes both small structures (*i.e.*, words and pairs) and richer, tree structures. This unified formulation achieves

state-of-the-art performance in recognizing and spatially localizing human actions and interactions in realistic benchmark video datasets. We also show generalization of the learned trees by cross-dataset validation: trees learned on one dataset can be used to recognize and localize similar actions of another dataset.

- **Novel ranking losses for training LSTM.** We propose formulations for ranking loss on the detection score and on the discriminative margin of a deep LSTM model to better learn the temporal structures of human actions. Our LSTM model shows significant improvement over LSTM models trained only with classification loss in the tasks of action detection and early detection.

- **Utilizing web action images in training deep CNN.** We study the utility of web action images for video-based action recognition using CNNs. Our findings show that web action images are complementary to video training data. This complementarity is insensitive to the depth of CNNs and is evident in many kinds of actions. Moreover, using web action images can boost the efficiency of CNN training and reduce the burden of collecting video training data. We also collect new web action image datasets which are made publicly available for the research community.

1.4 Roadmap of Thesis

We organize the rest of the thesis as follows:

Chapter 2: Related Work

This chapter reviews related works in action recognition, action spatial localization, action detection and utilization of web action images for action recognition

in videos.

Chapter 3: Hierarchical Space-Time Segments

This chapter describes the novel video representation, hierarchical space-time segments, for action recognition and localization. For an input video, we first apply hierarchical segmentation on each video frame to get a set of segment trees, each of which is considered as a candidate segment tree of the human body. In the second step, we prune the candidates by exploring several cues such as shape, motion, articulated objects' structure and global foreground color. Finally, we track each segment of the remaining segment trees in time both forward and backward. This process yields the final hierarchical space-time segments. These space-time segments are subsequently grouped into tracks according to their space-time overlap. We build simple Bag-of-Words models on the HSTSs, and show promising action localization and recognition results on two benchmark datasets.

Chapter 4: Ensemble of Space-Time Tree

we explore the hierarchical, spatial and temporal relationships among the space-time segments (STSs). This transforms a video into a graph. We discover a compact set of frequent and discriminative tree structures from graphs of training videos and learn discriminative weights for the tree nodes and edges. Finally, we construct action classifiers given the detection responses of these trees. We evaluate our model on three benchmark datasets and also show promising cross-dataset generalizability of the learned trees.

Chapter 5: Learning Action Progression in LSTMs

We introduce two explicit constraints in LSTM training to better capture action progression. The first is a ranking loss on the detection score of the correct category, which constrains the detection score of the correct category to be monotonically non-decreasing as the action progress. The second is a ranking loss on the detection score margin between the correct action category and all other categories, which constrains that this discriminative margin is monotonically non-decreasing. We show state-of-art action detection and early detection performances by our model on a large scale benchmark dataset.

Chapter 6: Utilizing Web Images for Training CNN Models

We start by collecting large web action image datasets. With these datasets, we train CNN models of different depths and analyze the effect of adding web action images to the training set of video frames for different action classes. We also train and evaluate models with varying numbers of action images to explore marginal gain as a function of the web image set size. We find that by combining web action images with video frames in training, a spatial CNN can achieve significant improvement in action recognition accuracy. We also replace videos by images to demonstrate that our performance gains are due to images providing complementary information to that available in videos, and not solely due to additional training data.

Chapter 7: Conclusions and Future Work

This chapter summarizes and discusses the key contributions of the thesis. Some open problems related to human action recognition and localization in videos are also discussed.

1.5 List of Related Papers

Material for this thesis is based on four earlier papers:

1. **Shugao Ma**, Leonid Sigal and Stan Sclaroff. Learning Activity Progression in LSTMs for Activity Detection and Early Detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

2. **Shugao Ma**, Sarah Adel Bargal, Jianming Zhang, Leonid Sigal and Stan Sclaroff. Do Less and Achieve More: Training CNNs for Action Recognition Utilizing Action Images from the Web. *arXiv*, 2015.

3. **Shugao Ma**, Leonid Sigal and Stan Sclaroff. Space-Time Tree Ensemble for Action Recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, oral, 2015.

4. **Shugao Ma**, Jianming Zhang, Nazli Ikizler-Cinbis and Stan Sclaroff. Action Recognition and Localization by Hierarchical Space-Time Segments. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2013.

Chapter 2

Related Work

In this section, we review the related literature on action recognition, spatial action localization and action detection which localize action in time. We also review previous methods that explore web action images for training models that are to be used for action recognition in videos.

2.1 Action Recognition

Action recognition is an important research topic for which a large number of methods have been proposed [88]. Among the previous methods, Bag-of-Words (BoW) representation of videos is a widely used effective technique on which action classification models can be built. In these approaches, local space-time features are often computed from space-time interest point operators [50, 100] or point trajectories [77, 79, 41] . Usually a large codebook, with thousands of words, is built by clustering these local features. The videos are then represented by (normalized) histogram counts of local space-time features over the codebook, on which action classifiers, *e.g.* SVMs, are trained. Advanced feature encoding methods such as Fisher Vector encoding [60] can be used to further improve the performance [78] of these methods. While effective in practice, these representations lack the spatio-temporal structure necessary to model progression or structure of an action.

One way to encode spatio-temporal structures is to impose a fixed spatio-temporal grid over the video and construct a BoW representation by concatenating the BoW representations from each grid cell [50, 68, 57]. While this encodes spatio-temporal structure, the fixed spatio-temporal grid only offers very coarse structural information and typically does not align well with the spatial location or temporal progression of the action. Other works have tried to encode simple structures more explicitly [54, 94]. In [54], quantized local features are augmented with relative space-time relationships between pairs of features; in [94] a visual location vocabulary is computed to incorporate spatio-temporal information.

Richer structures have also been explored. Generative models such as HMMs have been used, *e.g.*, [62, 87, 32]; however, the independence assumptions of HMMs are often not held in practice. Discriminative models, *e.g.*, HCRF, are also widely used [85, 86], [63], [64]. In each of these works a single manually defined structure is used to model human action as a constellation or temporal chain of action parts. In contrast, we model an action as an ensemble of tree structures. More importantly, we discover the structures from data as opposed to defining them by hand. [82] learns mixture models of body parts and spatio-temporal tree structures using annotated human joints on video frames. Our method only requires action labels of the videos.

Subgraph mining has also been used for action recognition [2]. However, subgraph mining techniques only consider exact matching of subgraphs and discover graphs that are frequent, not necessarily discriminative. We use subgraph mining only to produce a large set of candidate subtrees, from which, using the proposed clustering and ranking methods, we discover a compact subset of discriminative and non-redundant subtrees.

Other approaches transform videos into graphs and classify actions based on these

graphs. In contrast to these methods, which attempt to learn a single holistic graph per action class [5, 73] or a graph kernel that measures compatibility between whole graphs [83, 19, 90], we focus on identifying frequent and discriminative subtrees. This allows for a more versatile and compact representation that we also observe is often semantically meaningful.

CNN models learn discriminative visual features at different granularities, directly from data, which may be advantageous in large-scale problems. Such models may implicitly capture higher-level structural patterns in the features learned at the last layers of the CNN model. Ji *et al.* [38] use 3D convolution filters within a CNN model to learn space-time features. [42] train a deep CNN on a dataset of millions of sports videos and study multiple approaches for extending the connectivity of a CNN in the time domain. [69] propose using two separate CNNs to learn spatial and temporal features respectively and train action classifiers on the combined CNN features. [22] utilizes Region-CNN (RCNN) [21] to capture both the action performer and the relevant contextual features to recognize and localize the action. [8] propose a pose-based CNN for action recognition. Recurrent Neural Network (RNN) models, which are capable of capturing temporal dynamics in the video sequence, have also explored for the action recognition task. For example, the Long Short Term Memory (LSTM), a specific type of RNN, is used in [55, 92] for recognizing human actions. While these deep models are effective and produce state-of-art performance on benchmark datasets, typically the models have millions of parameters and the training of such models requires a large amount of training data.

2.2 Action Spatial Localization

Action recognition methods that use holistic representations of the human figure have the potential to localize the action performer, such as motion history images [4], space-time shape models [24] and human silhouettes [84]. But these approaches may not be robust enough to handle occlusions and cluttered backgrounds in realistic videos.

Works that use pre-trained human body or body part detectors also can localize the performer, such as [34, 93, 95]. However, their detectors may be constrained by the human body appearance priors implicitly contained in the training set and may not be flexible enough to deal with varying occlusions and poses in various actions. In [49] the bag of STIP approach was extended beyond action recognition to localization using latent SVM. In this paper, we show that by detecting the learned tree structures of human actions, we can effectively recover the spatial locations of the actions in videos.

Some recent works employ deep Convolutional Neural Network (CNN) models for both action recognition and localization. [23] detect human actions within regions of interest on each frame using a spatial CNN and a motion CNN, and subsequently link the detections in time to construct *action tubes* for localizing the action. Similarly, [89] use spatial and motion CNNs to score foreground proposals at the frame-level and then track high-scoring proposals for localizing an action in space and time. [22] explore context information by adapting the R-CNN [21] to use more than one region for detection while, at the same time, maintaining the ability to localize the action. These deep model based methods show promising results but training the effective CNN models require large amount of training data. None of these CNN approaches directly model the temporal dynamics and structure of an entire ac-

tion action through time which may lead to temporal inconsistency for localization, *e.g.* missing localization on certain frames. Moreover, human bounding box annotations on video frames are required in training in [23, 89, 22]. Note that our method shows promising action localization performance without requiring human bounding box annotations.

2.3 Action Detection

Human action detection aims at recognize as well as temporally localize actions in video sequences. In [43], simple actions are represented as space-time shapes that are matched against over-segmented space-time video volumes. In [99], action detection entailed searching for 3D subvolumes of space-time invariant points. In [49, 72], human actions are modeled as space-time structures, using deformable part models [15]. In [56, 66] discriminative hand-centric features are explored for fine grained action detection in cooking, *i.e.*, relatively short sub-activities such as *chop* and *fill*. In [23], the detector is trained on CNN features extracted from the action tubes in space-time; however, evaluation is on relatively short video clips (*i.e.*, several hundred frames) of relatively short actions. In [98] an LSTM is trained that takes CNN features of multiple neighboring frames as input to detect actions at every frame; while their model is similar to ours, they focus on detecting simple actions such as *stand up* that last only for a few video frames, and the training loss accounts only for classification errors. In this work, we focus on accurately localizing activities that are long and complex by learning and enforcing action progression as part of LSTM learning objective.

Early recognition of human action or activities, *i.e.*, recognizing human actions or activities given partial observations, has also been studied in previous works, *e.g.*,

by using dynamic bag-of-words of space-time features [67], by modeling actions as a sparse sequence of key-frames [64], or by using compositional kernels to hierarchically capture relationships between partial observations [45]. In [29] a structured output SVM is used for recognizing and also temporally localizing events given partial observations. Compared to [29], which is evaluated on lab collected videos of simple human actions, we use deep learning techniques to solve this problem on large scale realistic video dataset of human activities which are often long and complex.

2.4 Utilization of Web Action Images for Action Recognition in Videos

Web action images have been used for training non-CNN models for action recognition [9, 35] and event recognition [12, 81] in videos. Ikizler-Cinbis *et al.* [35] use web action images to train linear regression classifiers for small-scale action classification tasks (5 or 8 action classes). Chen *et al.* [9] use static action images to generate synthetic samples for training SVM action classifiers and evaluate on a small test set of 78 videos comprising 5 action classes. In [12], Duan *et al.* use SVMs trained on SIFT features of web action images in their video event recognition system and evaluate on datasets with 5$\sim$6 different events. Wang *et al.* [81] exploit semantic groupings of Web images for video event recognition and evaluate on the same datasets as [12]. Sun *et al.* [71] localize actions temporally using a domain transfer from web images. In contrast, our work gives the first thorough study on combining web action images with videos for training CNN models for large-scale action recognition.

Chapter 3

Hierarchical Space-Time Segments

Many local space-time representations have been proposed for use in the action recognition task. Among them, space-time interest points (STIPs) [50] and dense trajectories [76] are perhaps the most widely used. One major issue for both STIPs and dense trajectories is that they focus on non-static parts of the video, while the static parts are largely discarded. We argue that both non-static *and* relevant static parts in the video are important for action recognition and localization. There are at least two reasons:

- Some static parts of the space-time video volume can be helpful in recognizing human actions. For example, for the *golf swing* action, instead of just relying on the regions that cover the hands and arms, which have significant motion, the overall body pose can also indicate important information that may be exploited to better discriminate this action from others.
- In many applications, estimating the location of the action performer is also desired in addition to recognizing the action. Extracting only the non-static body parts may not lead to accurate localization of the whole body. Therefore, accounting for both static and non-static parts may help.

In this Chapter, we propose a representation that we call *hierarchical space-time segments* (HSTS) for both action recognition and localization. In this representation,

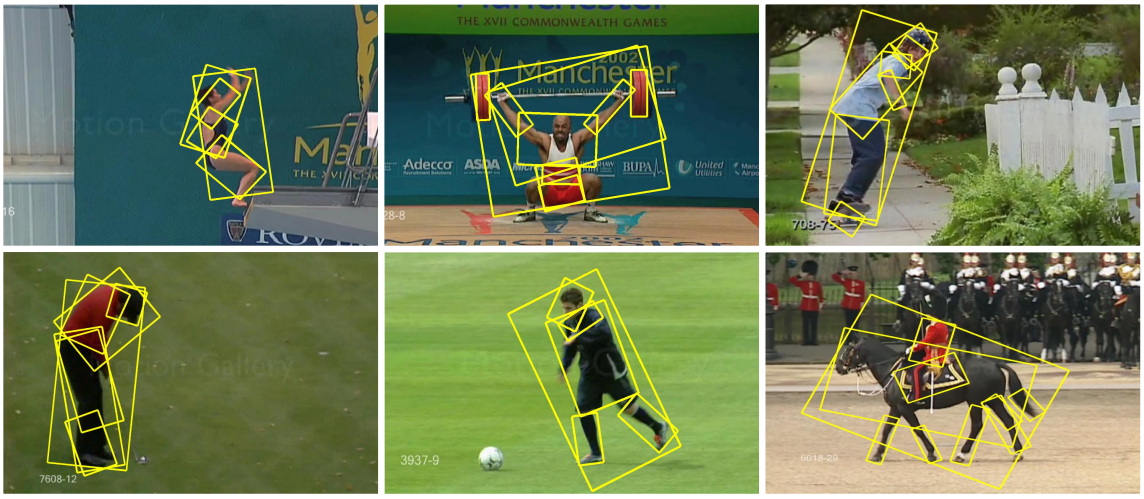

Figure 3.1: Extracted segments from example video frames of the UCF Sports dataset. Yellow boxes outline the segments. Boxes within a box indicate child-parent relationships.

the space-time segments of videos are organized in a two-level hierarchy. The first level comprises the root space-time segments that may contain the whole human body. The second level comprises space-time segments that contain parts of the root.

We present an algorithm to extract hierarchical space-time segments from videos. This algorithm is unsupervised, such that it does not need any pre-trained body or body part detectors that may be constrained by strong priors of common poses present in the related training set. Fig. 3.1 shows some example video frames and extracted hierarchical segments in the UCF-Sports video dataset [65] and more examples are shown in Fig. 3.7. The representation of parts is multi-grained in that the parts are allowed to overlap: some parts are actually parts of larger parts, e.g. lower leg and whole leg. These segments are then tracked in time to produce space-time segments as shown in Fig. 3.2.

Our algorithm comprises three major steps. We first apply hierarchical segmen-

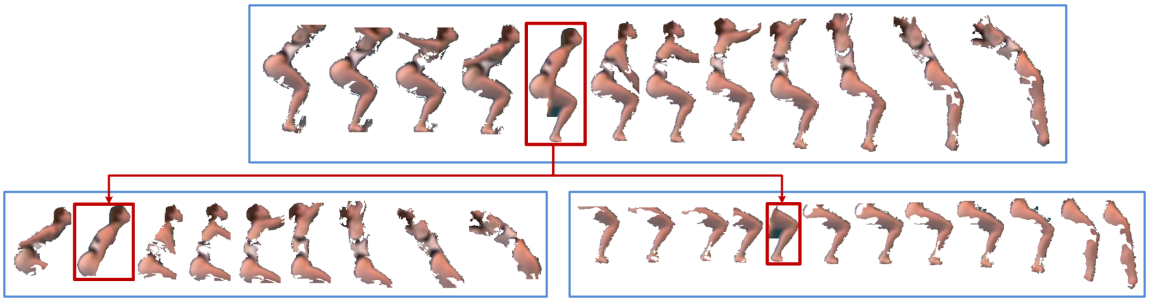

Figure 3.2: Hierarchical space-time segments extracted from a *diving* action in the UCF Sports dataset. Each blue box shows a space-time segment. Red boxes show a segment tree on a frame, and the space-time segments are produced by tracking these segments.

tation on each video frame to get a set of segment trees, each of which is considered as a candidate segment tree of the human body. In the second step, we prune the candidates by exploring several cues such as shape, motion, articulated objects' structure and global foreground color. Finally, we track each segment of the remaining segment trees in time both forward and backward. This process yields the final hierarchical space-time segments. These space-time segments are subsequently grouped into tracks according to their space-time overlap.

We then utilize these space-time segments in computing a bag-of-words representation. Bag-of-words representations have shown promising results for action recognition [50, 76]. Those representations, however, mostly lack the spatial and temporal relationships between regions of interest, whereas there are attempts to include these relationships later via higher order statistics [91, 46, 20, 63, 18]. Our hierarchical segmentation-based representation preserves hierarchical relationships naturally during extraction and, by following the temporal continuity of these space-time segments, the temporal relationships are also preserved. We show in experiments that by using both parts and root space-time segments together, better recognition is achieved. Leveraging temporal relationships among root space-time segments, we

can also localize the whole track of the action by identifying a sparse set of space-time segments.

Contributions: We make the following contributions in this chapter:

1. A new *hierarchical space-time segments* representation designed for both action recognition and localization that incorporates multi-grained representation of the parts and the whole body in a hierarchical way.

2. An algorithm to extract the proposed hierarchical space-time segments that preserves both static *and* non-static relevant space time segments as well as their hierarchical and temporal relationships. Such relationships serve for better recognition and localization.

We evaluated the proposed formulation on challenging benchmark datasets UCF-Sports: [65] and HighFive [59]. These datasets are representative of two major categories of realistic actions, namely sports and daily interactions. Using just a simple linear SVM on the *bag of hiearchical space-time segments* representation, promising action recognition performance is achieved without using human bounding box annotations. At the same time, as the results demonstrate, our proposed representation produces good action localization results.

Roadmap for this Chapter

We first describe our algorithm for hierarchical video frame segmentation in Sec. 3.1, which takes both motion and color into account and produce a set of candidate segment trees. We then prune the candidate trees of each frame, using shape, motion and color cues which are discussed in detail in Sec. 3.2. This pruning procedure removes most of the irrelevant / background segments so that we subsequently track the remaining segments in time to extract the HSTSs. This requires an efficient

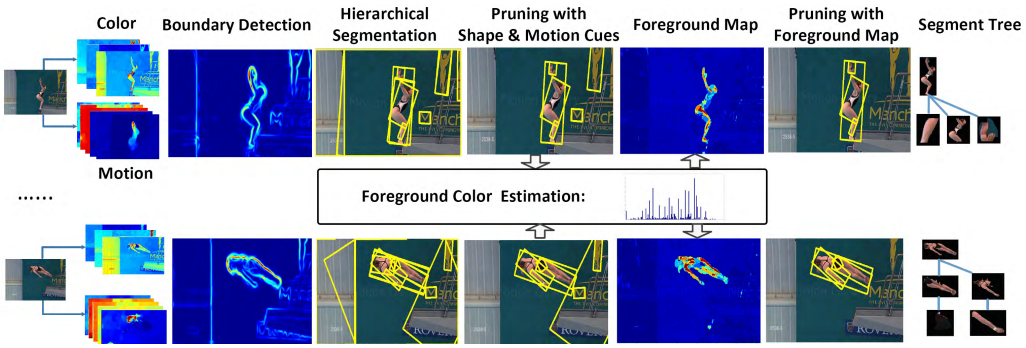

Figure 3.3: The pipeline for foreground video frame segments extraction.

tracking method, because we need to finish tracking all remaining segments of all frames in reasonable time. It also requires the tracking method to be robust, because our the tracking targets, which are mostly regions on human body, may undergo significant non-rigid deformation in time. For both ends, we propose an efficient non-rigid region tracking method in Sec. 3.3 and use it to effectively and efficiently extract HSTSs. In Sec. 3.4 we discuss our method for training Bag-of-Words model using HSTSs for action recognition and localization, and the evaluation experiments of this model is presented in Sec. 3.5.

3.1 Video Frame Hierarchical Segmentation

For human action recognition, segments in a video frame that contain motion are useful as they may belong to moving body parts. However, some static segments may belong to the static body parts, and thus may be useful for the pose information they contain. Moreover, for localizing the action performer, both static and non-static segments of the human body are needed. Based on this observation, we design our video frame segmentation method to preserve segments of the whole body and the body parts while suppressing the background. The idea is to use both color

and motion information to reduce boundaries within the background and strengthen internal motion boundaries of human body resulting from different motions of body parts.

In practice, on each video frame, we compute the boundary map by the method in [51] using three color channels and five motion channels including optical flow, unit normalized optical flow and the optical flow magnitude. The boundary map is then used to compute an Ultrametic Contour Map (UCM) by the method in [3].

The UCM represents a hierarchical segmentation of a video frame [3], in which the root is the whole video frame. We traverse this segment tree to remove redundant segments as well as segments that are too large or too small and unlikely to be a human body or body parts.

We then remove the root of the segment tree and get a set of segment trees $\mathbf{T}^t$ (t is index of the frame). Each $T_j^t \in \mathbf{T}^t$ is considered as a candidate segment tree of a human body and we denote $T_j^t = \{s_{ij}^t\}$ where each s_{ij}^t is a segment and s_{0j}^t is the root segment. Two example candidate segment trees, which remain after the subsequent pruning process, are shown in the rightmost images in Fig. 3.3.

3.2 Pruning Candidate Segment Trees

To localize the whole body of action performer, we want to extract both static and non-static relevant segments, so the pruning should preserve segments that are static but relevant. We achieve this by exploring the hierarchical relationships among the segments: we prune segment trees instead of individual segment. Specifically, a candidate segment tree is either removed altogether or kept with all its segments. In this way, we may extract the whole human body even if only a small body part has motion. We explore multiple action related cues to prune the candidate segment

trees, as described below, in the order of our pipeline as shown in Fig. 3.3.

3.2.1 Tree Pruning with Shape Cue

Background objects (*e.g.* buildings) with straight boundaries are common in man-made scenes, but human body boundaries contain fewer points of zero curvature. With this observation, for each candidate $T_j^t \in \mathbf{T}^t$, we compute the curvature at all boundary points of its root segment s_{0j}^t. If the ratio of points that have approximately zero curvature is large (> 0.6 in our system), we remove T_j^t. The curvature κ_a at a boundary point (x_a, y_a) is computed as follows:

$$\kappa_a = \left| \frac{x_a - x_{a+\delta}}{y_a - y_{a+\delta}} - \frac{x_{a-\delta} - x_a}{y_{a-\delta} - y_a} \right| \tag{3.1}$$

where $(x_{a-\delta}, y_{a-\delta})$ and $(x_{a+\delta}, y_{a+\delta})$ are two nearby points of (x_a, y_a) on the segment boundary.

3.2.2 Tree Pruning with Motion Cue

For each segment $s_{ij}^t \in T_j^t$, we compute the average motion magnitude of all pixels within s_{ij}^t. To compute the actual motion magnitude, we compute an affine transformation matrix between the current frame and the consecutive frame to approximate camera motion and calibrate the flow fields accordingly. If at least one segment has average motion magnitude higher than some threshold, T_j^t will be kept, otherwise it will be pruned. The threshold is set as the median value of the motion magnitudes of pixels that are not contained in any of $T_j^t \in \mathbf{T}^t$ since these pixels are likely to belong to the background.

3.2.3 Tree Pruning with Color Cue

We explore two general assumptions to estimate foreground maps for further pruning of the candidates. First, as an articulated object, the region of a non-static human body usually contains many internal motion boundaries resulting from different motions of body parts. Thus, the segment tree of the human body usually has a deeper structure with more nodes compared to that of the rigid objects. Second, segments of the foreground human body are more consistently present than segments caused by artificial edges and erroneous segmentation. Based on these two assumptions, we construct the foreground maps using color information as follows:

Denote $\widetilde{\mathbf{T}}^t$ as the set of remaining candidate trees after pruning with shape and motion cues, and let S represent the set of all remaining segments on all frames, i.e. $S = \{s_{ij}^t | \forall\ s_{ij}^t \in T_j^t,\ \forall\ T_j^t \in \widetilde{\mathbf{T}}^t,\ \forall\ t\}$. To avoid cluttered notation, in the following we simply denote $S = \{s_k\}$. We compute the L_∞ normalized color histogram $\mathbf{h}_k$ for every $s_k \in S$ (128 bins in our system). Then the foreground color histogram $\mathbf{h}$ is voted by all segments:

$$\mathbf{h} = \sum_{\mathbf{h}_k \in \mathbf{h}} 2^{d_k} \cdot \mathbf{h}_k \tag{3.2}$$

where d_k is the depth of segment s_k in its segment tree. For root segment s_{0j}^t we define its depth d_{0j}^t to be 1 and for a non-root segment s_{ij}^t its depth d_{ij}^t is set to the number of edges on the path from the root to it plus one. The color histogram $\mathbf{h}$ is then $L1$ normalized. As we can see from Eq. 3.2, colors of more frequently appearing segments and segments with greater depths will receive more votes.

Let $\mathcal{F}^t$ denote the foreground map of frame t. Its value at i-th pixel is set as $\mathcal{F}_i^t = \mathbf{h}(h_i^t)$, where h_i^t is the color of i-th pixel of frame t. For each segment s_{ij}^t in a candidate segment tree $T_j^t \in \widetilde{\mathbf{T}}^t$ on frame t, we compute its foreground probability

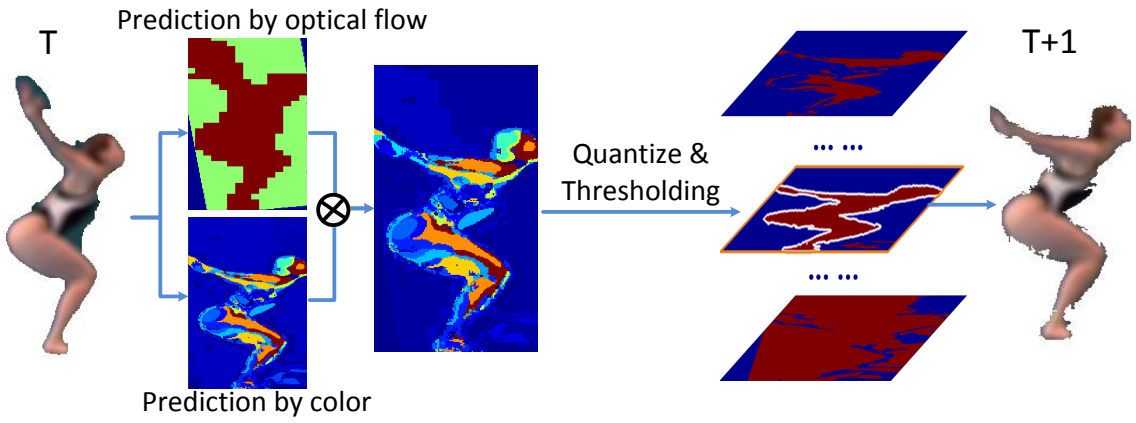

Figure 3.4: Overview of our non-rigid segment tracking method. Both color and motion information are explored to efficiently locate the target in the small search region on the next video frame. This method can deal with non-rigid deformation of the target segment, which is crucial for tracking segments of non-static human body parts.

as the average values covered by s_{ij}^t in $\mathcal{F}^t$. If all segments of T_j^t have low foreground probability, T_j^t is pruned, otherwise it will be kept. One can see in Fig. 3.3 that by using the foreground map, we can effectively prune the background segments in the example frames.

3.3 Extracting Hierarchical Space-Time Segments

After candidate segment tree pruning, we extract a set $\widehat{\mathbf{T}}^t$ that contains remaining candidate segment trees. To capture temporal dynamics of the human body or body parts, for each $T_j^t \in \widehat{\mathbf{T}}^t$ we track every segment $s_{ij}^t \in T_j^t$ to construct a space-time segment. This is a challenging task: first, we need to track segments of any shape, not simply rectangular bounding boxes; second, the segments mostly correspond to human body parts which may undergo significant non-rigid deformation over time; third, the tracking method need to be fast so that to allow efficient tracking of all the segments. Here we propose a segment tracking (Fig. 3.4) method that is

computationally efficient and can deal with non-rigid deformation of the target. We describe the details in Sec. 3.3.1. We then apply post-processing to reduce the depth of the HSTSs for robustness and group overlapping HSTSs into tracks to get temporally longer representations of the video, described in Sec. 3.3.2.

3.3.1 Non-rigid Region Tracking

Let R denote the tracked segment on the current frame. Let $a = (x_a,\ y_a)$ denote the coordinates of a point in R and let $(\Delta x_a,\ \Delta y_a)$ denote its corresponding flow vector from median filtered optical flow fields. The predicted segment at the next frame by flow is then $R' = \{(x'_a,\ y'_a)\} = \{(x_a + \Delta x_a,\ y_a + \Delta y_a)\}$. Let B' denote the bounding box of R' whose edges are parallel to horizontal and vertical axes and let $\hat{B}'$ represent the tight bounding rectangle of R' whose longer edge is parallel with R''s axis of least inertia. Let $\mathbf{h}$ represent the color histogram of the original segment being tracked. We then compute a flow prediction map $\mathcal{M}_f$ and color prediction map $\mathcal{M}_h$ over B'. Suppose a point $b' \in B'$ on the next frame has color $h_{b'}$, then we set $\mathcal{M}_h(b') = \mathbf{h}(h_{b'})$ and set $\mathcal{M}_f(b')$ as:

$$\mathcal{M}_f(b') = \begin{cases} 2 & b' \in R' \\ 1 & b' \in \hat{B}' \wedge b' \notin R' \\ 0 & otherwise \end{cases} \tag{3.3}$$

We combine the two maps, $\mathcal{M}(b') = \mathcal{M}_f(b') \cdot \mathcal{M}_h(b')$. In practice, when we compute $\mathcal{M}_f$ we use a grid over $\hat{B}'$ so that points in the same cell will be set to the maximum value in that cell. This is to reduce holes caused by noise in the optical flow field.

The map $\mathcal{M}$ is scaled and quantized to contain integer values in the range $[0, 20]$. By applying a set of thresholds δ_m of integer values from 1 to 20, we get 20 binary

maps. The size of every connected component in these binary maps is computed and the one with most similar size to R is selected as the candidate. Note that multiple thresholds are necessary. This is because the color distribution of the template may be either peaked, if the template has uniform color, or relatively flat, if the template contains many colors. Thus the range of values in $\mathcal{M}(b')$ may vary in different situations and a fixed threshold will not work well. The number of thresholds is experimentally chosen for the trade-off between performance and speed.

If the ratio of size between the selected candidate and R being tracked is within a reasonable range (we use $[0.7, 1.3]$), we set the candidate as the tracked segment, otherwise the target is considered as being lost. Since all the computations are performed locally in B' and implemented as matrix operations utilizing efficient linear algebra software packages, this tracking method is very fast. We track each segment at most 7 frames backward, where backward optical flow is used, and 7 frames forward. Thus a space-time segments has a maximum temporal length of 15 in our experiments.

3.3.2 Post-processing

Although segment tree T_j^t may have deep structures with height larger than 2, these structures may not persistently occur in other video frames due to the change in human motion and the errors in segmentation. For robustness and simplicity, we have only two levels in the resultant hierarchical structure of space-time segments: the root level is the space-time segment by tracking the root segment s_{0j}^t of T_j^t, and the part level are the space-time segments by tracking non-root segments $s_{ij}^t, \forall i \neq 0$.

Since space-time segments are constructed using segment trees of every video frame, many of them may temporally overlap and contain the same object, *e.g.*space-

time segments produced by tracking the segments that contain the same human body but are from two consecutive frames. This provides a temporally dense representation of the video. We explore this dense representation to construct longer tracks of objects despite the short temporal extent of individual space-time segment. Specifically, two root space-time segments that have significant spatial overlap on any frame are grouped into the same track. The spatial overlap is measured by the ratio of the intersection over the area of the smaller segment and we empirically set the overlap threshold to be 0.7. The part space-time segments are subsequently grouped into the same track as their roots. Note that now we have temporal relationships among root space-time segments of the same track.

For each track, we then compute bounding boxes on all frame it spans. As described before, many root space-time segments of a track may temporally overlap on some frames. On each of those frames, every overlapping root space-time segment will provide a candidate bounding box. We choose the bounding box that has the largest average foreground probability using the foreground map described in section 3.2. As we assume the foreground human body is relatively consistently present in the video, we further prune irrelevant space-time segments by removing tracks of short temporal extent. We set the temporal length threshold to be one fourth of the video length but keep the longest track if no track has length greater than the threshold.

3.4 Action Recognition and Localization

To better illustrate the effectiveness of HSTSs, we train simple Bag-of-Words model for action recognition and localization using HSTSs.

For each space-time segment, we divide its axis parallel bounding boxes using a

space-time grid, compute features within each space-time cell and concatenate the features from all cells to make the final feature vector. Here we want to mention that we do not limit the space-time segments to have the same length as the method in [76] did for dense trajectories. This is to deal with the variations of action speeds of different performers. We do not use space-time segments that are too short (length < 6) as they may not be discriminative enough. We also split long space-time segments (length > 12) into two to produce shorter ones while keeping the original one. This is to get a denser representation of the action.

We build separate codebooks for root and part space-time segments using k-means clustering. Subsequently each test video is encoded in the BoW (bag of words) representation using max pooling over the similarity values between its space-time segments and the code words, where the similarity is measured by histogram intersection. We train one-vs-all linear SVMs on the training videos' BoW representation for multiclass action classification, and the action label of a test video is given by:

$$y = arg\max_{y \in \mathcal{Y}} \begin{pmatrix} \mathbf{w}_y^r \\ \mathbf{w}_y^p \end{pmatrix} (\mathbf{x}^r \ \mathbf{x}^p) + b_y \tag{3.4}$$

where $\mathbf{x}^r$ and $\mathbf{x}^p$ are the BoW representations of root and part space-time segments of the test video respectively, $\mathbf{w}_y^r$ and $\mathbf{w}_y^p$ are entries of the trained separation hyperplane for roots and parts respectively, b_y is the bias term and $\mathcal{Y}$ is the set of action class labels under consideration.

For action localization, in a test video we find space-time segments that have positive contribution to the classification of the video and output the tracks that contain them. Specifically, given a testing video as a set of root space-time segments $S^r = \{\mathbf{s}_a^r\}$ and a set of part space-time segments $S^p = \{\mathbf{s}_b^p\}$, denote $C^r = \{\mathbf{c}_k^r\}$ and $C^p = \{\mathbf{c}_k^p\}$ as the set of code words that correspond to positive entries of $\mathbf{w}_y^r$ and $\mathbf{w}_y^p$

respectively. We compute the set U as

$$U = \left\{ \hat{\mathbf{s}}^r : \hat{\mathbf{s}}^r = \underset{\mathbf{s}_a^r \in S^r}{arg\max}\ h(\mathbf{s}_a^r,\ \mathbf{c}_k^r),\ \forall \mathbf{c}_k^r \in C^r \right\} \cup \left\{ \hat{\mathbf{s}}^p : \hat{\mathbf{s}}^p = \underset{\mathbf{s}_b^p \in S^p}{arg\max}\ h(\mathbf{s}_b^p,\ \mathbf{c}_k^p),\ \forall \mathbf{c}_k^p \in C^p \right\} \tag{3.5}$$

where function h measures the similarity of two space-time segments, for which we use histogram intersection of their feature vectors. We then output all the tracks that have at least one space-time segment in the set U as action localization results. In this way, although space-time segments in U may only cover a sparse set of frames, our algorithm is able to output denser localization results. Essentially these results benefit from the temporal relationships (*before, after*) among the root space-time segments in the same track.

3.5 Experiments

We conducted experiments on the UCF-Sports [65] and High Five [59] datasets to evaluate the proposed hierarchical space-time segments representation. These two datasets are challenging and representative of two different major types of actions: sports and daily interactions.

3.5.1 Experimental Setup

We implemented our formulation in Matlab.[1] The parameters for extracting hierarchical space-time segments are empirically chosen without extensive tuning and mostly kept the same for both datasets.

UCF-Sports Dataset: The UCF-Sports dataset [65] contains 150 videos of 10

[1]Code at http://www.cs.bu.edu/groups/ivc/software/STSegments/.

different classes of actions. We use the training/testing split of [49]. For each root space-time segment, we use a $3\times3\times3$ space-time grid and compute HoG (histogram of oriented gradients), HoF (histogram of optical flow) and MBH (histogram of motion boundary) features in each space-time cell. The number of orientation bins used is 9. For part space-time segments, we use a $2 \times 2 \times 2$ space-time grid and the other settings are the same. We build a codebook of 2000 words for root space-time segments and 4000 words for parts. This dataset contains bounding box annotations on each video frame. While the compared methods use these annotations in their training, we do not use them in ours. These annotations are only used for evaluation of action localization in our experiments.

HighFive TV-interactions: The HighFive dataset [59] contains 300 videos from TV programs. 200 of them contain 4 different classes of daily interactions, and the other 100 are labeled as negative. We follow the training/testing split of [59]. We use the same space-time grid setting as in UCF-Sports, but to fairly compare with previous results in [18], we compute only MBH features in each space-time cell. We build a codebook of 1800 words for root space-time segments and 3600 words for parts. Again, we do not use the body bounding box annotations in our training.

JHMDB contains 928 videos of 21 actions. This is a subset of the HMDB dataset [48] that comes with more detailed annotations such as foreground pixel masks, body joint positions, etc. [37]. In this chapter we only use the JHMDB dataset for evaluating the quality of space-time segments, taking advantage of its pixel level foreground annotation.

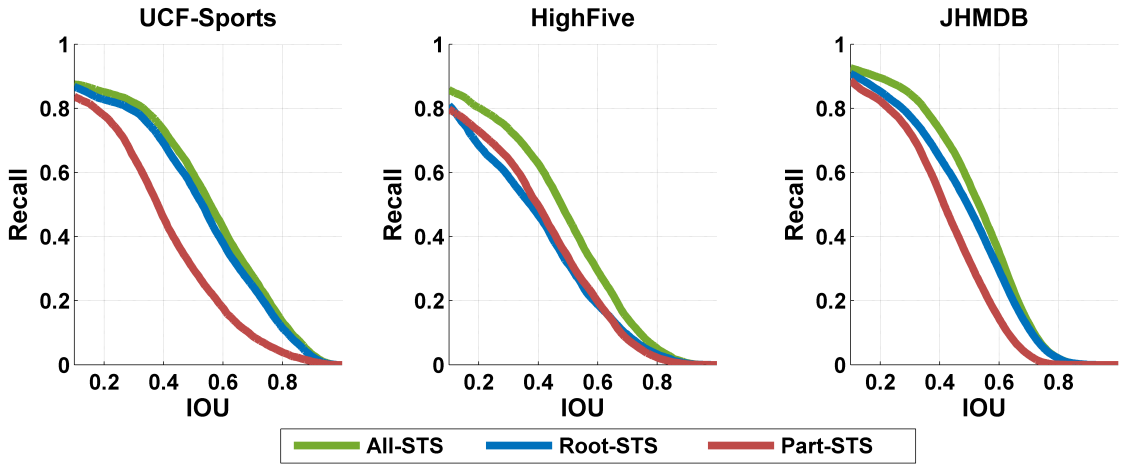

Figure 3.5: Detection rate of the ground truth foreground bounding boxes / regions by extracted space-time segments under different IOU thresholds.

3.5.2 Quality of Space-Time Segments

We first evaluate the quality of the extracted hierarchical space-time segments. Our method for extracting space-time segments can be thought of as generating proposals of action performer regions in space-time from videos. Following the protocols used for evaluating algorithms for object proposal in images, *e.g.*, as in [101], we measure the detection rate (*recall*) of the foreground bounding boxes or region masks under different thresholds of the Intersection-Over-Union (IOU). The results of this evaluation are reported in Fig. 3.5. Note that extracting space-time segments involves no training, so we evaluate over whole datasets. Also note that the ground truth annotations available in UCF-Sports and High5 are human bounding boxes, whereas the annotations for JHMDB are foreground masks.

From Fig. 3.5 we can see that, when using both root and part space-time segments, more than 80% of all foreground bounding boxes or regions are detected at IOU threshold 0.2 on all three datasets. When we increase the IOU threshold to 0.4, still around 50% of the foreground is covered. Further, we can see that the

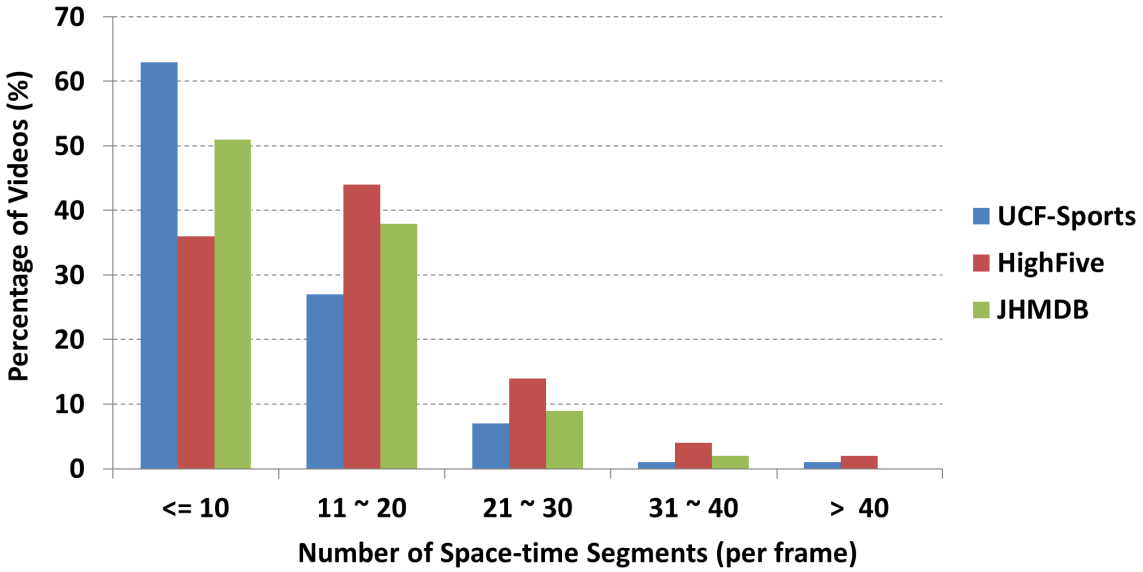

Figure 3.6: Distribution of the total number of space-time segments extracted from each video in UCF-Sports, HighFive and JHMDB. For most videos our algorithm extracts no more than 20 space-time segments per frame. Compare this with [77] which on average extracts 205 trajectories per frame.

root space-time segments are better in covering the foreground annotation, *i.e.* action performer(s), than part space-time segments. This illustrates the benefit of our algorithm, which not only extracts regions with motion (mostly part space-time segments) but also extracts relevant static regions that are contained in the root space-time segments. A combination of both the root and the part space-time segments provides a denser representation of the video, and a better covering of the foreground than using only root or part space-time segments alone.

The distributions for the total number of space-time segments extracted per video are shown in Fig. 3.6 for these three datasets. On all three datasets, for most videos, our algorithm extracts fewer than 20 space-time segments per frame. For UCF-Sports and JHMDB, in more than half of the videos, only around 10 space-time segments are extracted per frame. This shows that, while providing good coverage of the foreground (Fig. 3.5) and good action classification performance (Sec. 4.6.4), hierarchical

Method	Supervision	Accuracy
Ours (Root + Part)	label	81.7%
Ours (Part only)	label	71.3%
Raptis *et al.* [63]	label + box	79.4%
Lan *et al.* [49]	label + box	73.1%

Table 3.1: Mean per-class classification accuracies on the UCF-Sports dataset. The training/testing split follows [49]. Unlike [49, 63], we do not require bounding box annotation for training.

Method	mAP
Ours (Root + Part)	53.3 %
Ours (Part only)	46.3 %
Gaidon *et al.* [18]	55.6%
Wang *et al.* [76]	53.4%
Laptev *et al.* [50]	36.9%
Patron-Perez *et al.* [59]	32.8 %

Table 3.2: Mean average precision (mAP) on the HighFive dataset.

space-time segments provide a significantly more compact video representation than the widely used dense trajectory algorithm [77], which reports extracting 205 trajectories per frame on average on the videos of Hollywood2 [53] which are of similar resolution with UCF-Sports and HighFive (higher resolution than JHMDB).

3.5.3 Action Recognition and Localization Results

Action recognition: The action recognition results are shown in Table 3.1 and Table 3.2 for the UCF-Sports and HighFive datasets respectively.

On the UCF-Sports dataset, our method is compared with two state-of-the-art methods [63, 49]. The method in [63] learns an action model as a Markov random field over a fixed number of dense trajectory clusters. The method in [49] uses a figure-centric visual word representation in a latent SVM formulation for both action

	subset of frames				all frames			
	[74]	[75]	[49]	Ours	[74]	[75]	[49]	Ours
dive	16.4	36.5	43.4	**46.7**	22.6	37.0	-	**44.3**
golf	-	-	37.1	**51.3**	-	-	-	**50.5**
kick	-	-	36.8	**50.6**	-	-	-	**48.3**
lift	-	-	**68.8**	55.0	-	-	-	**51.4**
ride	62.2	**68.1**	21.9	29.5	63.1	**64.0**	-	30.6
run	50.2	**61.4**	20.1	34.3	48.1	**61.9**	-	33.1
skate	-	-	13.0	**40.0**	-	-	-	**38.5**
swing-b	-	-	32.7	**54.8**	-	-	-	**54.3**
swing-s	-	-	16.4	**19.3**	-	-	-	**20.6**
walk	-	-	28.3	**39.5**	-	-	-	**39.0**
Avg.	-	-	31.8	**42.1**	-	-	-	**41.0**

Table 3.3: Action localization results measured as average IOU (in %) on the UCF Sports dataset. '-' means result is not available. Note that, [74] and [75] need bounding boxes in training and their models are only for binary action detection, so their results are not directly comparable to ours.

Class	hand shake	high five	hug	kiss	**Avg.**
IOU	26.9	32.9	34.2	29.2	30.8
Recall	79.4	88.8	82.6	80.8	82.3

Table 3.4: Action localization performance measured as average IOU (in %) and recall (in %) on the High Five dataset.

localization and recognition. Both compared methods used more complex classifiers than the simple linear SVM used in ours. More importantly, both of them require expensive frame-wise human bounding box annotations, while ours does not. However, our method performs slightly better (by 2.3%) than [63] and significantly better (by 8.6%) than [49]. Although [63] achieves comparable classification performance with ours, it cannot provide meaningful action localization results. The method in [49] can output localization results, but its localization performance is significantly lower than ours (see Table 3.3), as will be discussed in detail later.

On the High Five dataset, our method is compared to four methods [50, 59, 76, 18]. The method in [18] uses non-linear SVM on a cluster tree of dense trajectories and produces state-of-the-art results. The results for [76] and [50] are produced by using SVM with the histogram intersection kernel on bag of dense trajectories and STIPs respectively. The method in [59] uses structured SVM. Despite its implicity, our method achieves comparable performance with [18] and [76] and significantly better performance than [50] and [59]. None of the compared methods perform action localization as our method does.

To assess the benefit of extracting the relevant static space-time regions that are contained in the root space-time segments, we compare with a baseline that only uses space-time segments of parts. The results show that there is a significant performance drop (10.4% on UCF-Sports and 7.0% on High Five) if space-time segments of roots are not used. This supports our hypothesis that pose information captured by root space-time segments is useful for action recognition.

Action localization: Table 3.3 and Table 3.4 show the action localization results on the UCF-Sports and HighFive datasets. Fig. 3.8 visualizes localization results on some example frames of both datasets. The localization score is computed as the

average IOU (intersection-over-union) over tested frames.

On the UCF-Sports dataset, the method of [49] can only produce localization results on a subset of frames, so we include comparisons on this subset. On this subset of frames, our method performs better than [49] on 9 out of 10 classes and the average performance is higher by 10.3%. We also provide our performances on all frames, which are similar to those on subset of frames. The methods of [74] and [75] only report action localization results on 3 classes (*running*, *diving* and *horse riding*) of UCF Sports. Our performance is higher than [74, 75] in one class (*diving*) but lower in the other two. All compared methods use expensive human bounding box annotations and their learning are much more complex than ours.

On the HighFive dataset, the IOU is measured over the frames that are annotated as containing the interactions. We achieve an IOU of 30.8%, which is still reasonably good but lower than our results on UCF Sports. We suspect this may be partly due to the low quality of human bounding box annotations used for evaluation, as most of them are too small, covering only the head area of the actors (see Fig. 3.8). To verify this, we also compute the recall, which is measured as the ratio of the area of intersection over the annotated action area. The high recall values reported in Table 3.4 confirm that the annotated action areas are mostly identified by our method. [74] and [75] have reported action localization results only on the *kiss* class, which are 18.5% and 39.5% respectively. Again, these results are not directly comparable, since our method requires much less supervision (only labels) compared to [74] and [75] which require human bounding boxes.

Figure 3.7: Extracted segments for example video frames from the UCF Sports and HighFive dataset. Each example frame is from a different video. The yellow boxes outlines the extracted segments. The inclusion of one box in another indicates the parent-children relationships

Figure 3.8: Example of our action localization results. The area covered by green mask is the ground truth from annotation, and the area covered by red mask is our action localization output. First five rows are from UCF-Sports, and last two rows are from HighFive.

3.6 Summary

In this Chapter, we propose hierarchical space-time segments for action recognition that can be utilized to effectively answer *what* and *where* an action happened in realistic videos as demonstrated in our experiments. Compared to previous methods such as STIPs and dense trajectories, this representation preserves both relevant static and non-static space-time segments as well as their hierarchical relationships, which helped us in both action recognition and localization. One direction for future work is to make the method more robust to low video quality, as it may fail to extract good space-time segments when there is significant blur or jerky camera motion.

Chapter 4

Ensemble of Space-Time Tree

Human actions, or interactions, are inherently defined by structured patterns of the human movement. As such, human actions can be modeled as spatio-temporal graphs, where the graph vertices encode movements of whole body or body parts, and the graph edges encode spatio-temporal relationships between pairs of movement elements, for instance temporal progression, *e.g.*, one movement followed by another movement, spatial composition, *e.g.*, movement of upper body coupled with movement of lower extremities, or even hierarchical relationships of elements, *e.g.*, movement of the body as a whole can be decomposed into local movements of the limbs.

A single spatio-temporal structure, however, is unlikely to be sufficient to represent a class of action in all but the simplest scenarios. First, the execution of the action may differ from subject to subject, involving different body parts or different space-time progressions of body part movements. Second, the video capture process introduces intra-class variations due to occlusions or variations in camera viewpoint. Thus, the resulting space-time and appearance variations necessitate using a *collection* of spatio-temporal structures that can best represent the action at large.

In this Chapter, we propose a formulation that discovers a collection of hierarchical space-time trees from video training data, and then learns a discrimina-

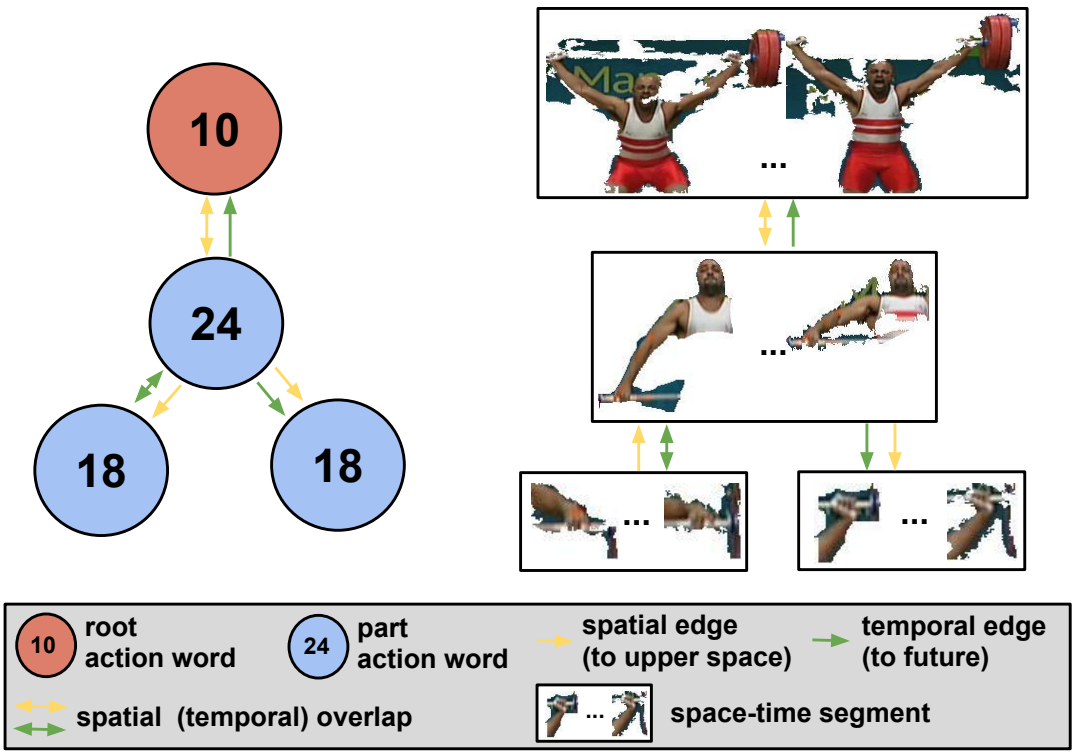

Figure 4.1: One example tree structure discovered by our approach for the *lifting* action and its best match in a testing video. In the tree, one node (red) indexes to a root action word and is matched to an STS of the upward movement of the whole body; three nodes (blue) index to part action words and are matched to STSs of the upward movement of the upper-body and two temporally consecutive movements of the left arm and lower left arm respectively.

tive action model that builds on these discovered trees to classify actions in videos. Fig. 4.1 illustrates one simple discovered tree and its best match in a testing video and Fig. 4.5 shows more examples. Fig. 4.3 contrasts our method, *i.e.* ensemble of space-time trees, with major previous approaches: our method captures the space-time structures of human actions that are discovered from the training data and, for supervision, only requires action labels of the training videos.

Chapter 3 describe the new video representation hierarchical space-time segments which captures both static and non-static foreground space-time sub-volumes of the

videos. We explore the hierarchical, spatial and temporal relationships among the space-time segments (STSs): this transforms a video into a graph. We discover a compact set of frequent and discriminative tree structures from graphs of training videos and learn discriminative weights for the tree nodes and edges. Finally, we construct action classifiers given the detection responses of these trees.

We use trees instead of graphs for multiple reasons: first, any graph can be approximated by a set of spanning trees; second, inference with trees is both efficient and exact; third, trees provide a compact representation as it is easy to account for multiple structures using a single tree by allowing partial matching during inference. Partial matching of trees during inference allows us to effectively deal with variations in action performance and segmentation errors.

We note that, as the size of the tree structures increases the trees get more specific, thereby capturing more structural information that can provide greater discriminative power in classification. On the other end of the spectrum, smaller structures, particularly singletons and pairs, tend to appear more frequently in their exact forms than larger tree structures. Thus, smaller structures can be helpful when the action is simple but the occlusions or video segmentation errors are significant. Therefore, for robust action classification, the best strategy is to exploit both the large and small structures in a unified model.

Contributions: We make the following contributions in this chapter:

- We propose an approach that enables discovery of high-level tree structures that capture space, time and hierarchical relationships among the action words.
- We propose a discriminative action model that utilizes both small structures (*i.e.*, words and pairs) and richer, tree structures. This unified formulation achieves state-of-the-art performance in recognizing and spatially localizing

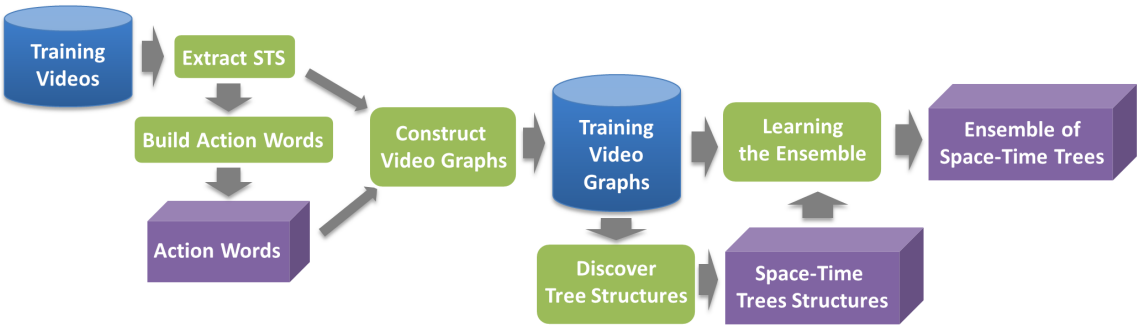

Figure 4.2: Overview of our approach for learning an ensemble of space-time trees.

human actions and interactions in realistic benchmark video datasets.

- We show generalization of the learned trees by cross-dataset validation: we achieve promising action recognition results on the Hollywood3D dataset using trees learned on the HighFive dataset, and action recognition and localization on the JHMDB dataset using trees learned on the UCF-Sports dataset.

Fig. 4.2 shows an overview of our approach for learning the ensemble of space-time trees. Given training videos, we first extract space-time segments from each video. Subsequently we build an action word vocabulary and construct video graphs from the training videos. From the training video graphs, we discover tree structures, and finally learn the ensemble of space-time trees. Given the learned ensemble of space-time trees, efficient inference can be done for recognizing the human actions in testing videos.

Roadmap for this chapter

We describe the core elements of our formulation: the space-time graph model in Sec. 4.1, inference with our model in Sec. 4.2, and how to discover various space-time graph structures using our model in Sec. 4.3. Action word vocabulary construction, based on STSs, and ensemble learning are then described in Sec. 4.4 and Sec. 4.5, respectively. The results of experimental evaluation are reported in Sec. 4.6.

	Space-time Structural Information	Number of Structures	Topology of the Structures	Supervision
Bag-of-Words *e.g.* Kantorov and Laptev 2014, Wang and Schmid 2013	Discarded	-	-	Action class label
Space-Time Pyramid *e.g.* Sadanand and Corso 2012 Oneata *et al.* 2013	Weakly captured	-	-	Action class label
Previous Structural Model *e.g.* Wang and Mori 2008, Raptis and Sigal 2013	Captured	Predefined, often one	Predefined	Action class label & human bounding boxes
Ensemble of Space-Time Trees (Ours)	Better captured	discovered from training data	discovered from training data	Action class label

Figure 4.3: A comparison of approaches for automatic human action recognition in videos. From the top row to the bottom row in the table, the approach(es) better capture and explore the space-time structural information of human actions.

4.1 Model Formulation

To capture both the appearance of local space-time segments (STSs) (Chapter 3) and the hierarchical, spatial and temporal structures among them, we transform a video into a graph where the nodes are STSs and the edges are labeled with the hierarchical, spatial and temporal relationships among the STSs. The graph nodes are subsequently given labels that are indices to a compact but discriminative action word vocabulary. Thus, in training, the training video set is converted to a set of labeled graphs, from which we discover a collection of discriminative tree structures for each action.

Formally, a video is represented as a graph $G = \{V,\ A^t,\ A^s,\ A^h,\ F\}$. V is the set of vertices that are the STSs. A^t, A^s and A^h are the time, space and hierarchical adjacency matrices containing edge labels (details in Sec. 4.3). The rows of matrix $F = [f_1, f_2, ..., f_{|V|}] \in \mathbb{R}^{|V| \times d}$ are visual features (*e.g.* HoG features) extracted from the STSs. For each action class a, a collection of trees is then used in constructing

an ensemble classifier:

$$M_a(G, \mathcal{T}) = \mathbf{w}^T \cdot \Phi(G, \mathcal{T}) = \sum_{m \in \{1,...,|\mathcal{T}|\}} w_m \phi_m(G, \mathcal{T}_m), \tag{4.1}$$

where G denotes a test input video, $\mathcal{T}$ is the set of learned tree structures for class a and $\mathcal{T}_m$ is one of such trees in this set, and $\mathbf{w} = \{w_m; m \in \{1, ..., |\mathcal{T}|\}\}$ is the learned weight vector. Each ϕ_m is a scoring function that measures compatibility (or degree of presence) of $\mathcal{T}_m$ in video G. [1] In the multi-class classification setting, the predicted action class a^* of G is computed by $a^* = \arg\max_a \ M_a(G, \mathcal{T})$.

We formalize a tree as $\mathcal{T}_m = \{N, \ E^t, \ E^s, \ E^h, \ \boldsymbol{\beta}\}$ where N, $\{E^t, E^s, E^h\}$ are the nodes and adjacency matrices respectively. $\boldsymbol{\beta}$ are discriminative weights associated with the nodes and edges. Each node $n_i \in N$ is an index into a learned discriminative action word vocabulary $\mathcal{W}_a$ for class a (described in Sec. 4.4); each edge E^k_{ij} ($k \in \{t, s, h\}$) is associated with a corresponding temporal, spatial or hierarchical relationship between nodes i and j, similar to the relations defined for A^k in graph G. The matching score of a tree to a graph is computed as follows:

$$\phi_m(G, \ \mathcal{T}_m) = \psi\left(\{ \ \boldsymbol{\beta} \cdot \varphi(G, \mathcal{T}_m, \mathbf{z}) \mid \mathbf{z} \in Z(G, \mathcal{T}_m)\}\right), \tag{4.2}$$

where $\mathbf{z}$ is latent variable that represents a match of a tree $\mathcal{T}_m$ to the video G: $\mathbf{z}$ is realized as $\mathbf{z} = (z_1, ..., z_{|N|})$ where z_i is the index of the vertex in G that is matched to the i-th node in $\mathcal{T}_m$. ψ is a pooling function over the matching scores of the set of all possible (partial) matches $Z(G, \mathcal{T}_m)$. The matching score of a specific match $\mathbf{z}$

[1] To avoid notation clutter, we omit the action class label a for $\mathcal{T}$, $\mathbf{w}$, Φ, ϕ and φ.

to $\mathcal{T}_m$ is:

$$\boldsymbol{\beta} \cdot \varphi(G, \mathcal{T}_m, \mathbf{z}) = \sum_{n_i \in N} \beta_i \; p_n(z_i, \; n_i) + \sum_{k \in \{t,s,h\}} \sum_{\substack{E^k_{ij} \in E^k \\ E^k_{ij} \neq 0}} \beta^k_{ij} \; p_k(A^k_{z_i z_j}, \; E^k_{ij}). \tag{4.3}$$

where β_i and β^k_{ij} ($k \in \{t, s, h\}$) are the tree node weights and edge weights respectively. The function p_n scores compatibility of the tree nodes with graph vertices; p_t, p_s and p_h score compatibility of the temporal, spatial and hierarchical graph edges with tree edges. Partial matching is possible by adding a *null* vertex $v_\varnothing$ to V as the 0-th vertex and also adding a 0-th row and column of zeros to A^s, A^t and A^h. Any node in $\mathcal{T}_m$ not matched to a vertex in G is assigned to match the 0-th vertex.

Our focus is on *discovering the tree structures* $\mathcal{T}$. These structures, their parameters $\boldsymbol{\beta}$, as well as parameters of the ensemble $\mathbf{w}$ are learned directly from the data. Given a tree structure, parameters can be learned in a variety of ways, *e.g.*, using latent SVM [86]. However, discovering the tree structures themselves is the key challenge as: (1) the space of tree structures is exponential in the number of tree nodes and types of relationships allowed among the tree nodes; (2) partial presence of the trees needs to be considered; (3) without annotation of body parts, the tree nodes themselves are to be discovered. Also note that our model unifies small structures, *e.g.* singletons in the extreme case, with rich tree structures: when the trees are singletons, the model essentially reduces to the BoW model.

4.2 Inference

Given a set of discovered tree structures $\mathcal{T}$, learned parameter vectors $\mathbf{w}, \boldsymbol{\beta}$ and a test video represented by a graph G, the score $S_a(G, \mathcal{T})$ of G containing an action a can be computed as a weighted sum of independently matched tree scores $\phi_m(G, \mathcal{T}_m)$. If we use average pooling in Eq. 4.2, then

$$\phi_m(G, \mathcal{T}_m) = \frac{\sum_{\mathbf{z} \in Z(G, \mathcal{T}_m)} \boldsymbol{\beta} \cdot \varphi(G, \mathcal{T}_m, \mathbf{z})}{|Z(G, \mathcal{T}_m)|}, \tag{4.4}$$

which requires looping through the whole possible set of latent values $Z(G, \mathcal{T}_m)$, which is expensive for trees with three or more nodes. Max pooling is more appropriate for larger trees:

$$\phi_m(G, \mathcal{T}_m) = \max_{\mathbf{z} \in Z(G, \mathcal{T}_m)} \boldsymbol{\beta} \cdot \varphi(G, \mathcal{T}_m, \mathbf{z}). \tag{4.5}$$

This can be efficiently computed by dynamic programming (DP), which we describe next.

Recall that every tree node n_i is an index into $\mathcal{W}_a$ and every z_i is an index into video graph vertices that are associated with features, we define the potentials in Eq. 4.3 as:

$$p_n(z_i,\ n_i) = \begin{cases} e^{c_{n_i}(f_{z_i})}, & c_{n_i}(f_{z_i}) \geq \delta \\ 0, & otherwise \end{cases} \tag{4.6}$$

$$p_k(A^k_{z_i z_j},\ E^k_{ij}) = \begin{cases} 1, & A^k_{z_i z_j} = E^k_{ij} \\ 0, & otherwise \end{cases} \tag{4.7}$$

where $k \in \{t, s, h\}$. The set of classifiers $c_{n_i}(f_{z_i}) \in \mathbb{R}$ score the mapping of the z_i-th

Figure 4.4: Tree structures of human actions are discovered from training video graphs via three steps: tree mining, tree clustering and tree ranking.

graph vertex (associated with feature f_{z_i}) to the action word which tree node n_i indexes to. The reader is referred to [16] for the details of DP on a tree, but note when matching n_i, the set of possible choices for graph vertices can be pruned for efficiency to those for which $c_{n_i}(f_{z_i}) \geq \delta$; further, nodes that cannot be matched are assigned to $v_\varnothing$ to allow partial tree matches.

The DP procedure may match a graph vertex (STS) to multiple tree nodes if those tree nodes index to the same action word. It is possible to alleviate this, *e.g.*, as in [63], but this would in general introduce high-order potentials requiring more expensive or approximate inference. We find that assigning multiple nodes is not problematic and can be beneficial in practice to account for video segmentation errors.

4.3 Discovering Structures

In Sec. 4.1 we describe how a video is represented as a graph of STSs. STSs are space-time sub-volumes of the video and can be produced by video segmentation methods. Given a video, we extract two types of STSs that can include both static and non-static body parts: root STSs often cover the whole human body and part STSs often cover the body parts. Note no person detector nor human bounding box annotations are required for this procedure. Chapter 3 describes the details of this

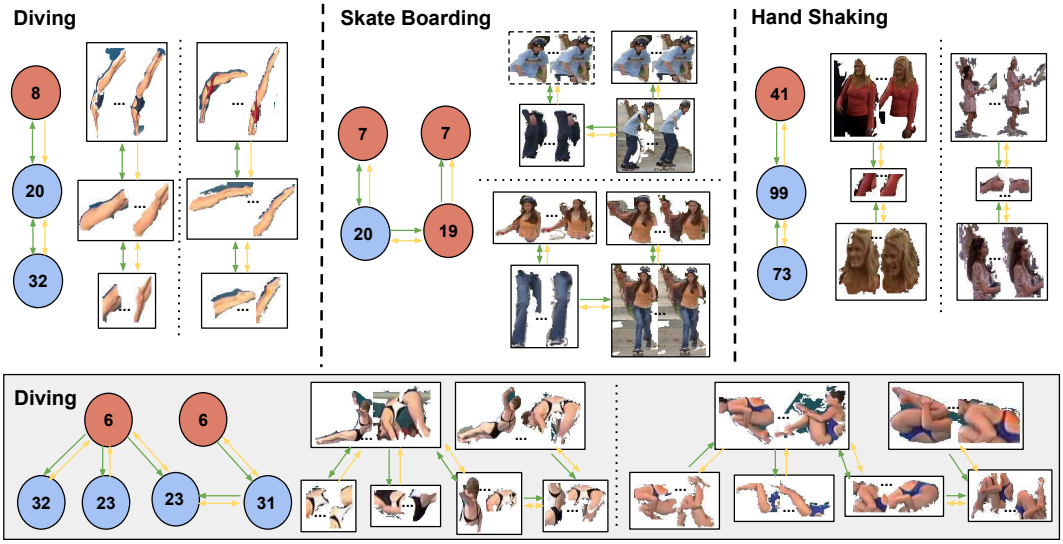

Figure 4.5: Examples of discovered trees. For each tree, we show inference results on two testing videos. See the legend of Fig 4.1 for the meaning of the figure components. The space-time segment enclosed by dotted box is matched to multiple tree nodes.

procedure.

We use a small discrete set for edge labels: $A^t \in \{0, \leftarrow, \rightarrow, \bowtie\}^{|V|\times|V|}$, $\leftarrow$, $\rightarrow$ and $\bowtie$ denote *after*, *before* and *temporal overlap*; similarly $A^s \in \{0, \uparrow, \downarrow, \bowtie\}^{|V|\times|V|}$, where $\uparrow$, $\downarrow$ and $\bowtie$ denote *above*, *below* and *spatial overlap*; $A^h \in \{0, r \rightarrow r, r \rightarrow p, p \rightarrow r, p \rightarrow p\}^{|V|\times|V|}$ denotes heirarchical root (r) - part (p) relationships. Using coarse discrete labels helps make the representation more robust to variations in action execution and STS segmentation errors.

Specifically, for a pair of the i-th and j-th STS in a video: (1) A^h_{ij} is set according to the part / root identity of i and j; (2) A^t_{ij} and A^s_{ij} are set as:

$$A^t_{ij} = \begin{cases} \bowtie, & |t_i - t_j| \leq \delta_t \\ \leftarrow, & t_i - t_j > \delta_t \\ \rightarrow, & t_j - t_i > \delta_t \end{cases} \tag{4.8}$$

$$A_{ij}^{s} = \begin{cases} \bowtie, & |y_i - y_j| \leq \delta_s \\ \uparrow, & y_j - y_i > \delta_s \\ \downarrow, & y_i - y_j > \delta_s \end{cases} \tag{4.9}$$

where t_i and t_j are the starting frame indices of i and j, y_i and y_j are the mean vertical center positions of i and j over the frames in which i and j co-exist; δ_t and δ_s are constants fixed to 3 (frames) and 20 (pixels) in all our experiments. If the temporal spans of i and j are disjoint, we let $A_{ij}^{s} = 0$.

Instead of constructing a complete graph, which can lead to high computational cost in the subsequent tree mining procedure, we construct a sparse graph. We group the STSs into tracks (Sec. 3.3), and only add edges for: 1) each pair of root STSs; 2) each pair of root STS and part STS in the same track; 3) each pair of part STSs in the same track that have significant temporal overlap.

4.3.1 Discovering Tree Structures

We want to find frequent and discriminative tree structures for action classification (Fig. 4.5). Enumerating the set of all possible trees is infeasible due to its exponential size. Furthermore, in practice, it is hard to find exact presence of complex trees due to noise in the STS extraction and variation of action execution, so we must account for partial matches when counting frequencies. We propose a three step approach (Fig. 4.4): first, we mine a large and redundant set of trees using a tree mining technique that only counts exact presence; second, we discover a compact set of trees by clustering the mined trees using a form of cosine distance on their discriminative parameter vectors $\boldsymbol{\beta}$ and pick one representative tree prototype per cluster; finally, we rank the prototype trees using a metric *activation entropy* (described in the

following) and pick the top ranked ones.

4.3.1.1 Tree Mining

We use the subgraph mining algorithm GASTON [1] to mine frequent trees with at most six vertices. The number of mined trees is controlled by the minimum support threshold parameter for each class so that $5 \times 10^3 \sim 1 \times 10^4$ trees are mined per action class. For each action class, we mine trees from both positive videos and negative videos and remove the trees that appear in both.

We then train discriminative parameters $\boldsymbol{\beta}$ for each mined tree. Given a mined tree $\mathcal{T}_m$, we initialize $\boldsymbol{\beta}$ by setting all node weights to 1 and edge weights to 0.5. We find the best matches of the tree in the graphs constructed from training videos, using the inference procedure in Sec. 4.2. $\boldsymbol{\beta}$ is then refined by training a linear SVM on the set of best matches. This procedure is similar to performing one iteration of latent SVM, except much faster.

4.3.1.2 Tree Clustering

The trees we mine using GASTON tend to be highly redundant, especially if one considers partial matching. To avoid the redundancy and discover a set of trees that effectively cover intra-class variations, we cluster the mined trees and select the best tree from each cluster. To compute similarity between two trees, $\mathcal{T}_m$ and $\hat{\mathcal{T}}_m$, we utilize the inference procedure in Sec. 4.2 by treating one of the trees (*e.g.*, $\hat{\mathcal{T}}_m$) as a weighted graph with weights on the nodes and edges corresponding to learned

parameters $\hat{\boldsymbol{\beta}}$. The potentials in $\phi_m(\mathcal{T}_{\hat{m}}, \mathcal{T}_m)$ for Eq. 4.3 are then altered as follows:

$$p_n(z_i,\ n_i) = \begin{cases} \hat{\beta}_{z_i}, & \hat{n}_{z_i} = n_i \\ 0, & \hat{n}_{z_i} \neq n_i \ or \ z_i = v_\varnothing \end{cases} \tag{4.10}$$

$$p_k(\hat{E}_{z_i z_j},\ E_{ij}) = \begin{cases} \hat{\beta}^k_{z_i z_j}, & \hat{E}^k_{z_i z_j} = E^k_{ij} \\ 0, & otherwise \end{cases} \tag{4.11}$$

where $\hat{n}_{z_i}$ is the z_i-th tree node of $\mathcal{T}_{\hat{m}}$ and $k \in \{t,\ s,\ h\}$. Note that this procedure will give us essentially the *edit* cosine distance (cosine distance since when structures are the same, $\phi_t(\mathcal{T}_{\hat{m}}, \mathcal{T}_m) = \boldsymbol{\beta} \cdot \hat{\boldsymbol{\beta}}$). This similarity measure between trees takes into account both similarity in structure and similarity in discriminative parameters $\boldsymbol{\beta}$. After computing the similarities between each pair of trees, we run affinity propagation [17] to cluster the trees and pick the tree that has the highest cross validation accuracy in each cluster as the non-redundant prototype.

4.3.1.3 Tree Ranking

The above procedure still tends to find a relatively large number of trees ($\approx$ 150 per class or more). Our hypothesis, which we validate in experiments, is that only a subset is needed for classification. To pick the best candidates we need some measure of quality among the candidates. We find cross-validation to be unstable here, and instead propose ranking based on entropy. We compute an *activation entropy* for each selected tree $\mathcal{T}_m$:

$$Entropy(\mathcal{T}_m, \mathcal{G}) = -\sum_c P_c(\mathcal{T}_m, \mathcal{G})\ log(P_c(\mathcal{T}_m, \mathcal{G})), \tag{4.12}$$

where $\mathcal{G}$ is the training set and $\mathcal{L}(G)$ is class label of G. $P_c(\mathcal{T}_m, \mathcal{G})$ is computed as:

$$P_c(\mathcal{T}_m, \mathcal{G}) = \frac{|\{G \mid \forall\ G \in \mathcal{G},\ \phi_m(G, \mathcal{T}_m) \geq 0,\ \mathcal{L}(G) = c\}|}{|\{G \mid \forall\ G \in \mathcal{G},\ \phi_m(G, \mathcal{T}_m) \geq 0\}|}, \tag{4.13}$$

The numerator encodes the number of videos in class c that are classified as positive using $\mathcal{T}_m$ and denominator the number of videos classified as positive over the training set. Intuitively, trees with smaller activation entropies have more consistent patterns in classification errors, and are more likely to be stable structures in human actions.

Despite the large set of trees that we mine, in the experiments we illustrate that by using only a compact set of trees attained through the above process (20 trees per action in UCF-Sports and Hollywood3D, and 50 per action in HighFive) promising recognition performance is achieved.

4.3.2 Discovering Pairwise Structures

When occlusion or video segmentation errors are significant, it is easier to find exact matches for smaller structures than larger ones. Pairs in particular are shown to be effective for classification *e.g.* as in [54]. Our tree discovering approach described in the previous section can find useful pairwise structures, but is by no means exhaustive. Meanwhile, our formulation of a video as a graph enables a simple exhaustive search for a set of compact and yet discriminative pairwise structures.

Specifically, denote $\mathcal{P}$ as the space of pairs for action class a, then $|\mathcal{P}| = (|\mathcal{W}_a| - 1)|\mathcal{W}_a||\mathcal{E}|/2$, where $\mathcal{E}$ denotes the space of edge labels and in our case $|\mathcal{E}| = 3 \times 3 \times 4 = 36$. In our experiments $|\mathcal{W}_a|$ is $50 \sim 150$, so $|\mathcal{P}| \leq 4 \times 10^5$. We construct a mean histogram $H \in \mathbb{R}^{|\mathcal{P}|}$ over the training graphs of all *positive* training videos. The p-th bin, $H_p = \frac{1}{n}\sum_{j=1}^{n} \kappa_p^j / (\sum_{i \in \{1,\ldots,|\mathcal{P}|\}} \kappa_i^j)$, where κ_p^j denotes appearance counts of the p-th pair in the video graph of the jth positive training video, and n is the number

Figure 4.6: Representative root and part action words for the *hand shake*, *hug*, *kick*, and *swing side* actions. For each action word, we show five space-time segments (illustrated by one temporal slice) that are in the cluster corresponding to that action word.

of positive training videos. Recall that we are after frequent and discriminative structures. Since our action words are discriminatively trained, our main criterion for selecting a pair is the average frequency H_p. As allowing partial matching for pairs may reduce to matching action words, we require exact matching in inference and set the node weights $\beta_1 = \beta_2 = 1/2$, the edge weights $\boldsymbol{\beta}^k_{12} = 0$ and $p_k(A^k_{z_1 z_2},\ E^k_{12}) = 1$ for $k \in \{t, s, h\}$.

4.4 Building the Action Word Vocabulary

In BoW approaches, large vocabularies with several thousands of words are typically used. We argue that if the words are discriminative enough, a small vocabulary can perform at least equally well as larger ones. Here we propose a method to construct a compact and yet discriminative action word vocabulary. Furthermore, such small vocabularies help greatly reduce the complexity of tree discovery, as the space of trees of length m is exponential in the size of this vocabulary, *i.e.*, $\mathcal{O}(|\mathcal{W}|^m)$. Fig. 4.6 shows some representative action words. We see that the STS clusters for these

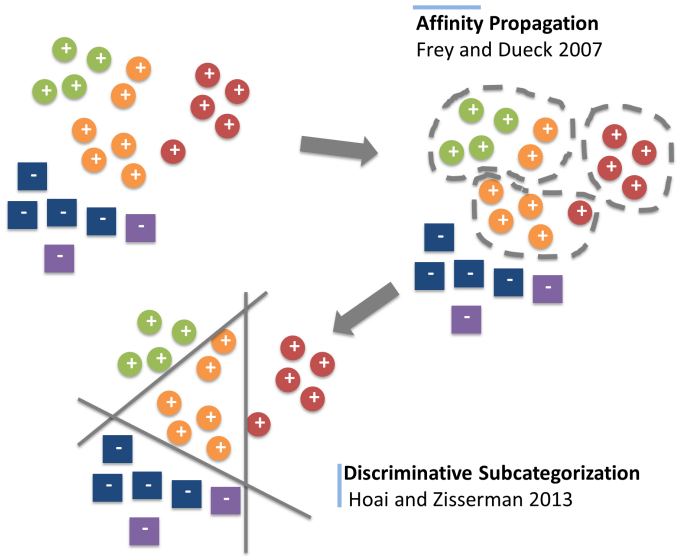

Figure 4.7: Building the action word vocabulary. Initial clusters are constructed via affinity propagation [17] on the space-time segments from the training videos of an action. This provides initialization of the samples' cluster memberships in the subsequent discriminative clustering of both positive and negative samples (*i.e.*, space-time segments from training videos of other actions) by the discriminative sub-categorization algorithm [30].

action words tend to be coherent and semantically meaningful.

Specifically, we learn the set of action words by discriminative clustering of the STSs. Recall there are two types of STSs extracted: root STSs and part STSs. Clustering is done separately for roots and parts; as a result, we get clusters $\mathcal{W}_i^r$ and $\mathcal{W}_i^p$ for the roots and the parts respectively and $\mathcal{W}_i = \mathcal{W}_i^r \cup \mathcal{W}_i^p$. We use discriminative sub-categorization (DSC) [30] to perform the clustering in each action class i, where the negative samples are the STSs extracted from videos of other action categories. DSC treats the cluster memberships of samples as latent variables and jointly learns one linear classifier per cluster using latent SVM. As we believe the vocabulary should be data-driven and adapt to the complexity of the dataset and action class, we perform an initial clustering using the affinity propagation algorithm [17] to decide the number of clusters and the initial cluster membership for each

sample. This procedure is illustrated in Fig. 4.7.

4.5 Learning the Ensemble

The main focus of our work is on discovery of larger space-time tree structures (Sec. 4.3.1). We also consider pairwise structures (Sec. 4.3.2) and trivial structures of action words (Sec. 4.4) as special cases that can be discovered through exhaustive search or clustering. We observe that larger structures (*i.e.*, trees) tend to be more discriminative than smaller ones (*i.e.*, action words and pairs), but appear in their exact form less frequently (thus, inexact matching is crucial). In practice, we have found that a combination of simpler and more complex structures tends to offer more robustness. Given the set of trees, and their responses on the training set obtained by inference on the training set of videos, we train parameters $\mathbf{w}$ using linear SVM jointly (using multi-class SVM [10]). When combining words, pairs and trees, we experiment with both early fusion and late fusion. In early fusion, the inference responses of different types of structures are concatenated and the ensemble weights are learned using SVM. In late fusion, one SVM is trained per structure type and we then train a separate SVM to combine their outputs.

4.6 Experiments

In this section, we report experimental evaluation of our formulation. We analyze the contributions of various components within our proposed formulation for improved action recognition and localization, along with comparisons with competing state-of-the-art methods. We also evaluate dataset model transfer, where tree structures and action vocabularies learned on one dataset are used in recognition and localization

of similar actions in a completely different dataset.

4.6.1 Datasets

We test our methods on four benchmark datasets: UCF-Sports [65], HighFive [59], Hollywood3D [26] and JHMDB [37]. These datasets encompass two major types of realistic videos: amateur videos that mainly depict sports and daily activities (UCF-Sports and JHMDB), and professional videos such as TV programs (HighFive) and movies (Hollywood3D).

Bounding box annotations are available for the action performers in UCF-Sports and HighFive. Image region masks are available for action performers in JHMDB. However, please note that we do not use these annotations in training our approach. We only use the bounding box and region annotations in evaluating localization performance in testing. In training, only action class labels of the training videos are used.

UCF-Sports contains 150 videos of 10 different sports actions. We use the train/test split provided by [49], in which 47 videos are used for testing and the rest for training.

HighFive contains 300 videos from TV programs. 200 of the videos contain four different interactions, including *Kiss*, *Hug*, *Hand Shake* and *High Five*, and the other 100 videos are labeled as not containing any of these four interactions. We use the train/test split that is provided with this dataset.

Hollywood3D contains 666 3D movie clips compiled from 14 films. For each video clip, the right view, the left view and the depth images are available. 588 of these video clips cover 13 actions and interactions, such as *Kick*, *Punch* and *Hug*. The other 78 videos are labeled as *NoAction*. We follow the train/test split provided with the dataset. It is important to note that we use Hollywood3D as a monocular

video dataset, without use of stereo video or depth: in our experiments we only use the left view of each clip.

JHMDB contains 928 videos of 21 actions. This is a subset of the HMDB dataset [48] that comes with more detailed annotations such as foreground pixel masks, body joint positions, etc. [37]. Among these annotations, we use the foreground mask for pixel-level evaluation of action localization performance. We use the JHMDB dataset for cross-dataset validation of the learned trees.

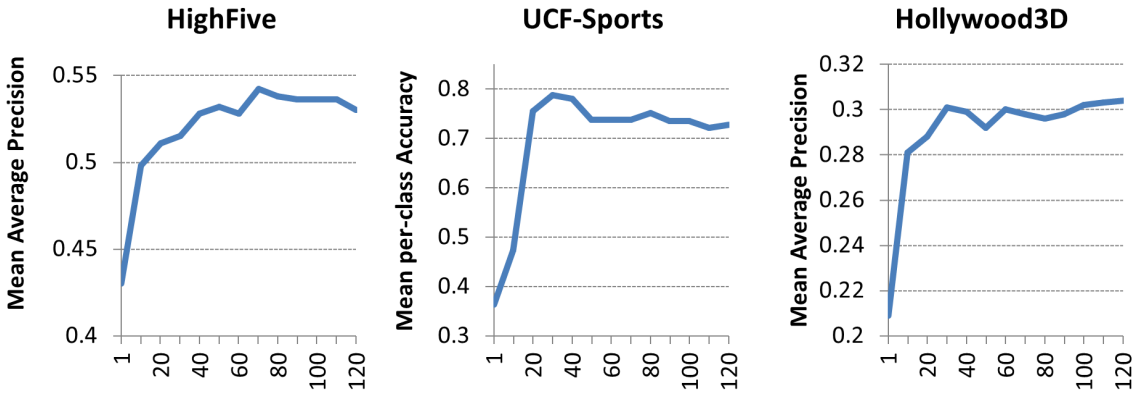

Figure 4.8: Action recognition performance vs. the number of trees. The performance improves quickly when more than one tree is used, but the performance gains tend to level off after approximately the first 20 trees.

4.6.2 Experiment Setup

From each space-time segment, three kinds of features are extracted: Histogram of Oriented Gradients (HOG), Histogram of Optical Flow (HOF), and Motion Boundary Histogram (MBH). For fair comparison with previous methods, for the HighFive dataset only MBH features are extracted from the space-time segments. We empirically found that max pooling works better for action words and trees, and average pooling is better for pairs. Thus, when combining them (Table 4.4), we use max pooling for words and trees and average pooling for pairs. We learn the ensemble

parameters by using the LIBLINEAR library [14].

4.6.3 Quality of Tree Discovery

Our goal is to discover a compact set of frequent and discriminative structures that can be used to attain good action recognition performance. Therefore, one critical question is how many structures should be selected from our discovered and ranked list of trees (ranked by $Entropy(\mathcal{T}_m, \mathcal{G})$) and pairs (ranked by H_p). Fig. 4.8 illustrates the performance as a function of the number of selected trees. We note that with a single tree, already reasonably good performance is achieved. When adding more trees, the performance improves quickly: this indicates the need for an ensemble and also validates our ranking method. After the first ~ 20 trees, the performance gains are smaller. For pairs, the trend is similar but significantly more pairs are needed than trees to achieve the same level of performance. Based on these observations, we choose to use 50 trees and 100 pairs per class for HighFive, 20 trees and 40 pairs per class for UCF-Sports, 20 trees and 50 pairs for Hollywood3D.

4.6.4 Action Recognition

Comparison of our full model vs. state-of-the-art methods is given in Tables 4.1, 4.2 and 4.3. Since we use the same space-time segments and low-level features as the Bag-of-Words model in Chapter 3, the gains of the hierarchical, spatial and temporal structures we discover are best illustrated in the comparison with their bag-of-words approach: we outperform by more than 8% on both HighFive and UCF-Sports.

We also compare with methods that use different video features on HighFive and UCF-Sports. In contrast to our simple linear model, the method in [19] and the method in [83] rely on complex graph kernels. The methods in [72], [63], [49] model

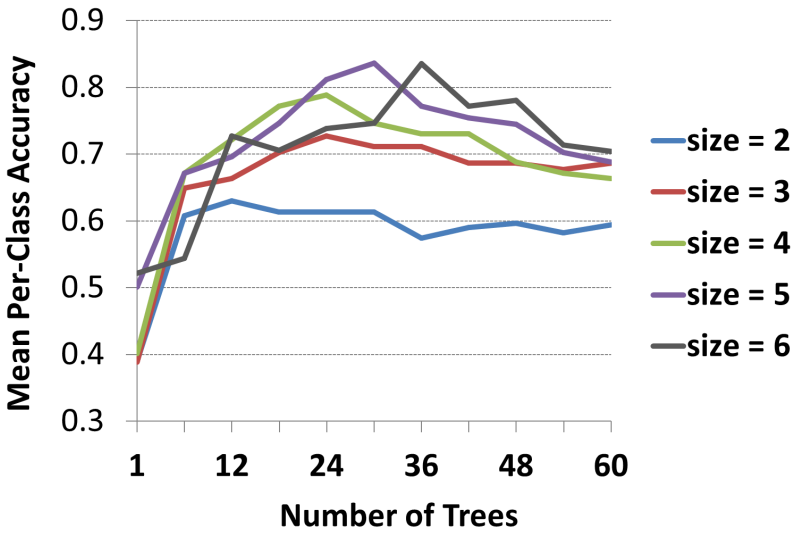

Figure 4.9: Action classification performance with varying numbers of trees for each evaluated tree size on the UCF-Sports dataset. Larger trees tend to outperform smaller ones.

human actions with space-time structures, but the number of nodes in the structures and their inter-connections are pre-determined; moreover, in training they require human bounding box annotations on video frames. Note that our method automatically discovers the structures and only needs action labels for the training videos. Nonetheless, our approach consistently achieves better performance, improving on the state-of-the-art. On HighFive our result is only slightly lower than the recent method reported by [78], which requires extracting much denser features (Sec. 3.5.2) as well as sophisticated motion stabilization and feature encoding.

On Hollywood3D, using only trees, we already achieve promising performance. When combining trees with words and pairs, the performance is further improved and is better than the methods in [26] and [36]. Although our method is outperformed by [27], we emphasize here that all the methods we compared with in Table 4.3 utilize multi-view and depth information, while we only use a single view for each video.

Method	mAP
Ensemble Model (*Early Fusion* **w+t+p**)	62.7
Ensemble Model (*Late Fusion* **w+t+p**)	64.4
[78]	**67.3**
[19]	62.4
BoW on HSTs, Sec. 3.5	53.3
[50]	36.9
[58]	42.4

Table 4.1: Mean average precision (mAP) on the HighFive dataset. We outperform all but [78], which extracts 10 times more features (Sec. 3.5.2) than our method.

Method	Accuracy
Ensemble Model (*Early Fusion* **w+t+p**)	**89.4**
Ensemble Model (*Late Fusion* **w+t+p**)	86.9
[83]	85.2
BoW on HSTs, Sec. 3.5	81.7
[63]	79.4
[72]	75.2
[49]	73.1

Table 4.2: Mean per-class accuracy on UCF Sports dataset.

Method	mAP
Ensemble Model (*Early Fusion* **w+t+p**)	31.3
Ensemble Model (Trees only)	28.9
[27]	36.9
[36]	30.8
[26]	14.1

Table 4.3: Mean average precision (mAP) on the Hollywood3D dataset. Our method only uses the left view of each video sequence, whereas the other methods exploit both (left,right) camera views and depth information. Using only a single view, our method can attain competitive mAP.

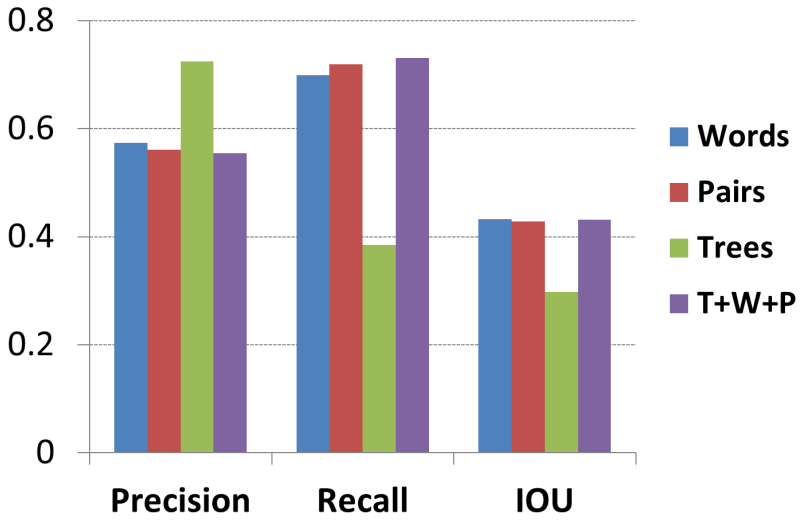

Figure 4.10: Action localization on UCF-Sports. Trees may miss some body parts, resulting in lower recall and IOU, but they localize the action performer(s) with much greater precision.

4.6.4.1 Impact of Action Words, Pairs and Trees

We decompose our full model and analyze the performance impact of each representational component: action words, pairs and trees. The results of this analysis are reported in Table 4.4 for UCF-Sports and HighFive. Some observations can be made. First, using only trees, pairs or action words, we attain performance that is comparable to state-of-the-art. Second, in all cases, combination works better than using a single type, especially when combining pair and tree structures with action words. This shows that the words, pairs and trees are complementary. We posit that larger structures are more discriminative but less frequently appear in their exact forms than smaller ones, leading to the complementarity observed.

4.6.4.2 Impact of Tree Size

We analyze the impact of tree size on the action classification performance for UCF-Sports in Fig 4.9, where the tree size is measured as the number of tree nodes. For

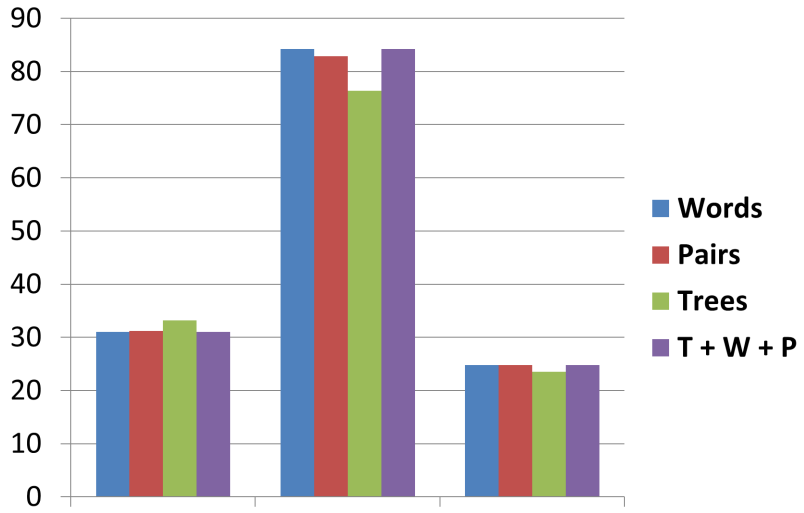

Figure 4.11: Action localization on HighFive. Trees can localize the action performer more precisely but may miss some body parts, leading to lower recall and IOU.

	w	**p**	**t**	**w+p**	**w+t**	**p+t**	**w+p+t**
High5	53.5	48.9	52.4	60.1	57.7	57.6	**62.7**
UCF	78.8	73.4	75.5	85.2	83.6	80.2	**89.4**

Table 4.4: Action recognition performance using only words, pairs or trees, as well as their combinations. To be consistent with experiments in the literature, we report mean average precision (mAP) for HighFive and mean per-class accuracy for UCF-Sports. Early fusion is used for combining the different types of structures.

trees of a given size, we order the trees by their *activation entropy* (Eq. 4.12) and measure the action classification performance when varying the number of trees used. When using only one tree, larger trees tend to outperform smaller trees. When more trees are used, larger trees produce better performance than smaller trees for most of the cases. These results show the increased discriminative power as we capture more complex hierarchical, spatial and temporal structures in the human actions. However, we also argue that smaller tree structures are complementary to larger tree structures. For instance, for videos of simple actions but with significant occlusion or video segmentation errors, smaller trees can be helpful when used in combination with larger tree structures; the combination attains state-of-the-art performance as

	dive	golf	kick	lift	ride horse	run	skate	swing-b	swing-s	walk
Ensemble Model	**58.6**	**54.9**	47.8	50.8	25.2	**35.5**	**38.9**	51.5	**25.0**	**43.0**
BoW (Sec. 3.5)	44.3	50.5	**48.3**	**51.4**	**30.6**	33.1	38.5	**54.3**	20.6	39.0

Table 4.5: Action localization performance on the UCF-Sports dataset using the full model (*i.e.*, T + W + P). The localization performance is measured by intersection over union (IOU). We compare with the Bag-of-Words model in Chapter 3. The proposed method in this Chapter improves the performance by exploring the space-time structures of human actions.

	hand shake	high five	hug	kiss
Ensemble Model	**27.1**	31.0	**34.8**	**30.0**
BoW (Sec. 3.5)	26.9	**32.9**	34.2	29.2

Table 4.6: Action localization performance on the HighFive dataset using the full model (*i.e.*, T + W + P), measured by IOU. We compare with the Bag-of-Words model in Chapter 3. The proposed method in this Chapter improves the performance in three out of the four interaction classes.

shown in Table 4.4.

4.6.5 Action Localization

Our method can predict the spatial location of the action performer(s) by computing the regions of the STSs that have positive contribution in classification. Localization results using the full model, *i.e.*, ensemble of trees, pairs and action words, on UCF-Sports and HighFive are shown in Table 4.5 and Table 4.6, respectively. Localization accuracy is measured by Intersection Over Union (IOU) with the ground-truth bounding boxes provided with these datasets. We compare with the Bag-of-Words model in Chapter 3, which employs the same bag-of-words model built upon the hierarchical space-time segments, but without using space-time structures of human actions. Even though the localization to the foreground is largely determined by how well the space-time segments align with the action performer, as analyzed in Sec. 3.5.2, exploring space-time structures of human actions offers a boost in local-

ization performance on both datasets.

We further analyze the action localization performance by different components of our model. For in-depth and complete analysis, we evaluate two more metrics besides the IOU: *precision*, which is the percentage of the predicted area that is within annotated bounding box, and*recall*, which is the percentage of the annotated bounding box areas that are predicted. We report the results for the UCF and HighFive datasets in Fig. 4.10 and Fig. 4.11, respectively. Interestingly, using only trees, the precision is consistently higher than using only words or pairs on both datasets, especially on the UCF-Sports dataset, but the recall is lower. Because the trees are learned in a discriminative approach, they may sometimes ignore body parts that are not discriminative; nevertheless, trees can more precisely localize the human(s) engaged in the action.

4.6.6 Cross-Dataset Validation

Our approach discoveres trees that capture discriminative hierarchical and spatiotemporal structures in human actions. These learned representations are generic, and could potentially be effective for recognizing actions in a dataset different from the training dataset.

To test this idea, we use the action word vocabulary and trees learned for *Kiss* and *Hug* on the HighFive dataset, and train action classifiers as ensembles of trees for the same actions on Hollywood3D. Similarly, we use the action word vocabulary and trees learned for *Golf Swing*, *Kick*, *Run Side* and *Walk Front* on the UCF-Sports dataset, and train action classifiers as ensembles of trees on JHMDB for the actions *Golf*, *Kick Ball*, *Run* and *Walk*. Thus, we consider cross-dataset evaluation for actions that appear in both datasets for each dataset pairing: (HighFive, Hollywood3D) and

Method	Kiss	Hug	Avg
Ours (trees from HighFive)	20.8	27.4	24.1
[26]	10.2	12.1	11.15
[27]	31.3	32.4	31.9

Table 4.7: Trees learned on one benchmark dataset are used to recognize actions in another: 50 trees learned for the Kiss and Hug actions in HighFive are used for recognizing these actions in Hollywood3D. The table reports the average precision (AP) for these classes. Note that our method only uses the left camera view, without using any multi-view or depth information exploited in [26], [27]. Moreover, the models for [26], [27] are trained for Hollywood3D.

(UCF-Sports, JHMDB). Only the ensemble weights (w_m in Eq. 4.1) are re-trained on Hollywood3D and JHMDB. As before, for the Hollywood3D dataset, we only use the left view of the RGB sequences (no multiview or depth information is used).

Action recognition results on Hollywood3D are shown in Table 4.7. Note that [26] and [27] use the depth information provided with Hollywood3D. Due to the differences between the HighFive and Hollywood3D datasets and the lack of multi-view and depth information, we do not expect to outperform state-of-the-art methods, but still we significantly outperform [26], doubling the performance in terms of average precision.

HighFive and Hollywood3D differ in many ways. Most importantly, the action categories differ significantly: while the trees are trained on HighFive to best discriminate *Hug* or *Kiss* from *High Five* and *Hand Shake*, on Hollywood3D they achieve promising performance in discriminating *Hug* or *Kiss* from 11 other actions such as *Eat* and *Use Phone* that the original tree discovery had no knowledge about. These results provide evidence for the generalization and discriminative power of the trees we discover, and show potential for re-use of the learned trees across datasets.

The action recognition results on JHMDB are shown in Table 4.8. As foreground mask annotations are available for the JHMDB dataset, we also compute action lo-

	Golf	Kick Ball	Run	Walk
AP	29.7	16.8	23.4	42.5

Table 4.8: Trees learned on the UCF-Sports dataset are used to recognize actions in the JHMDB dataset. Average precision (AP) of action recognition performance (averaged over three splits) is reported for the action categories that are included in both JHMDB and UCF-Sports.

	Golf	Kick Ball	Run	Walk
IOU	18.3	21.7	12.0	18.4
Recall	78.1	61.1	65.7	80.5
Precision	22.1	28.8	16.3	19.9

Table 4.9: Action localization on JHMDB using trees learned on UCF-Sports. Performance (averaged over three splits) is reported for the action categories that appear in both JHMDB and UCF-Sports.

calization performance and report the results in Table 4.9. Note that the localization performance is evaluated at the pixel level: we compare the foreground regions predicted by our method against the ground truth foreground masks, using the IOU measure.

Comparing to JHMDB, the UCF-Sports dataset is significantly smaller. The number of videos in UCF-Sports is only approximately one sixth of the number of videos in JHMDB, and UCF-Sports has only half the number of actions in JHMDB. Furthermore, the corresponding action classes are not exactly the same. Comparing the classes of UCF-Sports vs. JHMDB: *Golf Swing* vs. *Golf*, *Kick* vs. *Kick Ball*, *Run Side* vs. *Run*, and *Walk Front* vs. *Walk*. These differences in the action categories lead to differences in the training and testing videos between the two datasets. Nonetheless, with trees learned on UCF-Sports, we still achieve promising action recognition and localization results.

4.7 Summary

Our approach in this Chapter discovers a compact set of hierarchical space-time tree structures of human actions from training videos. Using an ensemble of the discovered trees, or in combination with simpler action words and pairwise structures, we build action classifiers that achieve promising results on three challenging datasets: HighFive, UCF-Sports and Hollywood3D. We also show cross-dataset generalization of the trees learned on HighFive on the Hollywood3D dataset, as well as of the trees learned on UCF-Sports on the JHMDB dataset.

Chapter 5

Learning Action Progression in LSTMs

Besides spatially localizing an action in a video (Chapter 3, Chapter 4), which answers *where* the action is, in many situations we also want be able to tell *when* an action happened in a video sequence, *i.e.* determining its starting and ending time point. In this chapter, we study human action detection and early detection in videos (Fig. 1.2). For action detection, we detect segments of human actions in a video sequence, recognizing the actions' categories and detecting their start and end points. For early detection, we detect the action segment after observing only a fraction of the action.

Automatic detection of human actions, in videos, has many potential applications, such as video understanding and retrieval, automatic video surveillance, and human-computer interaction. Further, for many applications, such as human-robot interaction it is desirable to detect the action as early as possible [29, 67], to make the interaction more natural, *e.g.*, deploying a robot to help an elderly patient stand up before he/she is upright and is risking a fall.

action detection in realistic settings is quite challenging. There is high variability in the viewpoint from which the action is observed, the actors and their appearance, as well as the execution and overall duration of the actions (see Fig. 5.5). This is particularly true for relatively long and complex actions [1]. For example, the action

[1]Some works such as [28] refer to long and complex actions as activities. We do not make such

"making pasta" typically entails *cutting vegetables*, *setting a pot on the fire*, *making boiling water*, *boiling pasta noodles*, *cooking pasta sauce*, and *combining pasta with sauce*. To better detect, *i.e.*, recognize and temporally localize such actions, we argue that it is critically important for the learned detector to model the actions' temporal progression.

Recurrent Neural Network (RNN) models are particularly helpful in this context: the prediction at each time instant is based not only on the observations at that time instant, but also on the previous model hidden states that provide temporal context for the progression of the action. More specifically, in the Long Short Term Memory (LSTM), a type of RNN, *memory* is used to capture useful patterns of previous observations, and is used in addition to the previous hidden states to provide longer-range context (*e.g.*, as compared to HMMs) for the current prediction.

While RNN models are powerful, using only classification loss in training such models typically fails to properly penalize incorrect predictions, *i.e.*, the prediction error is penalized the same no matter how much context the model has already processed. For example, given a video of the action *making pasta*, to output the action class label *preparing coffee* after the detector sees the action up to *combining pasta with sauce* should be penalized more than the same error when the detector only sees up to *making boiling water*.

The above mentioned defect in training RNN models is especially critical for action detection. Unlike conventional applications of RNNs in machine translation and speech recognition, in which specific output such as words or phonemes continue for a relatively short time, human actions such as *making pasta* may continue for a relatively long period, *e.g.*, several minutes or thousands of video frames. It is thus

distinctions in this thesis

very important for the model to learn the progresssion patterns of the actions in training.

In this Chapter, we introduce novel *ranking losses* within the RNN learning objective so that the trained model better captures progression of actions. These ranking losses are computed for the prediction at each time point, while also taking into consideration the past predictions starting from the very beginning of the action.

The intuition for our formulation is shown in Fig. 5.1. As the detector sees more of an action, it should: (1) become more confident of the correct action category, *i.e.*, output a higher detection score for the correct category as the action progresses, and (2) become more confident of the absence of incorrect categories, *i.e.*, the detection score margin between the correct and incorrect categories should be non-decreasing as the action progresses.

Thus, we introduce two explicit constraints in RNN training. The first is a ranking loss on the detection score of the correct category, which constrains the detection score of the correct categorty to be monotonically non-decreasing as the action progress. The second is a ranking loss on the detection score margin between the correct action category and all other categories, which constrains that this discriminative margin is monotonically non-decreasing.

Contributions: We make the following contributions in this chapter:

- We propose formulations for ranking loss on the detection score and on the discriminative margin to better learn models for human action progression.
- We implement our proposed ranking losses in training LSTM models, and show significant improvements over LSTM model trained only with classification loss in the tasks of action detection and early detection.
- We achieve start-of-the-art performance for action detection and early detection

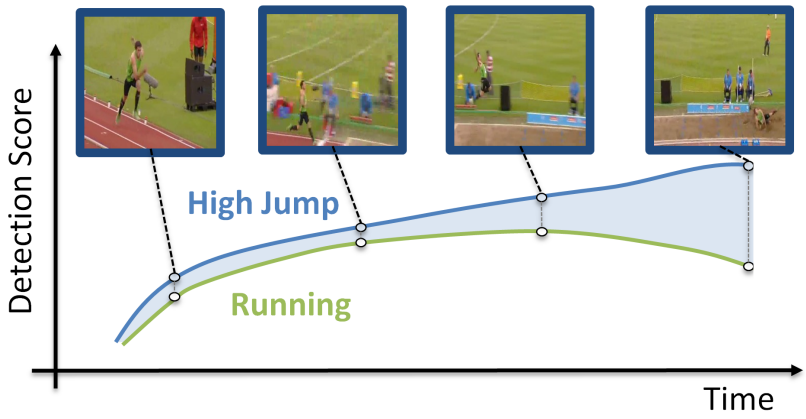

Figure 5.1: As the detector sees more of the action, it should become more confident of the presence of the correct category and absence of incorrect categories. This example sequence contains a *high jump*. The blue curve is the detection score of the correct category, which is encouraged to be non-decreasing. The green curve is the detection score of an incorrect category *running*, whose margin with respect to the correct category (shaded light blue area) is encouraged to be non-decreasing.

on a large-scale video dataset: ActivityNet [28].

Roadmap for this chapter

We first present an overview of our model in Sec. 5.1. Subsequently, in Sec. 5.2 we describe our two novel ranking losses: these losses are designed to be used in training of the model to better learn temporal progression of the human actions in videos. We then discuss training of the model in Sec. 5.2.3. Finally, in Sec. 5.3 we present extensive experimental evaluation and analysis of our model and the proposed ranking losses on a large scale video dataset ActivityNet [28].

5.1 Model Overview

Fig. 5.2 illustrates our model for action detection. It contains two major components: a CNN that computes visual features from each video frame, and an LSTM with a linear layer that computes action detection scores based on the CNN features of the

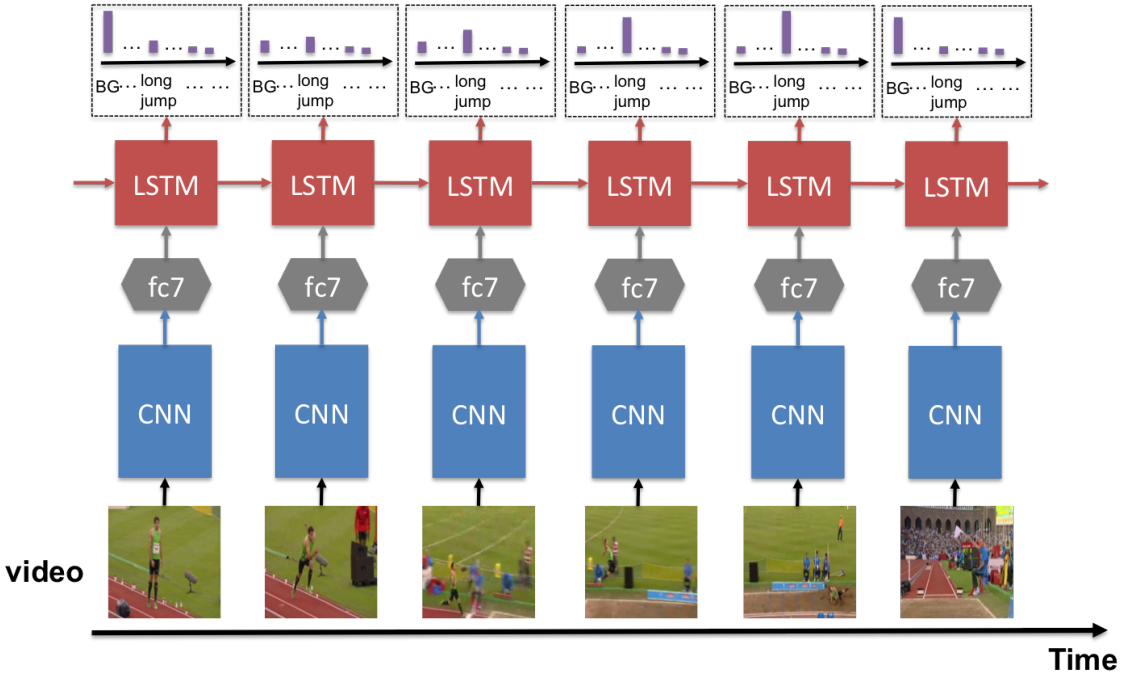

Figure 5.2: **Model overview.** At each video frame, the model first computes CNN features (illustrated as *fc7*) and then the features are fed into the LSTM to compute detection scores of actions and non-action (*BG* in the figure).

current frame and the hidden states and memory of the LSTM from the previous time step. We adopt the CNN architecture VGG19 [70], whose output of the second fully connected layer (*fc7*) is fed into the LSTM. We use the LSTM described in [61] that applies dropout on non-recurrent connections. A similar model has been used in [98] for detecting relatively short actions. Our key contributions are in exploring the rank losses, during training, that encourage monotinicity in detection score and margin produced by the model as a training action progresses.

5.2 Learning action Progression

To accurately detect the complete duration of human actions, especially for relatively long and complex ones, it is important for the model to capture the progression

patterns of actions during training. An RNN only implicitly considers progression via the context that is passed along time in the form of the previous hidden state and, in LSTM, memory as well. We introduce ranking loss into the learning objective, to explicitly capture action progression globally from the action start to the current time:

$$\mathcal{L}^t = \mathcal{L}_c^t + \lambda_r \mathcal{L}_r^t, \tag{5.1}$$

where $\mathcal{L}_c^t$ and $\mathcal{L}_r^t$ are the classification loss and the ranking loss, respectively, at the current time t, and λ_r is a positive scalar constant.

Usually, for training deep models, the cross entropy loss is used to formulate $\mathcal{L}_c^t$:

$$\mathcal{L}_c^t = -\log p_t^{y_t}, \tag{5.2}$$

where y_t is the ground truth action category of the training video sequence at the tth video frame, and $p_t^{y_t}$ is the detection score of the ground truth label y_t for the tth frame, *i.e.*, the softmax output of the model.

We explore two formulations of the ranking loss, $\mathcal{L}_r^t$. The first constrains the model to output a non-decreasing detection score for the correct category throughout the duration of the action. Our second ranking loss constrains the output of the model to have non-decreasing discriminative margin: at any point in the action, the margin between the detection score of the correct category and the maximum detection score among all other categories should be non-decreasing. Detailed formulations of these two ranking losses are given below. For easier reading, we use $\mathcal{L}_s^t$ and $\mathcal{L}_m^t$ to denote ranking loss on the detection score and margin, respectively. While these two ranking losses are different, they are related. Note that the output of softmax layer of the LSTM sums to 1, so $\mathcal{L}_s^t$ considers the margin between $p_t^{y_t}$ and

$\sum_{y' \neq y_t} p_t^{y'}$, whereas $\mathcal{L}_m^t$ considers the margin between $p_t^{y_t}$ and $\max_{y' \neq y_t} p_t^{y'}$. We will discuss this more at the end of Section 5.3.6.

5.2.1 Ranking Loss on Detection Score

Ideally we want the action detector to produce monotonically non-decreasing detection scores for the correct action category as the detector sees more of the action (Fig. 5.1). To this end, we introduce the ranking loss $\mathcal{L}_s^t$ into the learning objective at time step t as:

$$\mathcal{L}_s^t = \max(0,\ -\delta_t \cdot (p_t^{y_{t-1}} - p_t^*)), \tag{5.3}$$

where δ_t is set to 1 if $y_{t-1} = y_t$, *i.e.*, when there is no action transition from $t-1$ to t according to ground truth labeling (*e.g.* $\delta_t = 1$ for t_a, t_b and t in Fig. 5.3); otherwise, δ_t is set to -1.

In Eq. (5.3) p_t^* is computed as:

$$p_t^* = \begin{cases} p_t^{*y_t}, & \text{if} \quad \delta_t = 1, \\ 0, & \text{otherwise}, \end{cases} \tag{5.4}$$

where

$$p_t^{*y_t} = \max_{t' \in [t_s,\ t-1]} p_{t'}^{y_t}. \tag{5.5}$$

$$t_s = \min\{t' \mid y_{t'} = y_t,\ \forall t' \in [t_s,\ t]\}, \tag{5.6}$$

where t_s is the starting point of the current action y_t, and $p_t^{*y_t}$ is the highest previous detection score in $[t_s, t-1]$ (illustrated by the dashed horizontal line in Fig. 5.3).

In other words, if there is no action transition at time t, *i.e.*$y_t = y_{t-1}$, then we want the current detection score to be no less than any previous detection score for

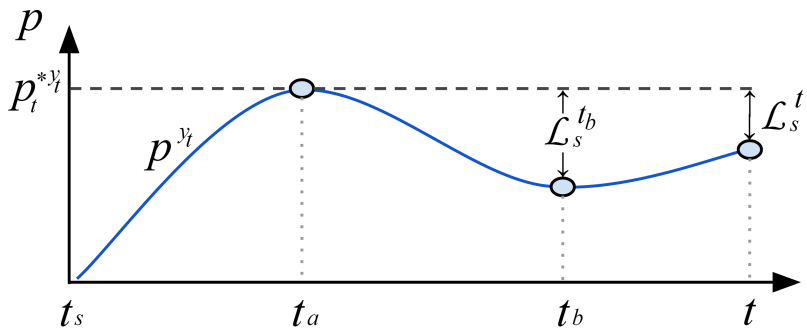

Figure 5.3: Detection score p^{y_t} (blue curve) of an action y_t spanning $[t_s, t]$. $p_{t_b}^{y_t}$ and $p_t^{y_t}$ are smaller than $p_{t_a}^{y_t}$ (which is also $p_t^{*y_t}$ in this example), violating the monotonicity of the detection score, so $\mathcal{L}_s^{t_b}$ and $\mathcal{L}_s^{t}$ are non-zero.

the same action, computing the ranking loss as:

$$\mathcal{L}_s^t = \max(0,\ p_t^{*y_t} - p_t^{y_t}). \tag{5.7}$$

On the other hand, if an action transition happens at time t, *i.e.*$y_t \neq y_{t-1}$, we want the detection score of the previous action to drop to zero at t and compute the ranking loss as:

$$\mathcal{L}_s^t = p_t^{y_{t-1}}. \tag{5.8}$$

Fig. 5.3 shows the detection scores p^{y_t} (the blue curve) of an action y_t spanning $[t_s,\ t]$. In $[t_a + 1,\ t]$, the detection scores are smaller than $p_{t_a}^{y_t}$, violating the monotonicity of the detection score, so the ranking losses in this period are non-zero, *e.g.*$\mathcal{L}_s^{t_b}$ and $\mathcal{L}_s^t$ as shown in the figure.

One may be tempted to simply require $p_t^{y_t}$ to be no less than $p_{t-1}^{y_t}$ when there is no action transition, replacing Eq. 5.7 with:

$$\mathcal{L}_s^t = \max(0,\ p_{t-1}^{y_t} - p_t^{y_t}). \tag{5.9}$$

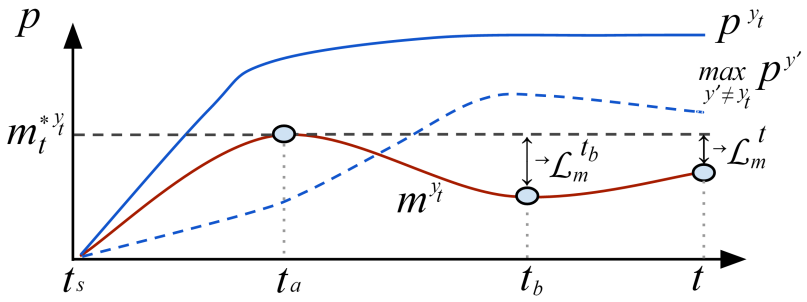

Figure 5.4: Discriminative margin m^{y_t} (red curve) of an action y_t spanning $[t_s,\ t]$. The margin m^{y_t} is computed as the difference between the ground truth action detection scores p^{y_t} (blue curve) and the maximum detection scores $\max_{y' \neq y_t} p^{y'}$ (dashed blue curve) of all incorrect action categories at each time point in $[t_s,\ t]$. $m_{t_b}^{y_t}$ and $m_t^{y_t}$ are smaller than $m_{t_a}^{y_t}$ (which is also $m_t^{*y_t}$), violating the monotonicity of the margin, so $\mathcal{L}_m^{t_b}$ and $\mathcal{L}_m^{t}$ are non-zero.

However, as shown in Fig. 5.3, in this situation, the ranking loss will be zero in $[t_b + 1,\ t_c]$ even though the monotonicity of detection score is also violated.

5.2.2 Ranking Loss on Discriminative Margin

When more of an action is observed, the detector should become more confident in discriminating between the correct category vs. the incorrect categories. We guide the training of our model to acquire such behavior by implementing the following ranking loss:

$$\mathcal{L}_m^t = \max(0,\ -\delta_t \cdot (m_t^{y_{t-1}} - m_t^*)). \tag{5.10}$$

where m_t^y is the discriminative margin of an action label y at time step t (the blue point on the red curve at time t in Fig. 5.4), computed as:

$$m_t^y = p_t^y - \max\{p_t^{y'} \mid \forall y' \in \mathcal{Y},\ y' \neq y\}, \tag{5.11}$$

where $\mathcal{Y}$ is the set of all action category labels. The m_t^* in Eq. 5.10 is computed as:

$$m_t^* = \begin{cases} m_t^{*y_t}, & \text{if } \delta_t = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{5.12}$$

where $m_t^{*y_t}$ is computed as:

$$m_t^{*y_t} = \max_{t' \in [t_s,\ t-1]} m_{t'}^{y_t}, \tag{5.13}$$

*i.e.*the largest previous discriminative margin of the current action y_t that started at t_s (illustrated by the dashed horizontal line in Fig. 5.4).

In other words, when there is no action transition at t, we want the current discriminative margin to be no less than any previous margin in the same action, computing the ranking loss as:

$$\mathcal{L}_m^t = \max(0,\ m_t^{*y_t} - m_t^{y_t}). \tag{5.14}$$

If an action transition happens at time t, we want the discriminative margin of the previous action to drop and compute the ranking loss as:

$$\mathcal{L}_m^t = m_t^{y_{t-1}}. \tag{5.15}$$

Fig. 5.4 illustrates the discriminative margins (red curve) m^{y_t} of the current action y_t spanning $[t_s, t]$. The margin m^{y_t} is equal to the difference between the detection scores p^{y_t} of the correct category y_t (blue curve) and the maximum of the detection scores $\max_{y' \neq y_t} p^{y'}$ for the incorrect categories (dashed blue curve). Note that within the time interval $[t_a + 1, t]$, the margins are smaller than $m_{t_a}^{y_t}$, violating

the monotonicity; consequently, the ranking losses are non-zero within the interval $[t_a + 1, t]$. Also note that simply requiring the current margin to be less then that of the previous timestep is insufficient, which will result in zero ranking loss in interval $[t_b + 1,\ t]$ in Fig. 5.4.

5.2.3 Training

In training, we compute the gradient of the ranking loss with respect to the softmax output at each time step:

$$\frac{\partial \mathcal{L}^t}{\partial p_t^y} = \frac{\partial \mathcal{L}_c^t}{\partial p_t^y} + \lambda_r \frac{\partial \mathcal{L}_r^t}{\partial p_t^y} \tag{5.16}$$

which is then back propagated through time to compute the gradients with respect to the model parameters. Also note that we fix the ranking loss of non-action frames to be 0. Although $\mathcal{L}_s^t$ and $\mathcal{L}_m^t$ are also functions of $p_{t'}^y$ for $t' < t$, *i.e.*, the softmax output of previous time steps, to simplify computation, we do not compute and back propagate the gradients of the ranking loss with respect to these variables.

5.3 Experiments

We evaluate our formulation on a large-scale, realistic action dataset: ActivityNet [28]. Using our proposed ranking losses in training significantly improves performance in both the action detection and early action detection tasks.

5.3.1 Dataset

The ActivityNet [28] dataset comprises 28K videos of 203 action categories collected from YouTube. Fig. 5.5 shows sample frames from video sequences of this dataset.

Figure 5.5: Each row contains sample frames of one example video sequence in ActivityNet. Frames with green borders contain the actions labeled on the left. Note the significant viewpoint and foreground object changes within the actions *Using ATM*, *Sailing* and *Preparing Pasta*.

The lengths of the videos range from several minutes to half an hour. The total length of the whole dataset is 849 hours. A single video may contain multiple actions and often also contains periods with none of the annotated actions. On average, 1.4 actions are annotated per video. The action category, along with the start and end point of each action are annotated by crowd-workers, leading to some annotation noise. Many of the videos are shot by amateurs in uncontrolled environments, and variances within the same action category are often large. More importantly, many actions are relatively long and complex, and the viewpoint and foreground objects

may change significantly within the same action, *e.g.*, *Using ATM* and *Preparing pasta* shown in Fig. 5.5. Given these challenges, it is important that the model learns the progression of actions for accurate action detection and early detection.

The authors of ActivityNet use one fourth of the dataset as a *validation set*, but have not released the *test set* used in their paper.[2] In our experiments, we use the validation set as our test set, and we split the remaining videos into one fifth for validation and four fifths for training. To reduce computational cost, we temporally down-sample the videos to 6 frames per second for all our experiments.

5.3.2 Model Training

For the CNN component (see Fig. 5.2), we first use training video frames of ActivityNet to fine-tune a VGG19 model [70] that is pre-trained on ImageNet. The output dimension of the softmax layer is 204, which corresponds to the 203 actions plus one additional class corresponding to non-action. We set the learning batch size to 32. The learning rate starts at 10^{-4} and is divided by 10 after every 40K iterations. The fine-tuning stops at 120K iterations.

For LSTM training, the output of the second fully connected layer (*fc7*) of the fine-tuned VGG19 model is used as input to the LSTM. We use learning batches of 64 sequences, where each sequence comprises 100 frames. Back propagation through time is performed for 20 time steps. The momentum and weight decay are set to 0.9 and 0.0005 respectively. The learning rate starts at 0.01 and is divided by 10 after every 20K iterations. Training stops after 50K iterations. In this training phase, the CNN *fc7* layer is also further trained together with the LSTM but with a lower starting learning rate of 10^{-4}.

[2] According to communication with the authors of [28], this test split is kept confidential for use in a future challenge.

Table 5.1: Action detection performance measured in mAP at different IOU thresholds α. Note that the results of Heilbron *et al.*[28] are produced on their own test split that is unavailable to us, so their results are not directly comparable to ours.

Model	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$	$\alpha = 0.7$	$\alpha = 0.8$
Heilbron *et al.*[28]	12.5%	11.9%	11.1%	10.4%	9.7%	-	-	-
CNN	30.1%	26.9%	23.4%	21.2%	18.9%	17.5%	16.5%	15.8%
LSTM	48.1%	44.3%	40.6%	35.6%	31.3%	28.3%	26.0%	24.6%
LSTM-m	52.6%	48.9%	45.1%	40.1%	35.1%	31.8%	29.1%	27.2%
LSTM-s	**54.0%**	**50.1%**	**46.3%**	**41.2%**	**36.4%**	**33.0%**	**30.4%**	**28.7%**

5.3.3 Experimental Setup

In testing, we run the model across the whole input sequence and output action detection scores at each input frame. We reset the LSTM memory whenever the model predicts non-action, which we find slightly improves performance. For both action detection and early detection, we detect video segments of actions from an input video sequence. To achieve this, we first classify each video frame to the action category for which the detection score is the highest at this frame. Note that non-action is treated simply as a special category. We then find continuous video frame segments that are classified to belong to the same action category; this produces the initial detection spans. Finally, we iteratively merge the detection spans that are temporally close (less than 20 frames apart in our experiments). The score of each detection is then computed as the mean of the detection scores of all its video frames.

Following [28], we use the mAP (mean average precision) in evaluating performance. A detection is a true positive if: 1) its IOU (intersection-over-union) of temporal duration with a ground truth action is above the IOU threshold, and 2) its action label is equal to the ground truth action label. If multiple detections overlap with the same ground truth action, only the one with the longest duration is treated as a true positive. All the other detections are false positives. For evaluating performance on early detection, we split each input test sequence into multiple sequences

so that each new sequence contains the non-action segment (if there is any) before a test action, and a portion of the test action.

We evaluate the performance of four models: i) the fine-tuned VGG19 CNN model; ii) the LSTM model shown in Fig. 5.2 trained with the classification loss only (Eq. 5.2); iii) the LSTM-s model trained with both the classification loss and ranking loss on the detection score (Eq. 5.3); iv) the LSTM-m model trained with classification loss and ranking loss on the discriminative margin (Eq. 5.10). In LSTM-s and LSTM-m, the weight for ranking loss (λ_r in Eq. 5.1) is empirically set to 6, according to performance on our validation set. We find that using a combination of both ranking losses in training offers no further improvement over using just one so we do not include results for this in the paper.

5.3.4 Action Detection

Table 5.1 shows the action detection performance of the evaluated models under different IOU thresholds, α. The results of Heilbron *et al.*[28] are produced on their test set, which is not publicly available; therefore, their results are not directly comparable to ours. Heilbron *et al.*[28] use a sliding window approach to detect actions in the video sequences, where the temporal lengths of the sliding windows are empirically selected and fixed. In our approach the length of each detection is automatically determined as described in Section 5.3.3.

The LSTM models greatly outperform the CNN model. This demonstrates the benefit of using a recurrent neural network model in action detection. Both of the proposed ranking losses are beneficial in training a better LSTM model for action detection: significant improvements are achieved over the LSTM model trained only using classification loss. For LSTM-s the improvements are consistently around 4.1$\sim$5.9%

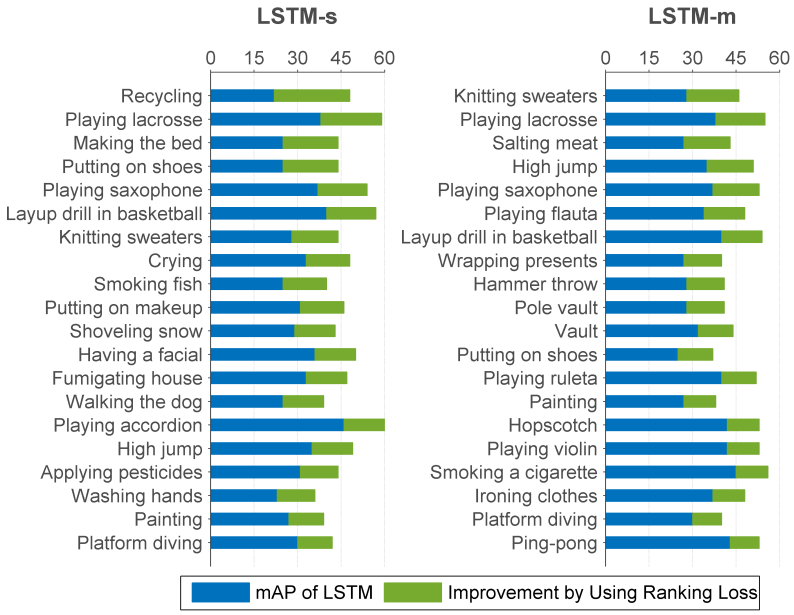

Figure 5.6: Top 20 action categories for which the detection performance improved the most by using either the ranking loss on the detection score (LSTM-s) or discriminative margin (LSTM-m) in training.

at all IOU thresholds. Note that the relative improvement of LSTM-m and LSTM-s over LSTM increases when requiring the detection to more accurately overlap with ground truth, *e.g.*, growing from 12.3% when $\alpha = 0.1$ to 16.7% when $\alpha = 0.8$ with LSTM-s. This shows that the proposed ranking-losses are even more useful in applications where accurate temporal localization is required.

Fig. 5.6 shows the top 20 actions for which the detection performance are improved the most by using ranking loss (LSTM-s or LSTM-m) in training (IOU threshold $\alpha = 0.5$). It is interesting to note that using the proposed ranking losses, detection performance improves both for relatively simple actions such as *playing saxophone* and for relatively complex actions such as *high jump*. This shows that the proposed ranking losses may improve the detection of various types of actions.

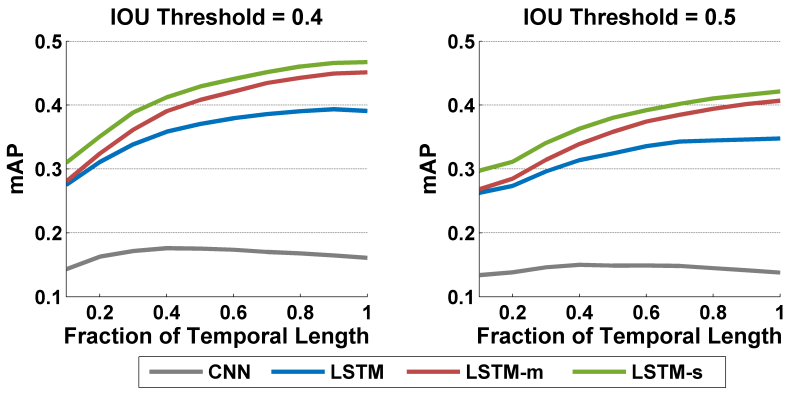

Figure 5.7: Action early detection performance plotted as a function of the observed fraction of each test action. LSTM models greatly outperform the CNN model irrespective of the fraction of the action observed. Using the proposed ranking losses in training, LSTM-s and LSTM-m, outperform LSTM and the performance gaps increase as more of the action is observed. Also note the performance gap between LSTM-s and LSTM at the start of the curves (when we only observe one tenth of each action).

Table 5.2: Action early detection performance at different IOU thresholds (α), when *only* 3/10 of each action is observed.

Model	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$	$\alpha = 0.7$	$\alpha = 0.8$
CNN	27.0%	23.4%	20.4%	17.2%	14.6%	12.3%	11.0%	10.3%
LSTM	49.5%	44.7%	38.8%	33.9%	29.6%	25.6%	23.5%	22.4%
LSTM-m	52.6%	47.9%	41.5%	36.2%	31.4%	27.1%	24.8%	23.5%
LSTM-s	**55.1%**	**50.3%**	**44.0%**	**38.9%**	**34.1%**	**29.8%**	**27.4%**	**26.1%**

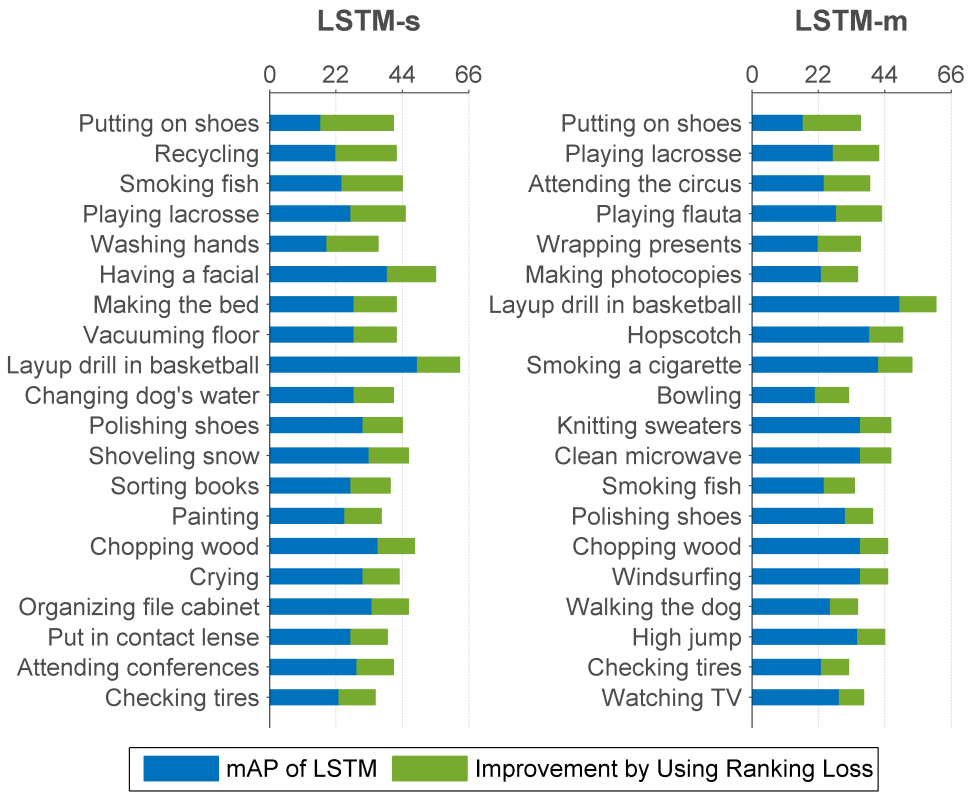

Figure 5.8: Top 20 action categories for which the early detection performance improved the most by using either the ranking loss on the detection score (LSTM-s) or discriminative margin (LSTM-m) in training. Only the first 3/10 of each test action is observed. The IOU threshold $\alpha = 0.5$.

5.3.5 Action Early Detection

In this experiment, the goal is to recognize and also temporally localize partially observed actions. Table 5.2 shows the detection performances when we only observe 3/10, *i.e.*, approximately the first third, of each testing action. The LSTM models greatly outperform the CNN model on the early detection task. Moreover, the LSTM models trained with the proposed ranking losses (LSTM-s or LSTM-m) clearly outperform the LSTM model trained only with classification loss. For instance, with LSTM-s, the absolute improvements are consistently around 5.6∼3.7% at all IOU thresholds α, with relative improvement increasing from 11.3% at $\alpha = 0.1$ to 16.5%

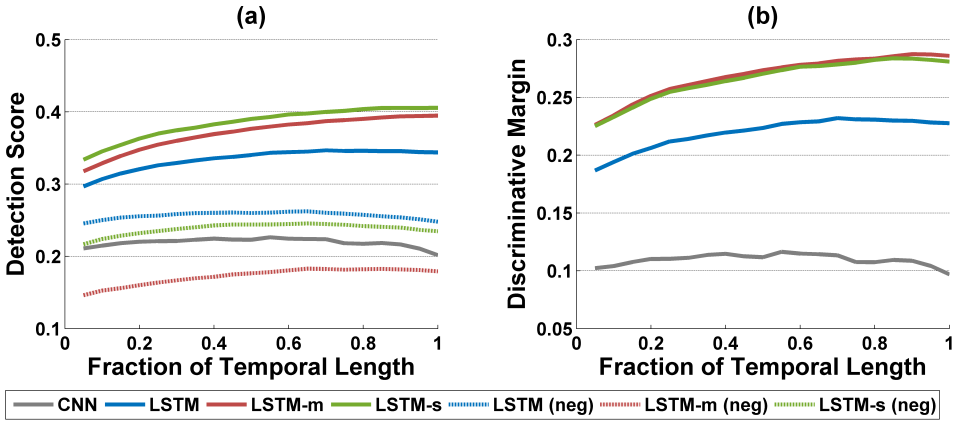

Figure 5.9: Mean curves of (a) the detection score and (b) the discriminative margin, as function of time over all test sequences for CNN, LSTM, LSTM-m and LSTM-s. The mean curves of the detection score for the *worst negative category*, *i.e.*negative action category with the highest detection score, are also shown as the dashed curves.

at $\alpha = 0.8$.

Fig. 5.7 shows the performance of early detection when the observed fraction of each test action increases from 0.1 to 1 when the IOU threshold is fixed at 0.4 or 0.5. All LSTM models greatly outperform the CNN, no matter how much of each action is observed. Both ranking losses, LSTM-m and LSTM-s (red and green curves) outperform LSTM (blue curve). Although the increase in detection performance slows down after observing approximately half of each action, the performance gap between LSTM-s (LSTM-m) and LSTM increases as more of each action is observed. More interestingly, LSTM-s significantly outperforms LSTM even when we only observe a small faction of each action, *e.g.*, one tenth. This could be quite useful for applications that require detecting actions as early as possible.

Fig. 5.8 lists the top 20 action categories for which the early detection performance improves the most when using either of the proposed ranking losses in training. Only the first 3/10 of each action is observed. The IOU threshold $\alpha = 0.5$. Interestingly, among these actions, some may have relatively little visual content change across the

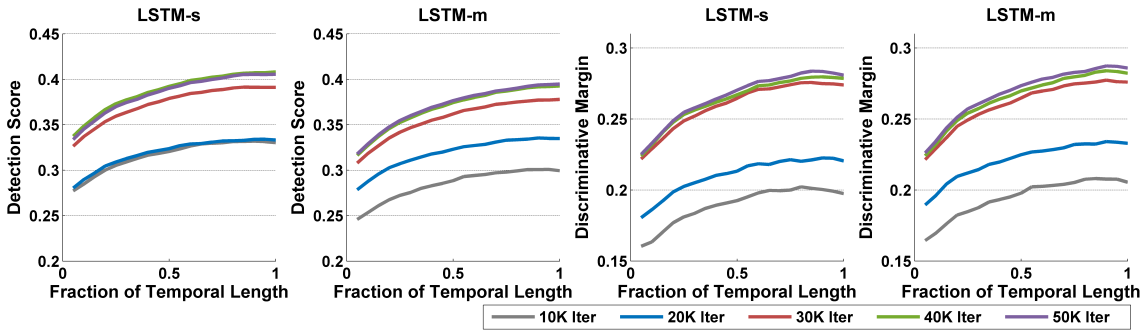

Figure 5.10: Mean curves of the detection score and the discriminative margin as function of time over all test action sequences produced by snapshots of the LSTM-m and LSTM-s models trained after 10K, 20K, 30K, 40K and 50K iterations.

whole duration of the action, such as *Playing lacrosse*, whereas others may undergo significant visual content change, such as *Layup drill in basketball*. This suggests that the benefits of the proposed ranking losses are applicable to various types of actions in the task of early detection.

5.3.6 Effects of the Ranking Losses

We now analyze what effects the proposed ranking losses introduce over the evolving time scale of model training. We first analyze how the detection score of the correct action category and the discriminative margin (Eq. 5.11) change as we train LSTM-m and LSTM-s. We compute the detection scores and the discriminative margins at every frame in each test sequence using snapshots of the LSTM models trained after 10K, 20K, 30K, 40K and 50K iterations. This produces for each action sequence a curve of the detection score (or discriminative margin) as a function of time. We normalize the curves so that each has a length of 20 points, and finally compute the mean curve over the whole test set. Fig. 5.10 shows the mean curves. For both models, the mean curves are approximately non-decreasing, and such monotonicity becomes more apparent as we train for more iterations. The absolute values of the

detection scores and discriminative margins increase as we train for more iterations, but converge after roughly 40K training iterations.

Fig. 5.9 compares the mean curves of the detection score and discriminative margin produced by LSTM-s, LSTM-m and LSTM trained after 50K iterations, as well as the CNN model. The mean curves of LSTM-s and LSTM-m (solid green curves and solid red curves) for both the detection score and discriminative margin are significantly higher than those of LSTM (blue curves). The LSTM-s and LSTM-m curves also show a more apparent monotone increasing trend compared to LSTM, which tends to be flat after approximately the first half of the action. We also show the mean detection score curves for the *worst negative category*, *i.e.*, the negative action category with the highest detection score for LSTM, LSTM-s and LSTM-m using the dashed curves. The curves of LSTM-m and LSTM-s for the *worst negative category* are lower than that of LSTM.

It is interesting to note that each of the proposed ranking losses has useful impacts on both the detection score and the discriminative margin, despite the fact that they are either computed based on detection scores only or discriminative margins only. This conforms to our intuition that encouraging a non-decreasing detection score may help in producing a non-decreasing discriminative margin and vice versa. Also note that LSTM-s produces higher detection scores for the correct category than LSTM-m, while LSTM-m pushes the detection scores of the *worst negative category* significantly lower, as shown Fig. 5.9 (a). In practice one can use either of these ranking loss formulations, depending on the application, *e.g.*selecting the best one via cross-validation.

5.4 Summary

In this Chapter we explore deep LSTM model for action detection and early detection. We introduce and exploit two novel formulations for ranking loss in LSTM training, designed to encourage consistent scoring and margin for detecting the correct action as more of the action sequence is observed. We show significant performance improvements in action detection and early detection on ActivityNet by using the proposed ranking losses in training. In future work, we plan to conduct further in-depth study of the relative advantages of the two ranking losses.

Chapter 6

Utilizing Web Images for Training CNN Models

The model in Chapter 5 utilizes deep Convolutional Neural Network (CNN) for learning visual features from video frames. Recent works [42, 69] also show that deep CNN models are promising for action recognition in videos. However, CNN models typically have millions of parameters [7, 47, 70], and usually large amounts of training data are needed to avoid overfitting. For this purpose, work is underway to construct datasets consisting of millions of videos [42]. However, the collection, pre-processing, and annotation of such datasets can require a lot of human effort. Moreover, storing and training on such large amounts of data can consume substantial computational resources.

In contrast, collecting and processing images from the Web is much easier. For example, one may need to look through all, or most, video frames to annotate the action, but often a single glance is enough to decide on the action in an image. Videos and web images also have complementary characteristics. A video of 100 frames may convey a complete temporal progression of an action. In contrast, 100 web action images may not capture the temporal progression, but do tend to provide more variations in terms of camera viewpoint, background, body part visibility, clothing, *etc.* Moreover, videos often contain many redundant and uninformative frames, *e.g.*,

standing postures, whereas action images tend to focus on discriminative portions of the action (Fig. 6.1). This property can further focus the learning, making action images inherently more valuable.

Figure 6.1: Sample action images from our dataset, BU101. Action images on the Web often capture well-framed descriminative poses of the actions they represent. Left to right: *Hammer Throw*, *Body Weight Squats*, *Jumping Jack*, *Basketball*, *Tai Chi*, *Cricket Shot*, *Lunges*, *Still Rings*. Utilizing web action images in training CNNs, for all these action classes, results in more than 10% absolute increase in recognition accuracy in videos compared to CNNs trained only on video frames (see Fig. 6.3).

In summary, two intuitions emerge about why web action images may be useful in training CNNs for action classification of videos:

- *Complementarity*: Action images may complement training videos when video data is scarce, particularly since images may be easier to collect and process.
- *Efficiency*: Web action images usually contain discriminative poses of the actions, making them intrinsically higher-quality training data compared to video frames, which may be redundant or contain less relevant poses.

However, it is not enough to stop at these seemingly natural intuitions: scientific verification is necessary. In this work, we analyze and empirically evaluate these intuitions. To our best knowledge, we are the first to perform an in-depth analysis of this problem by extensive and large-scale empirical evaluation.

We start by collecting large web action image datasets. The first dataset, BU101, contains 23.8K images of 101 action classes. It is more than double the size of the largest previous action image dataset [97], both in the number of images and the number of actions. And, to the best of our knowledge, this is the first action image

dataset that has one-to-one correspondence in action classes with the large-scale action recognition video benchmark dataset, UCF101 [44]. Images of the dataset are carefully labeled and curated by human annotators; we refer to them as *filtered* images. Two other, even larger, web image datasets are also collected: BU101-unfiltered and BU203-unfiltered, which are crawled automatically by querying action class names on multiple image search engines, *e.g.* Google Image Search. The BU101-unfiltered dataset contains ∼0.2M images crawled by querying the 101 action class names of UCF101, and BU203-unfiltered contains ∼0.4M images crawled by querying the 203 activity names of ActivityNet [6]. All these datasets will be made publicly available to the research community [1].

We train CNN models of different depths and analyze the effect of adding web action images of BU101 to the training set of video frames. We also train and evaluate models with varying numbers of action images to explore the marginal gain as a function of the web image set size. We find that by combining web action images with video frames in training, a spatial CNN can achieve an accuracy of 83.5% on UCF101, which is more than a 10% absolute improvement over a spatial CNN trained only on videos [69]. When combining with motion features, we can achieve 91.1% accuracy, which is the highest result reported to-date on UCF101. We also replace videos by images to demonstrate that our performance gains are due to images providing complementary information to that available in videos, and not solely due to additional training data.

We further investigate at a larger scale, *i.e.* use many more web action images as additional training data, where these action images are simply automatically crawled and without further annotation. We compare the performance of using BU101 and

[1] http://www.cs.bu.edu/groups/ivc/BU-action/

BU101-unfiltered images on UCF101. Using BU101-unfiltered we obtain similar performance to that obtained using BU101, even though collecting BU101-unfiltered requires much less human labor. We also obtain comparable performance when replacing half the training videos in ActivityNet (which correspond to 16.2M frames) by $\sim$ 400K images of BU203-unfiltered.

We then delve deeper and examine one major mechanism which may deliver the benefits of web action images in training the CNN models. We bring to light an artifact of finetuning a pre-trained CNN: *zombie filters* – CNN filters that undergo small changes during fine-tuning and make little, if any, contribution to the target task. These zombie filters reduce the number of effective parameters in the CNN model and are potentially harmful to the modeling capacity of the CNN. We illustrate that, by using web action images as additional training data, the number of zombie filters is greatly reduced, *e.g.* by an order of magnitude. This enables re-targeting more filters of the pre-trained CNN to visual patterns of the new task, *i.e.* action recognition.

Contributions: We make the following contributions in this chapter:

- We collect three large web action image datasets: BU101, BU101-unfiltered and BU203-unfiltered. These dataset are in one-to-one correspondence with the actions in the UCF101 or ActivityNet benchmark datasets.

- By extensive experimental evaluation, we verify the intuition that web action images are complementary to video training data. This complementarity appears to be insensitive to the CNN depth and is evident in many kinds of actions. Benefits are observed even when only a few filtered images are used in training and the benefits grow with number of web images.

- We illustrate that both filtered and unfiltered web action images are complementary to video training data. This points to an approach that requires little human annotation labor and is especially useful for large-scale problems.

- We show that using web action images can boost the efficiency of CNN training. With the same number of training samples, the trained model can achieve significantly higher recognition performance if half of the samples are web images. Moreover, to achieve the same recognition performance, we can greatly reduce the number of training videos and use unfiltered web action images instead.

- We provide insight into an artifact of finetuning a pre-trained CNN model for a new task: *zombie filters*. We show that, in our action recognition task, by using web action images as additional training data, the number of zombie filters can be significantly reduced. This reveals an underlying mechanism that brings in the benefits of web action images in the CNN model finetuning.

Roadmap for this chapter

We first present the new web action image datasets we collected in Sec. 6.1. Based on these datasets, in Sec. 6.2 we explore the utilization of web action images for training deep CNN models for action recognition in videos. We also analyse the zombie filters in Sec. 6.3 , a defect of finetuning pre-trained CNN models using video frames and web action images. In Sec. 6.4 we present extensive experiment evaluations of the proposed approaches on two large scale video dataset, UCF101 [44] and ActivityNet [28].

6.1 Web Action Image Dataset

To study the usefulness of web action images for learning better CNN models for action recognition, we collect action images that correspond with the 101 action classes in the UCF101 video dataset and the 203 activities in the ActivityNet dataset. This leads to 3 large image datasets: BU101, BU101-unfiltered and BU203-unfiltered. All three datasets will be made publicly available for research.

We collect images by crawling the web using the action class names of UCF101 or ActivityNet as queries on image search engines (Google Image Search, Flickr, etc.). Some queries are augmented by the words *exercise*, *train* and *play* when appropriate, *e.g.*, *juggling balls* to *play juggling balls*. BU101-unfiltered and BU203-unfiltered, containing 204K images and 387K images respectively, are simply compiled by this crawling procedure using action (activity) names of UCF101 (ActivityNet), without any further human annotation.

In the following text, we focus our discussion on the dataset BU101. For BU101, we inspect each crawled image and remove images that do not contain the action or are cartoons or drawings. We also include 2769 images of relevant actions from the Standford40 dataset [97]. The resulting dataset comprises 23.8K images. Because the images are automatically collected, and then filtered for irrelevant ones, the number of images per category varies. Each class has at least 100 images and most classes have 150-300 images. We will make our dataset publicly available for research.

Table 6.1 compares existing action image datasets with our new dataset, BU101. Both in the number of images and the number of actions, our dataset exceeds double the scale of existing datasets. More importantly, to the best of our knowledge, this is the first action image dataset that has one-to-one action class correspondence with a large-scale action recognition benchmark video dataset. We believe that our dataset

Table 6.1: Comparison of BU101 with existing action image datasets. *Visibility varies?* refers to variance in the partial visibility of the human bodies.

Dataset	# Actions	# Images	Clutter?	Poses vary?	Visibility varies?
Gupta [25]	6	300	Small	Small	No
Ikizler [33]	5	1727	Yes	Yes	Yes
VOC2012 [13]	11	4588	Yes	Yes	Yes
PPMI [96]	24	4800	Yes	Yes	No
Standford40 [97]	40	9532	Yes	Yes	Yes
Ours	**101**	**23800**	**Yes**	**Yes**	**Yes**

will enable further study of the relationship between action recognition in videos and in still images.

UCF101 action classes are divided into five types: *Human-Object Interaction*, *Body-Motion Only*, *Human-Human Interaction*, *Playing Musical Instruments*, and *Sports* [44]. Fig. 6.2 shows sample images in our dataset for five action classes, one in each of the five action types.

These action images collected from the Web are originally produced in a variety of settings, such as amateur *vs.* professional photos, artistic *vs.* educational *vs.* commercial photos, etc. For images collected in each action category, wide variation can exist in viewpoint, lighting, human pose, body part visibility, and background clutter. For example, commercial photos may have clear backgrounds while backgrounds of amateur photos may contain much more clutter. Such variance also differs for different types of actions. For example, for *Sports*, there is significant variance in body pose among images that capture different phases of the actions, whereas body pose variance is minimal in images of *Playing Musical Instruments*.

Many of the collected action images significantly differ from video frames in camera viewpoint, lighting, human pose, and background. One interesting thing to notice is that action images often capture *defining poses* of an action that are highly discriminative, *e.g.* standing with both hands over head and legs spread in *jumping jack*

Figure 6.2: Sample images from BU101. Each row shows images of one action. Top to bottom: *Hula Hoop, Jumping Jack, Salsa Spin, Drumming, Frisbee Catch.* Variations in background, camera viewpoint and body part visibility are common in web images of the same action.

(Fig. 6.2, row 2). In contrast, videos may have many frames containing poses that are common to many actions, *e.g.* in *jumping jack* the upright standing pose with hands down. Also, n images will have more unique content than n video frames, for example more clothing variation. Clearly there exists a compromise between temporal information available in videos and discriminative poses and variety of unique content in images.

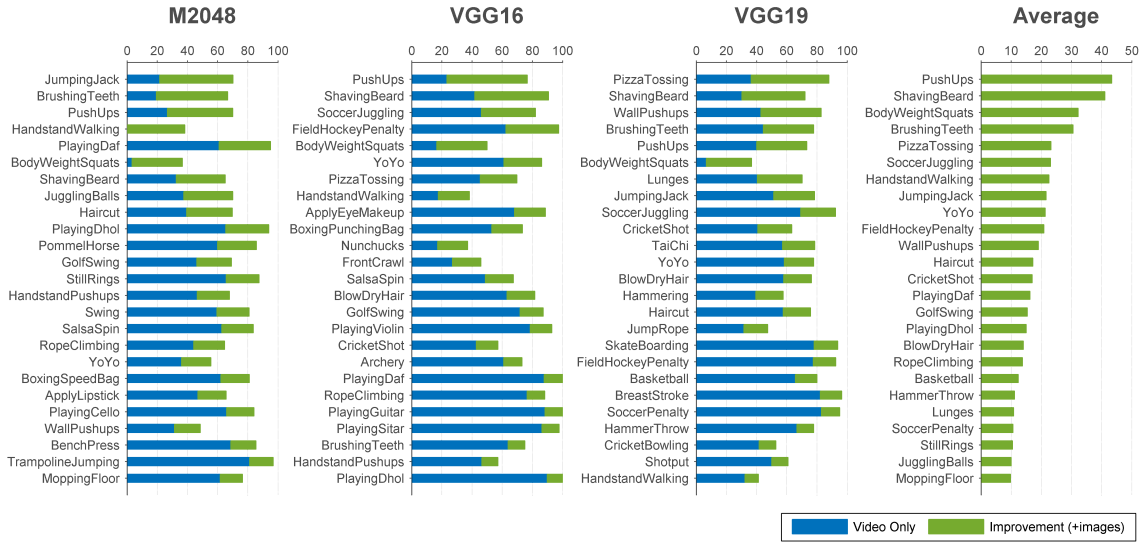

Figure 6.3: The 25 action classes with the largest accuracy improvement in the three CNN architectures as well as on average over the three architectures. The blue bars show the accuracy of CNN models trained only on videos. The green bars show the absolute increase in accuracy of CNN models trained using both web action images of BU101 and training videos.

6.2 Training CNNs with Web Action Images

Spatial CNNs trained on single video frames for action recognition are explored in [69]. Karpathy *et al.* [42] observe that spatio-temporal networks show similar performance compared to spatial models. A spatial CNN effectively classifies actions in individual video frames, and action classification for a video is accomplished via fusion of the spatial CNN's outputs over multiple frames, *e.g.* via voting or SVM. Because the spatial CNN is trained on single video frames, its parameters can be learned by fine-tuning of a CNN that was trained for a different task, *e.g.*, using a CNN that is pre-trained on ImageNet [11]. The fine-tuning approach is especially beneficial in training a CNN model for action classification in videos, since we often only have limited training samples; given the large number of parameters in a CNN, initializing the parameters to random values leads to overfitting and inferior

performance as shown in [69]. In this work, we study improving the spatial CNN for action recognition using web action images as training data in fine-tuning. This is then combined with motion features via state-of-the-art techniques.

In our experiments and analysis, we explore the following key questions:

- Is it beneficial to train CNNs with web action images in addition to video frames and, if so, which action classes benefit most?
- How do different CNN architectures, in particular ones with different depths, perform when web action images are used as additional training data?
- How do the performance gains change when more web action images are used in training the CNN?
- Are performance gains solely due to additional training data or also due to a single image being more informative than a randomly sampled video frame?
- Can we make the procedure of leveraging web images scalable by using crawled (unfiltered) web images rather than manually filtered ones?

We experiment on three CNN architectures: M2048 [7], VGG16, and VGG19 [70]. To avoid cluttering the discussion, implementation details are provided later in Sec. 6.4.

Is adding web images beneficial? Significant performance gains are achieved when we train spatial CNNs using our web action image dataset as auxiliary training data (see Table 6.2). For example, with the VGG19 CNN architecture, 5.7% absolute improvement in mean accuracy is achieved.

Most encouragingly, such improvements are easy to implement, without the need

Table 6.2: Accuracy on UCF101 split1 using three different CNN architectures.

Model	# Layers	# Parameters (in Millions)	Accuracy video only	Accuracy video + images
M2048	7	91	66.1%	**75.2%**
VGG16	16	138	77.8%	**83.5%**
VGG19	19	144	78.8%	**83.5%**

to introduce additional complexity to the CNN architecture and/or requiring significantly longer training time.

We further analyze which classes improve the most. Fig. 6.3 shows the 25 action classes for which the largest improvement in accuracy is achieved with the three different CNN architectures on UCF101 split1. The 25 action classes of top average accuracy improvement over all three tested architectures are also shown (rightmost column), all of which have no less than 10% absolute increase in accuracy and 10 classes have more than 20% absolute improvement. Some action classes are consistently improved irrespective of the CNN architecture used, such as *push ups*, *YoYo*, *handstand walking*, *brushing teeth*, *jumping jack*, etc. This suggests that utilizing web action images in CNN training is widely applicable.

While classification accuracy improvements in actions that are relatively *stationary* such as *Playing Daf* and *Brushing Teeth* are somewhat expected, it is interesting to see that improvements for actions of fast body motion such as *Jumping Jack* and *Body Weight Squats* are also significant.

Are images benefitial irrespective of CNN depth? While there are numerous ways that CNN architectures may differ from each other, here we focus on one of the most important factors. We evaluate the performance changes for CNNs of different depths when web action images of BU101 are used in addition to video frames in training. We train spatial CNNs of three depths: 7 layers (M2048), 16

layers (VGG16) and 19 layers (VGG19). These are the prototypical choices of CNN depths in recent works [7, 47, 52, 69, 70].

Table 6.2 shows the mean accuracy of the three CNN models trained *with* and *without* web action images of BU101 on UCF101 split1. Using web action images in training leads to a consistent $5\% \sim 9\%$ absolute improvement for all three architectures of different depths. This shows the usefulness of web action images and suggests a wide applicability of this approach. Furthermore, our results in action recognition confirm [70]'s observation that deeper CNNs of 16-19 layers significantly outperform the shallower 7-layer architecture. However, the margin of performance gain diminishes when we increase the depth from 16 to 19.

Does adding more web images improve accuracy? We further explore how, for the same CNN architecture, the number of web action images used as additional training data can influence the classification accuracy of the resulting CNN model. We sample 1/10, 1/5, 1/3 and 2/3 of the images of each action in our dataset, and for each sampled set we train the spatial CNN by fine-tuning VGG16 using both the training videos and sampled action images. For each sample size, we repeat the experiment three times, each with a different randomly sampled set of web action images. The evaluation is performed on UCF101 split1.

Fig. 6.4 summarizes the results of this experiment. The increase in classification accuracy is most significant at the beginning of the curve, *i.e.* when a few thousand web action images are used in training. This increase continues as more web action images are used, even though the increase becomes slower. Firstly, this indicates that using web action images in training can make a significant difference in performance by providing additional supervision to that provided by video frames. Secondly, it indicates that it is good practice to collect a moderate number of web action images

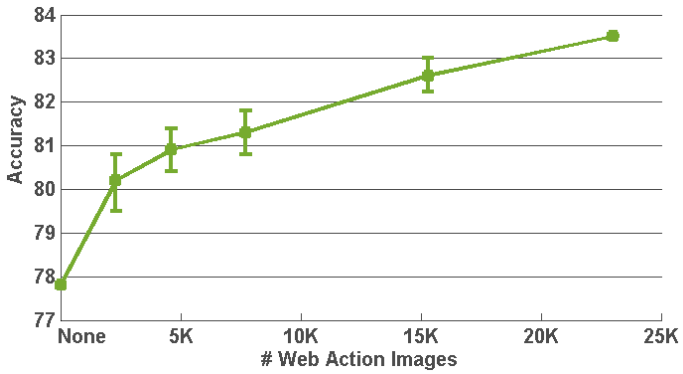

Figure 6.4: Performance of the spatial CNNs (VGG16) trained on UCF101 split1 using different numbers of web action images of BU101 as additional training data.

for each action as a cost-effective way to boost model performance (*e.g.*, 100 $\sim$ 300 images per action for a dataset of the same scale as UCF101).

Do web images complement video frames? Although augmenting with images is more efficient than augmenting with videos, we further investigate whether the achieved performance gains are solely due to additional training data or whether a web image provides more information to the learning algorithm than a video frame. This is done by replacing video frames by web images, keeping the total number of training samples constant. For each sample size, we repeat the experiment three times, each with a different randomly sampled set of web action images. The evaluation is performed on UCF101 split1 and a VGG16 model.

Fig. 6.5 summarizes the results of this experiment. A consistent improvement in performance is achieved when half the video frames are replaced by web images. The number of training samples (images and video frames) required to obtain the maximum accuracy presented in Fig. 6.4 is much less (50K *vs.* 230K). This suggests that images are augmenting the information learnt by the classifier. We posit that discriminative poses in action images may provide implicit supervision, in training, to help learn better discriminative models for classification.

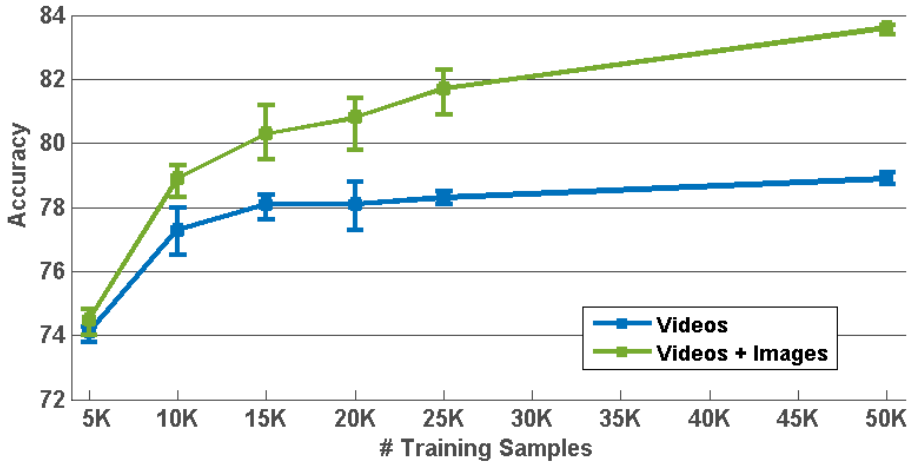

Figure 6.5: Performance of the spatial CNNs (VGG16) trained on UCF101 split1 using video frames only and replacing 50% of the video frames by web images of BU101.

Can this be made scalable? While we have demonstrated the ability to collect a filtered dataset for our desired classes, this is not scalable. Given a different dataset having the same order of magnitude as UCF101 we would have to manually label a dataset for its classes. Given an even larger dataset with more classes and more samples per class, this becomes very cumbersome although still better than collecting videos. We now investigate the possibility of using crawled (unfiltered) web images for the same purpose, utilizing BU101-unfiltered.

Table 6.3 summarizes the results of this experiment. The performance of using unfiltered images approaches that of manually filtered images, but the number of web images utilized is much larger. We further investigate whether *all* the crawled unfiltered images are required to obtain such performance. We do this by randomly selecting one quarter (65.5K) of the 207K unfiltered web images. We select 3 random samples and report the average result in Table 6.3. Three quarters of the images only contribute with an additional accuracy of 1%; this is consistent with Fig. 6.4 observations.

Table 6.3: Accuracy on UCF101 split1 using spatial CNN (VGG16) of manually filtered and unfiltered web images.
* means average of three random sample sets.

Image Type	# Images	Accuracy (%)
Manually filtered	23.8K	83.5
Unfiltered (all)	207K	83.1
Unfiltered (rand select)	65.5K	82.1*

Having demonstrated the feasibility of using crawled web images, we now apply this to a larger-scale dataset: ActivityNet [28] using BU203-unfiltered. ActivityNet contains more classes (203) and more samples per class than UCF101. ActivityNet classes are more diverse; they belong to the categories: *Personal Care, Eating and Drinking, Household, Caring and Helping, Working, Socializing and Leisure, and Sports and Exercises.* "ActivityNet provides samples from 203 activity classes with an average of 137 untrimmed videos per class and 1.41 activity instances per video, for a total of 849 video hours." [28] Mostly, videos have a duration between 5 and 10 minutes and have a 30 FPS frame rate. About 50% of the videos are in HD resolution. Results on ActivityNet are reported in Section 5.

6.3 Caution: "Zombies" Around

For tasks that have limited training data, training a deep CNN by *fine-tuning* from a CNN pre-trained on a large-scale dataset (but for a different task) is an important and popular technique [21, 69]. In such an approach, most parameters (usually all but the last layer) of the target model are initialized to the parameter values of the pre-trained model. This initialization usually works significantly better than random parameter initialization. In this work we train the spatial CNNs for action recognition by fine-tuning models originally trained on ImageNet for object recognition. Despite

the benefits of the fine-tuning technique, we dive deeper and provide insight to its downside–what we call **zombie filters**: the filters whose parameters do not change during fine-tuning *and* do not contribute to the new task. These zombie filters reduce the effective number of parameters in the CNN and may be harmful to the CNN's modelling capacity.

In typical deep CNNs [47, 70], the ReLU (Rectified-Linear Unit) layer follows the convolutional layers or fully connect layers and introduces nonlinearity to the model. Denote $\mathbf{w}_n^k$ as the kth convolution filter in the nth convolutional layer. Its ith output can be simply written as a dot product $x_n^{k,i} = \mathbf{w}_n^k \cdot \mathbf{x}_{n-1}^i$ where $\mathbf{x}_{n-1}^i$ is the part of the input to the nth layer that participates in the ith convolution. Now suppose there is a ReLU layer following this layer, and its ith output on the kth input channel can be denoted as $r_n^{k,i} = max(0, x_n^{k,i})$. During training, in back propagation, the gradients of training loss with respect to $\mathbf{w}_n^k$ are

$$\frac{\partial L}{\partial \mathbf{w}_n^k} = \sum_i \frac{\partial L}{\partial x_n^{k,i}} \frac{\partial x_n^{k,i}}{\partial \mathbf{w}_n^k} = \sum_i \frac{\partial L}{\partial r_n^{k,i}} \frac{\partial r_n^{k,i}}{\partial x_n^{k,i}} \cdot \mathbf{x}_{n-1}^i. \tag{6.1}$$

Notice that $\partial r_n^{k,i} / \partial x_n^{k,i} = 0$ when $x_n^{k,i} < 0$ and otherwise equal to 1, so Eq. 6.1 can be written as

$$\frac{\partial L}{\partial \mathbf{w}_n^k} = \sum_{i:\ x_n^{k,i} \geq 0} \frac{\partial L}{\partial r_n^{k,i}} \cdot \mathbf{x}_{n-1}^i. \tag{6.2}$$

Thus $\partial L / \partial \mathbf{w}_n^k$ is determined by the non-negative convolution outputs, *i.e.* the set $X_n^{k,+} = \{x_n^{k,i} | x_n^{k,i} \geq 0,\ 1 \leq i \leq N\}$ (N: the total number of convolutions by $\mathbf{w}_n^k$ in the nth layer). In training a CNN, typically the following weight update is used in backpropagation in the tth iteration:

$$\mathbf{v}_{n,t}^k = \mu \cdot \mathbf{v}_{n,t-1}^k - \delta \cdot \epsilon \cdot \mathbf{w}_{n,t-1}^k - \epsilon \cdot \left\langle \frac{\partial L}{\partial \mathbf{w}_n^k} \right\rangle_{D_t}, \tag{6.3}$$

$$\mathbf{w}_{n,t}^{k} = \mathbf{w}_{n,t-1}^{k} + \mathbf{v}_{n,t}^{k}. \tag{6.4}$$

where $\mathbf{v}_{n,t}^{k}$ is the momentum variable, μ is the momentum coefficient, δ is the weight decay coefficient and ϵ is the learning rate. Typical choices for μ and δ are 0.9 and 0.0005 respectively. The learning rate ϵ is usually small for fine-tuning, *e.g.* initialized to 10^{-3} and further reduced during training in our experiments. $\left\langle \frac{\partial L}{\partial \mathbf{w}_n^k} \right\rangle_{D_t}$ represents the average gradient over the training batch D_t. If $\mathbf{w}_n^k$ produces negative outputs for most samples in D_t, $\left\langle \frac{\partial L}{\partial \mathbf{w}_n^k} \right\rangle_{D_t}$ is very likely to be very small, so the update to $\mathbf{w}_n^k$ will be mainly due to weight decay, which is usually small for the small weight decay coefficient and learning rate. In the situation of fine-tuning a pre-trained model, let $\mathbf{w}_n^k$ represent the parameter values of the filter in the pre-trained model and $\hat{\mathbf{w}}_n^k$ the parameter values after fine-tuning. If $X_n^{k,+}$ is empty or small most of the time in training, the difference of $\hat{\mathbf{w}}_n^k$ with $\mathbf{w}_n^k$ is likely to be small too, *i.e.* $\hat{\mathbf{w}}_n^k \approx \mathbf{w}_n^k$.

This situation is possible, especially when the training data of target task (*target data*) differs significantly from the data used for pre-training (*source data*). Some filters in the pre-trained model may have learned some visual patterns in the source data that rarely appear in the target data, so that in fine-tuning their outputs may tend to be negative most of the time in the forward passes and receive very small updates in the backward passes, which can make them *stale* in the training, *i.e.* becoming *zombies*. Clearly, these zombies will make small, if any, contributions to the target task. Also, note that each unit of a fully connected layer can be seen as a filter with the size equal to its whole input, so the discussion above also holds for the fully-connected layers.

We investigate zombie filters in the VGG16 model that is fine-tuned using only video frames of the training videos of UCF101 split1. For the investigation, we

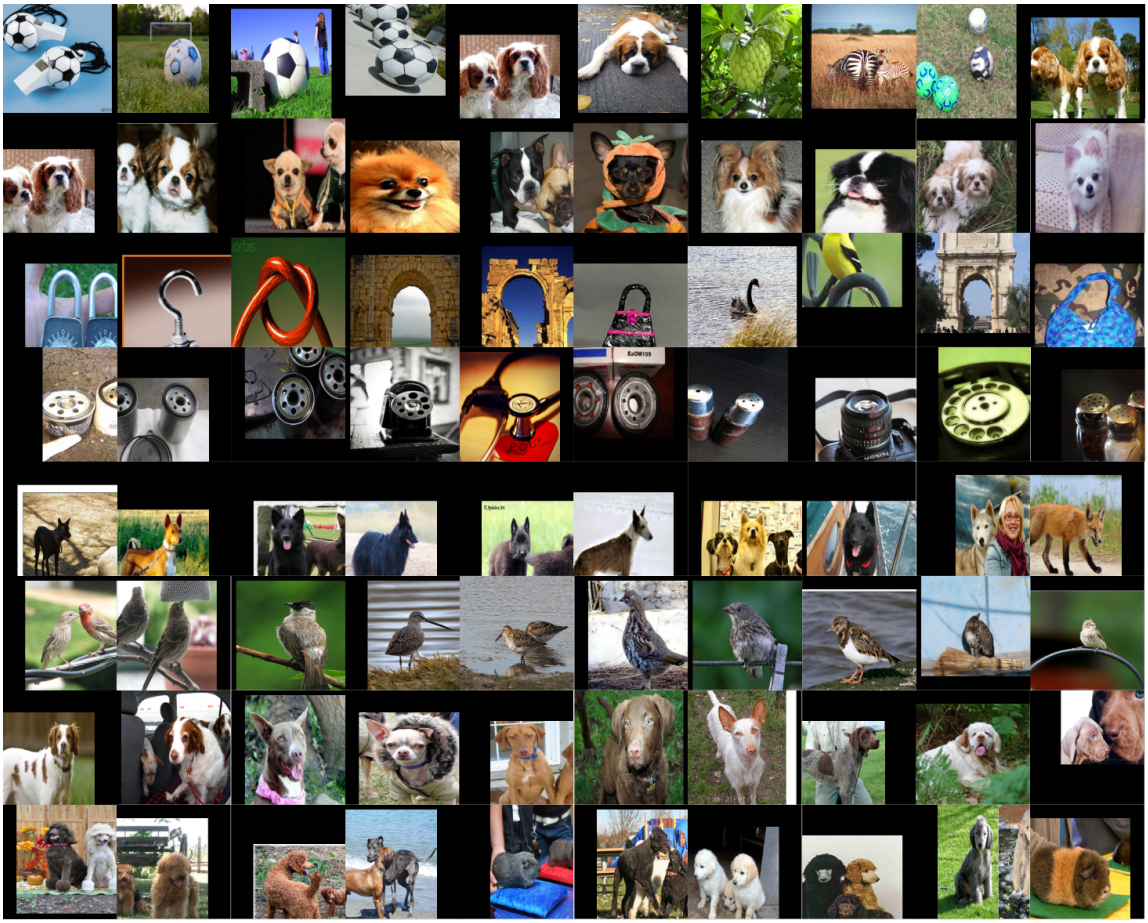

Figure 6.6: Top activations for 8 example zombie filters. Each row shows top 10 activations of one zombie in the last convolutional layer (conv5-3) of the VGG16 model that is fine-tuned using only video frames of the training videos in UCF101 split1.

compile an image pool containing 50K images from ImageNet and 55K video frames (*i.e.* 0.5K video frames per action class). All images are re-sized to 224×224, which is the input size of the CNNs we fine-tuned. For a filter $\hat{\mathbf{w}}_n^k$ in the fine-tuned CNN, we find its maximum activation for each image in the pool. For example, for a filter in the 13th convolutional layer (*conv5-3*), its receptive field is 211×211, so for each image in the pool we find an image patch of 211×211 that causes maximum activation. For the kth filter in the nth layer, we sort the images in descending order of their maximum activation value for this filter, and then compute the percentage of the

images among the top 100 that are video frames (denoted as α_n^k). We also compare the filter's parameter change against the original pre-trained model, measured by $\Delta_n^k = \|\hat{\mathbf{w}}_n^k - \mathbf{w}_n^k\|$. We then look for zombie filters that have both small α and Δ, *i.e.* filters that stay almost the same during fine-tuning and whose top activations are mostly ImageNet images. In Fig. 6.6 we depict the top 10 activations of the 8filters from *conv5-3* that result from the intersection of the 30 filters of the least α and the 30 filters of the least Δ. Even though our image pool has almost equal numbers of images from ImageNet and UCF101 video frames, all these top activations come from ImageNet and correspond to dogs, objects, architecture *etc.*, which are indeed rare in UCF101.

We believe that zombie filters essentially take up parameters in the CNN that otherwise could have been used for learning visual patterns in the target task. Fine-tuning from a pre-trained model provides good initial values for some of the parameters, but may set some parameters to bad local minima with respect to the target task. However, this situation can be significantly improved by utilizing web action images as additional training data. We compare the number of zombie filters in the *conv5-2*, *conv5-3* and *fc6* layers in the VGG16 model fine-tuned only with training video frames of UCF101 split1 and the VGG16 model fine-tuned with both these training video frames and the web action images in BU101. Table 6.4 shows this comparison: for each model and layer, we list the number of filters whose $\alpha < 0.1$, *i.e.* less than 10% of the top 100 activation images in our image pool are video frames, and whose Δ is small, *e.g.* less than 0.05. We can observe a significant reduction of the number of zombie filters when fine-tuning using both video frames and web action images. For example, for layer *conv5-2*, the number of filters with $\alpha < 0.1$ and $\Delta < 0.06$ is reduced from 87 to 8 when web action images are used as additional

Table 6.4: Using web action images can significantly reduce the number of zombie filters. For the VGG16 model finetuned with only training video frames of UCF101 split1 (noted as *video only* in the table) and the VGG16 model finetuned with both video frames and images of BU101 (noted as *video + image*), we compute the number of filters which satisfy: 1) $\alpha < 0.1$, *i.e.* only less than 10% of top 100 activation images in our image pool are video frames; 2) Δ is less than a small value (given in the table), *i.e.* doesn't change much during finetuning. Notice the large reduction in the numbers of such filters when using web action images as additional training data.

		$\Delta < 0.02$	$\Delta < 0.03$	$\Delta < 0.04$	$\Delta < 0.05$	$\Delta < 0.06$
conv5-2	*video only*	0	2	10	30	87
	video + image	0	0	0	2	8
conv5-3	*video only*	0	13	33	82	123
	video + image	0	1	4	20	38
fc6	*video only*	12	84	212	399	636
	video + image	1	24	63	121	185

training data, which is more than an order-of-magnitude reduction. Recall that the action classification accuracy of the VGG16 model finetuned only with video frames is 77.8%, and the VGG16 model finetuned with both video frames and web images of BU101 is 83.5% (Table 6.2): the absolute improvement after using web action images is 5.7%. We posit that the reduction of zombie filters may be an important reason for this performance improvement: web action images may help reduce the number of zombie filters so that their parameters could be *re-used* in learning visual patterns for the new task, *i.e.* action recognition in videos.

6.4 Experiments

Using insights from the experiments performed on UCF101 split1 in Section 4, we now perform experiments following the standard evaluation protocol [40] and report the average accuracy over the three provided splits.

We also perform experiments on ActivityNet. Following [28], we evaluate clas-

sification performance on both trimmed and untrimmed videos. Trimmed videos contain exactly one activity. Untrimmed videos contain one or more activities. We use the mAP (mean average precision) in evaluating performance. Results reported on ActivityNet are produced using the validation data, as the authors are reserving the test data for a potential future challenge.

6.4.1 Implementation

6.4.1.1 Experimental Setup for UCF101

Fine-tuning: We use the Caffe [39] software for fine-tuning CNNs. We use models VGG16, VGG19 [70], and M2048 [7] that are pre-trained on ImageNet by the corresponding authors. We only test M2048 on the first split for analysis, as it is shown to be significantly inferior to the other two architectures (Table 6.2). Due to hardware limitations, we use a small batch size: 20 for M2048 and 8 for VGG16 and VGG19. Accordingly, we use a smaller learning rate than those used in [7, 70]. For M2048, the initial learning rate 10^{-3} is changed to 10^{-4} after 40K iterations; training stops at 80K iterations. For both VGG16 and VGG19, the initial learning rate 10^{-4} is changed to 10^{-5} after 40K iterations, and is further lowered to 2×10^{-6} after 80K iterations. Training stops at 100K iterations. Momentum and weight decay coefficients are always set to 0.9 and 5×10^{-4}. In each model, all layers are fine-tuned except the last fully connected layer which has to be changed to produce output of 101 dimensions with initial parameter values sampled from a zero-mean Gaussian distribution with $\sigma = 0.01$.

We resize video frames to 256×256, and random crops to 224×224 with random horizontal flipping for training. For web action images, since their aspect ratios vary significantly, we first resize the short dimension to 256 while keeping the aspect

ratio, and subsequently crop six 256×256 patches along the longer dimension in equal spacing. Random cropping of 224×224 with random horizontal flipping is further applied to these image patches in training. Equal numbers of web images and video frames are sampled in each training batch.

Video Classification: A video is classified by fusing over the CNN outputs for the individual video frames. For a test video, we select 20 frames of equal temporal spacing. From each of the frames, 10 samples are generated following [47]: four corners and the center (each is 224×224) are first cropped from the 256×256 frame, making 5 samples; horizontal flipping of these samples makes another 5. Their classification scores are averaged to produce the frame's scores. We classify each frame to the class of the highest score, and the class of the video is then determined by voting of the frames' classes.

We also test SVM fusion, concatenating the CNN outputs for the 20 frames (averaged over the 10 cropped and flipped samples) from the second fully-connected layer (fc7), *i.e.* the 15th layer in VGG16 and 18th layer in VGG19. This produces a vector of 81,920 (4096×20) dimensions, which is then L2 normalized. One-vs-rest linear SVMs are then trained on these features for video classification. The SVM parameter $C = 1$ in all experiments.

Combining with Motion Features: The output of spatial CNNs can be combined with motion features to achieve significantly better performance, as shown in [69]. We present an alternative by combining the output of the spatial CNNs with the conventional expert-designed features, namely the improved dense trajectories with Fisher encoding (IDT-FV) [80]. We follow the same settings in [80] to compute the IDT-FV for each video except that we do not use a space-time pyramid. The IDT-FV of each video is then combined with the concatenated fc7 outputs of 20 frames to

Table 6.5: Mean accuracy of spatial CNNs (averaged over three splits) on UCF101.

Model	Accuracy (%)
slow fusion CNN [42]	65.4
spatial CNN [69]	73.0
VGG16, voting	77.9
VGG16 + Images, voting	82.5
VGG16 + Images, SVM fusion on fc7	**83.5**
VGG19, voting	77.8
VGG19 + Images, voting	83.3
VGG19 + Images, SVM fusion on fc7	83.4

Table 6.6: Mean accuracy (averaged over three splits) when combining spatial CNNs with motion features for UCF101.

Model	Accuracy (%)
IDT-FV [80]	85.9
Two-stream CNN [69]	88.0
RCNN using LSTM [55]	88.6
VGG16 + Images + IDT-FV	**91.1**
VGG19 + Images + IDT-FV	90.8

form the final feature vector for a video. One-vs-rest linear SVMs are then trained on these features for video classification. The SVM parameter $C = 1$.

6.4.1.2 Experimental Setup for ActivityNet

We use the Caffe [39] software for fine-tuning CNNs. We use a VGG19 model [70] that is pre-trained on ImageNet by the authors. Due to hardware limitations, we use a small batch size of 8. Accordingly, we use a smaller learning rate than [70]. The initial learning rate 10^{-4} is changed to 10^{-5} after 80K iterations. Training stops at 160K iterations. Momentum and weight decay coefficients are set to 0.9 and 5×10^{-4}. All layers are fine-tuned except the last fully connected layer which has to be changed to produce output of 203 dimensions with initial parameter values sampled from a zero-mean Gaussian distribution with $\sigma = 0.01$.

Table 6.7: Although ActivityNet is large-scale, using unfiltered web images still helps in both trimmed and untrimmed classification (results are averaged over three random sample sets.).

Model	# Images	Untrimmed mAP (%)	Trimmed mAP (%)
fc8 [28]	none	25.3	38.1
DF [28]	none	28.9	43.7
Ours (video frames only)	none	52.3	47.7
Ours (unfiltered: all)	393K	53.8	49.5
Ours (unfiltered: rand select)	103K	53.3*	49.3*

Resizing and cropping of images and frames are performed in the same way as previously described for UCF101. Samples in each training batch are randomly selected from web action images and video frames with equal probability.

6.4.2 Results

6.4.2.1 Experimental Results for UCF101

Here we report the performance of our spatial CNNs averaged over three splits of UCF101 (Table 6.5), as well as the performance of our models when motion features are also used (Table 6.6).

As seen in Table 6.5, all our spatial CNNs trained using both videos and images improved ∼10% (absolute) in accuracy over the spatial CNN of [69], which is a 7-layer model. We believe this improvement is due to two main factors: using a deeper model and using web action images in training. Comparing the performance of the spatial CNN of [69] to the deeper models trained only on videos (rows 3 and 6 in Table 6.5), we find that the improvements solely due to differences of CNN architectures are 4.9% and 4.8% for VGG16 and VGG19 respectively. When web action images are used in addition to videos in training (rows 4 and 7 in Table 6.5), these improvements are doubled: 9.5% and 10.3% respectively.

Results reported in Table 6.5 show that, in the models we tested, the simple approach of using web action images in training contributes at least equally with introducing significant complexities to the CNN model, *i.e.*, adding at least 9 more layers. It is also interesting to note that, without using optical flow data, our spatial CNNs already approach performance attained using state-of-the-art expert designed features that use optical flow, *i.e.* IDT-FV [80] in Table 6.6. Performance gains obtained by our approach are especially encouraging compared to deepening the model or incorporating motion features, as leveraging web images during training will not add any additional computational or memory burden during test time.

The slow fusion CNN [42] is not a spatial CNN as it is trained on multiple video frames instead of single video frames. We list it here as it presents a different approach; collecting millions of web videos for training. However, despite the fact that 1M web videos are used as pre-training data, its performance is far lower than our models.

We further test the features learned by our spatial CNNs when combined with motion features, *i.e.* Fisher encoding on improved dense trajectories. Table 6.6 compares our results with state-of-the-art methods that also use motion features. Our method (VGG16 + Images + IDT-FV) outperforms all, improving by 2.5% over [55] that trains recurrent CNNs with long short-term memory cells; by 3.1% over [69], which combines two separate CNNs trained on video frames and optical flow respectively; and by 5.2% over [80] that uses Fisher encoding on improved dense trajectories.

Table 6.8: Comparable performance is achieved when half the training videos of ActivityNet are replaced by 393K images (row 4 *vs.* row; results are averaged over three random sample sets.

Experiment	# Frames	# Images	mAP (%)
All vids	32.3M	none	47.7
1/2 vids	16.2M	none	40.9*
1/4 vids	8.1M	none	33.4*
1/2 vids + imgs	16.2M	387K	46.3*
1/4 vids + imgs	8.1M	387K	41.7*

6.4.2.2 Experimental Results for ActivityNet

Here we report the performance of our spatial CNNs on ActivityNet for the task of action classification in trimmed and untrimmed videos with and without auxiliary web images of BU203-unfiltered (Table 6.7). We then further investigate the use of web images as a substitute for many training videos (Table 6.8).

In Table 6.7 we observe that utilizing web images still helps ∼1.5% even with a very large scale dataset like ActivityNet. Using a random sample of approximately one quarter of the crawled web images gives nearly the same results, suggesting that performance gains diminish as the number of web action images greatly increase. This result is consistent with results on UCF101 (Figure 6.4).

In Table 6.8 we observe that comparable performance is achieved when half the training videos, are replaced by web images (rows 1 and 4 in Table 6.8). A similar pattern is observed when repeating the experiment at a smaller scale. This suggests that using a relatively small number of web images can help us reduce the effort of curating and storing millions of video frames for training.

6.5 Summary

In this chapter we study the benefits of web images that are in one-to-one action category correspondence with training videos. We collect three datasets of web action images: the BU101 *filtered* dataset and two *unfiltered* datasets, BU101-unfiltered and BU203-unfiltered. We show that utilizing web action images in training CNN models for action recognition is an effective and low-cost approach to improve performance.

While videos contain a lot of useful temporal information to describe an action, and while it is more beneficial to use videos only than to use web images only, web images can provide *complementary* information to a finite set of videos allowing for a significant reduction in the video data required for training. We observe that this complementarity is insensitive to different CNN architectures and is evident in many kinds of actions. Both filtered and unfiltered web action images are complementary to video training data. However, human filtering of the web action images is still useful: considerably fewer filtered images are required to achieve similar performance improvements.

Using web action images can also boost the efficiency of CNN training. We show that when using the same number of training samples, the trained model can achieve significantly higher recognition performance if half of the samples are web images. We also show that, to achieve the same recognition performance, we can greatly reduce the number of training videos and use unfiltered web action images instead.

For CNN finetuning, using web action images as training data in addition to training video frames can greatly reduce the number of zombie filters, *i.e.* CNN filters that undergo minimal changes and have low activation on video frames. Such zombie filters reduce the effective number of parameters in the CNN model and thus may be harmful for its modeling capacity. We speculate that one underlying mechanism

that delivers the benefits of web action images, as shown in our experiments, is that the reduction of zombie filters enables *re-use* of those parameters for modeling visual patterns of the new task, which is action recognition in our application.

Chapter 7

Conclusions and Future Work

In this last chapter, we first summarize the key contributions of this thesis: a new video representation, the Hierarchical Space-Time Segments, designed for effective action recognition and localization; a new action model, the Ensemble of Space-Time Trees, which models spatial, temporal and hierarchical structures in human actions for improved action recognition and localization performance; two novel ranking losses designed for training deep LSTM models that better learns the temporal progression of actions for action detection and early detection; and, finally interesting findings of the utilization of web action images in training CNN models for action recognition in videos. We then describe the major strengths and limitations of our work. Finally, we point out some interesting directions for future research.

7.1 Main Contributions

In this thesis we focus on the problem of action recognition and localization in videos. Specifically, when the video is temporally trimmed so that the starting and ending frames are the starting and ending points of an action, we want to recognize the action class and also estimate the spatial location of the action performer(s) in each video frame. When the video is not temporally trimmed, we want to recognize the action and also estimate its starting and ending time points in the video. We argue

that human actions are by nature structured patterns. Thus, for both problems, we propose methods that explore such structures for better action recognition and localization.

For the first problem, we first propose a novel space-time representation of video, the Hierarchical Space-Time Segments, that preserves the hierarchical relationships of extracted local space-time segments of videos. Unlike previous local space-time representations of videos such as STIP [50] and dense trajectories [77] that mainly focus on non-static parts of video, HSTSs cover both non-static and relevant static foregrounds of a video. Consequently, HSTSs better capture pose information for recognizing actions and can be used to more accurately localize the whole visible part of action performer(s). One significant benefit of this new representation is that, for the action spatial localization task, we do not need human bounding box annotations in training but only need action labels of the training videos.

Based on the HSTSs, we further learn space, time and hierarchical structures of human actions by an action model, the Ensemble of Space-Time Trees. After extracting the HSTSs from a video, we consider the spatial, temporal and hierarchical relationships among the HSTSs, by converting each video to a graph. From the graphs of training videos, we discover discriminative tree structures of actions and construct action classifiers as linear ensembles of the discovered trees. Subsequently, recognizing and localizing actions in a test video is converted to matching the trees of the ensemble to the graph representation of the testing video. We show strong experimental results of our approach on three challenging benchmark datasets. We also show cross-dataset generalizability by using the trees discovered in one dataset to successfully recognize and localize similar actions in another dataset.

For the second problem, we use a deep model that learns temporal structures of

actions for better recognizing and temporally localizing the actions. This deep model can be used to handle large scale datasets. Specially, the model learns temporal dynamics captured by a LSTM model and visual features of video frames discovered by a CNN model. We propose two novel ranking losses that are used in model training to improve the LSTM model's ability of learning the temporal structure of actions. One of the ranking losses enforces that the model outputs a non-decreasing detection score as the current training action continues. The other ranking loss enforces that, during the current training action sequence, the model's outputs should have a non-decreasing discriminative margin between the correct and incorrect action class. Intuitively, these ranking losses encourage the model to be more confident of predicting the correct action class and denying the incorrect action class as the model *sees* more of the action. Our model, trained with the proposed ranking losses, achieves state-of-the-art action detection and early detection performance on a large scale video dataset.

We also note that our deep model, as well as many deep models proposed by other researchers for action recognition, uses a deep CNN model for learning visual features from video frames. Deep CNN architecture typically has millions of parameters and requires a large number of training videos to train a good model. We study the utility of web action images for training CNNs for action recognition. Our main findings can be summarized as follows:

- Web action images are complementary to video training data. This complementarity is insensitive to the depth of CNNs and is evident in many kinds of actions. Benefits are observed even when only a few filtered images are used and grow with number of web images.

- Both filtered and unfiltered web action images are complementary to video

training data. However, human filtering of the web action images is still useful: much fewer filtered images are required to achieve similar performance improvements.

- Using web action images can boost the efficiency of CNN training: 1.) with the same number of training samples, the trained model can achieve significantly higher recognition performance if half of the samples are web images; 2.) to achieve the same recognition performance, we can greatly reduce the number of training videos and use unfiltered web action images instead.

7.2 Strength and Limitations

As a local space-time representation of video, the HSTS representation capture both static and non-static relevant foreground of the video, which serves for better action recognition and spatial localization. It also preserves the hierarchical relationships of the space-time segments, which facilitates learning hierarchical structures of human actions. The major limitation is that this approach requires computing dense optical flow and hierarchical image segmentation on each video frame, which is computationally demanding, so it falls short in handling long videos and large-scale datasets.

Our novel action model, the ensemble of space-time trees, explicitly models the hierarchical, spatial and temporal structures of human actions, while the tree structures are all discovered from the training data. Using this model, action recognition and localization is done by simply matching the tree structures to a graph representation of a video and, in training, human bounding box annotations are not needed. The drawback of this approach is that our tree discovery procedure requires multiple steps and does not guarantee discovery of the most discriminative tree structures.

For action detection and early detection in large scale datasets, our LSTM model

trained with novel ranking losses effectively learns the temporal progression of actions and performs well on a large benchmark dataset. The proposed ranking losses are simple to implement, introduce small additional computation in training and no extra computation in testing. Our model can handle long videos containing multiple actions, and can detect an action even when only one tenth of the action is seen. When running on a model GPU with scientific computation ability, action detection can be done in real time using this model. The main defect of this model is that it does not have a explicit mechanism to model actions of significantly different temporal granularities, *e.g.* while *preparing pasta* may last more than several minutes, *clean and jerk* may finish in a few seconds.

Our findings concerning utilization of web action images for improving CNN training for action recognition in videos illuminate a simple way to significantly reduce the burden of collection video training data. However, our approach of utilizing these web action images is quite naive: simply using them as additional training data. Lacking a more innovative way of using these web action images is a limitation of this study.

7.3 Interesting Directions for Future Research

As described in Sec. 7.2, there are limitations of the works in this thesis. These open interesting research directions for future research.

One interesting direction is to explore a more efficient approach to extract the HSTSs from videos so that they can be used to handle longer videos and larger datasets. Note that we use dense optical flow only for producing motion maps for segmentation, and we do hierarchical segmentation of the whole frame only to extract the foreground segments. Thus, in a future study, we may want to figure out how

to avoid exact computation of dense optical flows and hierarchical segmentation and only compute the information we really need.

A more principled way of space-time tree discovery from graphs of training videos is worthwhile for future study. It will be highly desirable if the pipeline of the tree discovery can be simplified when we acquire a deeper understanding of the tree structure discovery problem. Such a shortened procedure may be easier to analyze for the tree discovery performance, and we may be able to provide some kind of guarantee of the optimality of the tree discovery for the action recognition and localization problem.

Deep LSTM formulations with the ability to better model multiple temporal granularities of human actions worth exploring in future study. Such a formulation would be useful for at least two reasons: first, different actions may have significantly different temporal durations such as *preparing pasta* and *clean and jerk*; second, a long action may contain multiple relatively short sub-actions, *e.g.* *cutting vegetable* in *preparing pasta*, and in certain applications it may be useful to also detect such sub-actions;

It would also be interesting to study innovative way to use web action images for training CNNs. One potential direction is to investigate domain adaptation methods for deep models: the web action images and video frames are images of different domains, and they have many differences such as composition, lighting and etc. It will be useful to explore ways to train a CNN with both web action images and video frames so that the training model learns the common semantic concepts, *i.e.* human actions, in these images but ignores their domain differences.

Note that our deep LSTM model can perform action detection in untrimmed videos; however, it only localizes actions in time but not in space. An interesting

topic for future study would be to extend the approaches in this thesis to solve the combined task of spatial and temporal action localization in untrimmed videos. First, if we improve the HSTS formulation as mentioned above so that it can be used to handle longer videos, then we may extend our ensemble of space-time trees for both spatial and temporal localization in untrimmed videos. In principle, there is no limitation in our ensemble of space-time tree formulation to handle temporal localization and the major practical barrier is in the computation of HSTSs in untrimmed videos which may be long. Second, our deep LSTM model may be extended to handle action localization in both space and time for untrimmed videos. One potential idea is that, at every frame, besides the CNN feature, the spatial location information of the feature may also be used as input to the LSTM model, so that the model has information to detect the spatial location of the actions. For example, one can use an R-CNN [21] model to replace the current CNN model to detect candidate regions of humans and compute their CNN features. The LSTM model can then take as input the CNN features of these candidate regions and their spatial locations to learn space-time dynamics of the actions.

Bibliography

[1] The Gaston tool for frequent subgraph mining. *Electronic Notes in Theoretical Computer Science 127*, 1 (2005), 77 – 87.

[2] Aoun, N. B., Mejdoub, M., and Amar, C. B. Graph-based approach for human action recognition using spatio-temporal features. *Journal of Visual Communication and Image Representation 25*, 2 (2014), 329 – 338.

[3] Arbelaez, P., Maire, M., Fowlkes, C. C., and Malik, J. From contours to regions: An empirical evaluation. In *Conference on Computer Vision and Pattern Recognition* (2009).

[4] Bobick, A. F., and Davis, J. W. The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*, 3 (2001), 257–267.

[5] Brendel, W., and Todorovic, S. Learning spatiotemporal graphs of human activities. In *IEEE International Conference on Computer Vision* (2011).

[6] Caba Heilbron, F., Escorcia, V., Ghanem, B., and Carlos Niebles, J. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 961–970.

[7] Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A. Return of the devil in the details: Delving deep into convolutional nets. *British Machine Vision Conference* (2014).

[8] Cheáron, G., Laptev, I., and Schmid, C. P-CNN: pose-based CNN features for action recognition. In *IEEE International Conference on Computer Vision* (2015).

[9] Chen, C.-Y., and Grauman, K. Watching unlabeled video helps learn new human actions from very few labeled snapshots. In *Conference on Computer Vision and Pattern Recognition* (2013).

[10] Crammer, K., and Singer, Y. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research 2* (2001), 265–292.

[11] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition* (2009).

[12] Duan, L., Xu, D., and Chang, S.-F. Exploiting web images for event recognition in consumer videos: A multiple source domain adaptation approach. In *Conference on Computer Vision and Pattern Recognition* (2012).

[13] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision 88*, 2 (2010), 303–338.

[14] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research 9* (2008), 1871–1874.

[15] Felzenszwalb, P. F., Girshick, R. B., McAllester, D. A., and Ramanan, D. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 9 (2010), 1627–1645.

[16] Felzenszwalb, P. F., and Zabih, R. Dynamic programming and graph algorithms in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence 33*, 4 (2011), 721–740.

[17] Frey, B. J., and Dueck, D. Clustering by passing messages between data points. *Science*, 315 (2007), 972–976.

[18] Gaidon, A., Harchaoui, Z., and Schmid, C. Recognizing activities with cluster-trees of tracklets. In *British Machine Vision Conference* (2012).

[19] Gaidon, A., Harchaoui, Z., and Schmid, C. Activity representation with motion hierarchies. *International Journal of Computer Vision 107*, 3 (2014), 219–238.

[20] Gilbert, A., Illingworth, J., and Bowden, R. Fast realistic multi-action recognition using mined dense spatio-temporal features. In *IEEE International Conference on Computer Vision* (2009).

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

[22] Gkioxari, G., Girshick, R., and Malik, J. Contextual action recognition with R*CNN. In *IEEE International Conference on Computer Vision* (2015).

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

[24] Gorelick, L., Blank, M., Shechtman, E., Irani, M., and Basri, R. Actions as space-time shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*, 12 (2007), 2247–2253.

[25] Gupta, A., Kembhavi, A., and Davis, L. S. Observing human-object interactions: Using spatial and functional compatibility for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 31*, 10 (2009), 1775–1789.

[26] Hadfield, S., and Bowden, R. Hollywood 3D: Recognizing actions in 3D natural scenes. In *Conference on Computer Vision and Pattern Recognition* (2013).

[27] Hadfield, S., Lebeda, K., and Bowden, R. Natural action recognition using invariant 3D motion encoding. In *European Conference on Computer Vision* (2014).

[28] Heilbron, F. C., Escorcia, V., Ghanem, B., and Niebles, J. C. Activitynet: A large-scale video benchmark for human activity understanding. In *Conference on Computer Vision and Pattern Recognition* (2015).

[29] Hoai, M., and De la Torre, F. Max-margin early event detectors. *International Journal of Computer Vision 107*, 2 (2014), 191–202.

[30] Hoai, M., and Zisserman, A. Discriminative sub-categorization. In *Conference on Computer Vision and Pattern Recognition* (2013).

[31] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural Computing 9*, 8 (1997), 1735–1780.

[32] Ikizler, N., and Forsyth, D. A. Searching for complex human activities with no visual examples. *International Journal of Computer Vision 80*, 3 (2008), 337–357.

[33] Ikizler-Cinbis, N., Cinbis, R. G., and Sclaroff, S. Learning actions from the web. In *IEEE International Conference on Computer Vision* (2009).

[34] Ikizler-Cinbis, N., and Sclaroff, S. Object, scene and actions: Combining multiple features for human action recognition. In *European Conference on Computer Vision* (2010).

[35] Ikizler-Cinbis, N., and Sclaroff, S. Web-based classifiers for human action recognition. *IEEE Transactions on Multimedia 14*, 4 (2012), 1031–1045.

[36] Iosifidis, A., Tefas, A., and Pitas, I. Human action recognition based on bag of features and multi-view neural networks. In *IEEE International Conference on Image Processing* (2014).

[37] Jhuang, H., Gall, J., Zuffi, S., Schmid, C., and Black, M. J. Towards understanding action recognition. In *IEEE International Conference on Computer Vision* (2013).

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

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

[40] Jiang, Y.-G., Liu, J., Roshan Zamir, A., Laptev, I., Piccardi, M., Shah, M., and Sukthankar, R. THUMOS challenge: Action recognition with a large number of classes. `http://crcv.ucf.edu/ICCV13-Action-Workshop/`, 2013.

[41] Kantorov, V., and Laptev, I. Efficient feature extraction, encoding, and classification for action recognition. In *Conference on Computer Vision and Pattern Recognition* (2014).

[42] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. Large-scale video classification with convolutional neural networks. In *Conference on Computer Vision and Pattern Recognition* (2014).

[43] Ke, Y., Sukthankar, R., and Hebert, M. Event detection in crowded videos. In *IEEE International Conference on Computer Vision* (2007).

[44] Khurram Soomro, A. R. Z., and Shah, M. UCF101: A dataset of 101 human action classes from videos in the wild. In *CRCV-TR-12-01* (2012).

[45] Kong, Y., and Fu, Y. Max-margin action prediction machine. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2015).

[46] Kovashka, A., and Grauman, K. Learning a hierarchy of discriminative space-time neighborhood features for human action recognition. In *Conference on Computer Vision and Pattern Recognition* (2010).

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

[48] Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., and Serre, T. HMDB: a large video database for human motion recognition. In *IEEE International Conference on Computer Vision* (2011).

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

[50] Laptev, I., Marszalek, M., Schmid, C., and Rozenfeld, B. Learning realistic human actions from movies. In *Conference on Computer Vision and Pattern Recognition* (2008).

[51] Leordeanu, M., Sukthankar, R., and Sminchisescu, C. Efficient closed-form solution to generalized boundary detection. In *European Conference on Computer Vision* (2012).

[52] Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. *Conference on Computer Vision and Pattern Recognition* (2015).

[53] Marszałek, M., Laptev, I., and Schmid, C. Actions in context. In *Conference on Computer Vision and Pattern Recognition* (2009).

[54] Matikainen, P., Hebert, M., and Sukthankar, R. Representing pairwise spatial and temporal relations for action recognition. In *European Conference on Computer Vision* (2010).

[55] Ng, J. Y.-H., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., and Toderici, G. Beyond short snippets: Deep networks for video classification. In *Conference on Computer Vision and Pattern Recognition* (2015).

[56] Ni, B., Paramathayalan, V. R., and Moulin, P. Multiple granularity analysis for fine-grained action detection. In *Conference on Computer Vision and Pattern Recognition* (2014).

[57] Oneata, D., Verbeek, J., and Schmid, C. Action and event recognition with Fisher vectors on a compact feature set. In *IEEE International Conference on Computer Vision* (2013).

[58] Patron-Perez, A., Marszalek, M., Reid, I., and Zisserman, A. Structured learning of human interactions in TV shows. *IEEE Transactions on Pattern Analysis and Machine Intelligence 34*, 12 (2012), 2441–2453.

[59] Patron-Perez, A., Marszalek, M., Zisserman, A., and Reid, I. D. High five: Recognizing human interactions in TV shows. In *British Machine Vision Conference* (2010).

[60] Perronnin, F., Sánchez, J., and Mensink, T. Improving the Fisher kernel for large-scale image classification. In *European Conference on Computer Vision* (2010).

[61] Pham, V., Kermorvant, C., and Louradour, J. Dropout improves recurrent neural networks for handwriting recognition. *CoRR* (2013).

[62] Ramanan, D., and Forsyth, D. A. Automatic annotation of everyday movements. In *Neural Information Processing Systems* (2003).

[63] Raptis, M., Kokkinos, I., and Soatto, S. Discovering discriminative action parts from mid-level video representations. In *Conference on Computer Vision and Pattern Recognition* (2012).

[64] Raptis, M., and Sigal, L. Poselet key-framing: A model for human activity recognition. In *Conference on Computer Vision and Pattern Recognition* (2013).

[65] Rodriguez, M. D., Ahmed, J., and Shah, M. Action MACH: A spatio-temporal maximum average correlation height filter for action recognition. In *Conference on Computer Vision and Pattern Recognition* (2008).

[66] Rohrbach, M., Rohrbach, A., Regneri, M., Amin, S., Andriluka, M., Pinkal, M., and Schiele, B. Recognizing fine-grained and composite activities using hand-centric features and script data. *International Journal of Computer Vision* (2015).

[67] Ryoo, M. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *IEEE International Conference on Computer Vision* (2011).

[68] Sadanand, S., and Corso, J. J. Action bank: A high-level representation of activity in video. In *Conference on Computer Vision and Pattern Recognition* (2012).

[69] Simonyan, K., and Zisserman, A. Two-stream convolutional networks for action recognition in videos. In *Neural Information Processing Systems* (2014).

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

[71] Sun, C., Shetty, S., Sukthankar, R., and Nevatia, R. Temporal localization of fine-grained actions in videos by domain transfer from web images. *arXiv preprint arXiv:1504.00983* (2015).

[72] Tian, Y., Sukthankar, R., and Shah, M. Spatiotemporal deformable part models for action detection. In *Conference on Computer Vision and Pattern Recognition* (2013).

[73] Todorovic, S. Human activities as stochastic Kronecker graphs. In *European Conference on Computer Vision* (2012).

[74] Tran, D., and Yuan, J. Optimal spatio-temporal path discovery for video event detection. In *Conference on Computer Vision and Pattern Recognition* (2011).

[75] Tran, D., and Yuan, J. Max-margin structured output regression for spatio-temporal action localization. In *Neural Information Processing Systems* (2012).

[76] Wang, H., Kläser, A., Schmid, C., and Liu, C.-L. Action recognition by dense trajectories. In *Conference on Computer Vision and Pattern Recognition* (2011).

[77] Wang, H., Kläser, A., Schmid, C., and Liu, C.-L. Dense trajectories and motion boundary descriptors for action recognition. *International Journal of Computer Vision* (2013), 1–20.

[78] Wang, H., Oneata, D., Verbeek, J., and Schmid, C. A robust and efficient video representation for action recognition. *International Journal of Computer Vision* (2015).

[79] Wang, H., and Schmid, C. Action recognition with improved trajectories. In *IEEE International Conference on Computer Vision* (2013).

[80] Wang, H., and Schmid, C. LEAR-INRIA submission for the THUMOS workshop. In *IEEE International Conference on Computer Vision, Workshop on Action Recognition with a Large Number of Classes* (2013).

[81] Wang, H., Wu, X., and Jia, Y. Video annotation via image groups from the web. *Multimedia, IEEE Transactions on 16*, 5 (Aug 2014), 1282–1291.

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

[83] Wang, L., and Sahbi, H. Directed acyclic graph kernels for action recognition. In *IEEE International Conference on Computer Vision* (2013).

[84] Wang, Y., Huang, K., and Tan, T. Human activity recognition based on R transform. In *Conference on Computer Vision and Pattern Recognition* (2007).

[85] Wang, Y., and Mori, G. Learning a discriminative hidden part model for human action recognition. In *Neural Information Processing Systems* (2008).

[86] Wang, Y., and Mori, G. Hidden part models for human action recognition: Probabilistic versus max margin. *IEEE Transactions on Pattern Analysis and Machine Intelligence 33*, 7 (2011), 1310–1323.

[87] Weinland, D., Boyer, E., and Ronfard, R. Action recognition from arbitrary views using 3D exemplars. In *IEEE International Conference on Computer Vision* (2007).

[88] Weinland, D., Ronfard, R., and Boyer, E. A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding 115*, 2 (2011), 224–241.

[89] Weinzaepfel, P., Harchaoui, Z., and Schmid, C. Learning to track for spatio-temporal action localization. In *IEEE International Conference on Computer Vision* (2015).

[90] Wu, B., Yuan, C., and Hu, W. Human action recognition based on context-dependent graph kernels. In *Conference on Computer Vision and Pattern Recognition* (2014).

[91] Wu, X., Xu, D., Duan, L., and Luo, J. Action recognition using context and appearance distribution features. In *Conference on Computer Vision and Pattern Recognition* (2011).

[92] Wu, Z., Wang, X., Jiang, Y., Ye, H., and Xue, X. Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. In *ACM Multimedia Conference* (2015).

[93] Xie, Y., Chang, H., Li, Z., Liang, L., Chen, X., and Zhao, D. A unified framework for locating and recognizing human actions. In *Conference on Computer Vision and Pattern Recognition* (2011).

[94] Yang, X., and Tian, Y. Action recognition using super sparse coding vector with spatio-temporal awareness. In *European Conference on Computer Vision* (2014).

[95] Yang Wang, Duan Tran, Z. L., and Forsyth, D. Discriminative hierarchical part-based models for human parsing and action recognition. *Journal of Machine Learning Research 13* (2012), 30753102.

[96] Yao, B., and Fei-Fei, L. Grouplet: A structured image representation for recognizing human and object interactions. In *Conference on Computer Vision and Pattern Recognition* (2010).

[97] Yao, B., Jiang, X., Khosla, A., Lin, A. L., Guibas, L., and Fei-Fei, L. Human action recognition by learning bases of action attributes and parts. In *IEEE International Conference on Computer Vision* (2011).

[98] Yeung, S., Russakovsky, O., Jin, N., Andriluka, M., Mori, G., and Li, F. Every moment counts: Dense detailed labeling of actions in complex videos. *arXiv preprint arXiv:1507.05738* (2015).

[99] Yuan, J., Liu, Z., and Wu, Y. Discriminative subvolume search for efficient action detection. In *Conference on Computer Vision and Pattern Recognition* (2009).

[100] Zhang, H., Zhou, W., Reardon, C. M., and Parker, L. E. Simplex-based 3D spatio-temporal feature description for action recognition. In *Conference on Computer Vision and Pattern Recognition* (2014).

[101] Zitnick, C. L., and Dollár, P. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision* (2014).

Curriculum Vitae of Shugao Ma

Address MCS 211, 111 Cummington Mall
Department of Computer Science, Boston University
Boston, 02215, MA, US

Email shugaoma@bu.edu

Website http://cs-people.bu.edu/shugaoma/

Phone 617-875-2992

Education

Ph.D in Computer Science
· Boston University, Sep. 2010 – May. 2016
· Advisor: Prof. Stan Sclaroff

Master of Engineering in Computer Applied Technology
· Chinese Academy of Sciences, China, Sep. 2006 – May. 2009
· Advisor: Prof. Weiqiang Wang

Bachelor of Engineering in Software Engineering
· Fudan University, China, Sep. 2002 – May. 2006

Publications

1. **Shugao Ma**, Leonid Sigal and Stan Sclaroff. Learning Activity Progression in LSTMs for Activity Detection and Early Detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
2. **Shugao Ma**, Sarah Adel Bargal, Jianming Zhang, Leonid Sigal and Stan Sclaroff. Do Less and Achieve More: Training CNNs for Action Recognition Utilizing Action Images from the Web. *arXiv*, 2015.
3. **Shugao Ma**, Leonid Sigal and Stan Sclaroff. Space-Time Tree Ensemble for Action Recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, oral, 2015.
4. Jianming Zhang, **Shugao Ma**, Mehrnoosh Sameki, Stan Sclaroff, Margrit Betke, Zhe Lin, Xiaohui Shen, Brian Price and Radomir Mech. Salient Object Subitizing. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

5. Jianming Zhang, **Shugao Ma**, and Stan Sclaroff. MEEM: Robust Tracking via Multiple Experts using Entropy Minimization. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2014.
6. Svebor Karaman, Lorenzo Seidenari, **Shugao Ma**, Alberto Del Bimbo and Stan Sclaroff. Adaptive Structured Pooling for Action Recognition. In *Proceedings of British Machine Vision Conference (BMVC)*, 2014.
7. **Shugao Ma**, Jianming Zhang, Nazli Ikizler-Cinbis and Stan Sclaroff. Action Recognition and Localization by Hierarchical Space-Time Segments. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2013.
8. **Shugao Ma**, Nazli Ikizler-Cinbis and Stan Sclaroff. Unsupervised Learning of Discriminative Relative Visual Attributes. In *Proceedings of 2nd International Workshop on Parts and Attributes, in conjunction with European Conference on Computer Vision (ECCV)*, 2012.
9. **Shugao Ma** and Weiqiang Wang. Effective Fighting Shot Discrimination in Action Movie. In *Journal of Computer Science and Technology (JCST)*, Vol.26, No.1, pp.187-19, 2011.
10. **Shugao Ma** and Weiqiang Wang. Effective Camera Motion Analysis Approach. In *Proceedings of IEEE International Conference on Networking, Sensing and Control (ICNSC)*, 2010.
11. **Shugao Ma**, Weiqiang Wang, Shuqiang Jiang, Qingming Huang and Wen Gao. Effective Scene Matching with Local Feature Representatives. In *Proceedings of International Conference on Pattern Recognition (ICPR)*, 2008.