

Electronic Theses and Dissertations, 2004-2019

2013

Human Action Localization And Recognition In Unconstrained Videos

Hakan Boyraz
University of Central Florida

 Part of the [Electrical and Electronics Commons](#)
Find similar works at: <https://stars.library.ucf.edu/etd>
University of Central Florida Libraries <http://library.ucf.edu>

This Doctoral Dissertation (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations, 2004-2019 by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation

Boyraz, Hakan, "Human Action Localization And Recognition In Unconstrained Videos" (2013). *Electronic Theses and Dissertations, 2004-2019*. 2734.
<https://stars.library.ucf.edu/etd/2734>

HUMAN ACTION LOCALIZATION AND RECOGNITION IN UNCONSTRAINED VIDEOS

by

HAKAN BOYRAZ
B.S. Hacettepe University, 2002
M.S. Bilkent University, 2005

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

Fall Term
2013

Major Professor: Marshall Tappen

© 2013 Hakan Boyraz

ABSTRACT

As imaging systems become ubiquitous, the ability to recognize human actions is becoming increasingly important. Just as in the object detection and recognition literature, action recognition can be roughly divided into classification tasks, where the goal is to classify a video according to the action depicted in the video, and detection tasks, where the goal is to detect and localize a human performing a particular action. A growing literature is demonstrating the benefits of localizing discriminative sub-regions of images and videos when performing recognition tasks. In this thesis, we address the action detection and recognition problems.

Action detection in video is a particularly difficult problem because actions must not only be recognized correctly, but must also be localized in the 3D spatio-temporal volume. We introduce a technique that transforms the 3D localization problem into a series of 2D detection tasks. This is accomplished by dividing the video into overlapping segments, then representing each segment with a 2D video projection. The advantage of the 2D projection is that it makes it convenient to apply the best techniques from object detection to the action detection problem. We also introduce a novel, straightforward method for searching the 2D projections to localize actions, termed Two-Point Subwindow Search (TPSS). Finally, we show how to connect the local detections in time using a chaining algorithm to identify the entire extent of the action. Our experiments show that video projection outperforms the latest results on action detection in a direct comparison.

Second, we present a probabilistic model learning to identify discriminative regions in videos from weakly-supervised data where each video clip is only assigned a label describing what action is present in the frame or clip. While our first system requires every action to be manually outlined in every frame of the video, this second system only requires that the video be given a single high-level tag. From this data, the system is able to identify discriminative regions that correspond well

to the regions containing the actual actions. Our experiments on both the MSR Action Dataset II and UCF Sports Dataset show that the localizations produced by this weakly supervised system are comparable in quality to localizations produced by systems that require each frame to be manually annotated. This system is able to detect actions in both 1) non-temporally segmented action videos and 2) recognition tasks where a single label is assigned to the clip. We also demonstrate the action recognition performance of our method on two complex datasets, i.e. HMDB and UCF101.

Third, we extend our weakly-supervised framework by replacing the recognition stage with a two-stage neural network and apply dropout for preventing overfitting of the parameters on the training data. Dropout technique has been recently introduced to prevent overfitting of the parameters in deep neural networks and it has been applied successfully to object recognition problem. To our knowledge, this is the first system using dropout for action recognition problem. We demonstrate that using dropout improves the action recognition accuracies on HMDB and UCF101 datasets.

To my family and my beautiful niece Zeynep Nil

ACKNOWLEDGMENTS

I would like to gratefully and sincerely thank my adviser Dr. Marshall Tappen for his valuable guidance, encouragement and continuing support during my PhD program.

I also would like to thank Dr. Rahul Sukthankar for his assistance and guidance, both academically and personally, during the course of this thesis. I would like to thank rest of my committee members, Dr. Hassan Foroosh, Dr. Mingjie Lin and Dr. Shaojie Zhang, for their time and valuable suggestions. I also thank my labmates Baoyuan Liu and Syed Zain Masood for all the help they provided.

I would like to specially thank my wife Özge, my parents Hasan and Sultan Boyraz, my brother Gökhan and his wife Meryem, for their unconditional love and support.

TABLE OF CONTENTS

LIST OF FIGURES	xi
LIST OF TABLES	xvii
CHAPTER 1: INTRODUCTION	1
1.1 Applications	1
1.2 Challenges and Datasets	3
1.3 Our Contributions	4
1.3.1 Localizing Actions through Sequential 2D Video Projections	5
1.3.2 A Weakly-Supervised Probabilistic Model for Action Localization and Recognition	7
1.3.3 Action Recognition using Neural Networks with Dropout	9
1.4 Organization of Dissertation	9
CHAPTER 2: LITERATURE REVIEW	11
2.1 Action Recognition	11
2.1.1 Global Representation	12
2.1.2 Local Representation	14

2.2	Action Localization	17
2.3	Learning to Detect Objects from Weakly-Supervised Data	18
CHAPTER 3: LOCALIZING ACTIONS THROUGH SEQUENTIAL 2D VIDEO PROJECTIONS		19
3.1	Video Representation	20
3.2	Reducing Action Localization to 2D Search	22
3.3	Subvolume Search in 2D Video Projection	25
3.3.1	Efficient Subwindow Search	26
3.3.2	Two Point Subwindow Search	28
3.4	Chaining Subvolumes Across Time	31
3.5	Computational Complexity	32
3.6	Experimental Results	34
3.6.1	Cross-Dataset Action Detection Comparisons	36
3.6.2	Additional Results	38
CHAPTER 4: A WEAKLY-SUPERVISED PROBABILISTIC MODEL FOR ACTION LOCALIZATION AND RECOGNITION		40
4.1	Overview of Video Representation	41

4.2	A Weakly-Supervised Model for Action Localization	42
4.2.1	Probabilistic Model for Localizing Discriminative Sub-Regions	43
4.2.2	Incorporating Weakly Supervised Tags	44
4.2.3	Detecting Actions	44
4.2.4	Action Detection Experiments	47
4.3	Action Recognition by Discriminative Sub-Region Localization	50
4.3.1	Model Implementation	50
4.3.1.1	Localizing Discriminative Sub-Regions	52
4.3.1.2	Estimating Histograms to Represent Localized Sub-Regions	53
4.3.1.3	Kernel Map	54
4.3.1.4	Action Classification	56
4.3.2	Learning	57
4.3.3	Experimental Evaluation	58
4.3.3.1	UCF Sports Dataset	59
4.3.3.1.1	Accuracy Results	60
4.3.3.1.2	Localization Results	63
4.3.3.2	HMDB Dataset	67

4.3.3.3 UCF101 Dataset 76

CHAPTER 5: ACTION RECOGNITION USING NEURAL NETWORKS WITH DROPOUT
82

5.1 Video Representation 83

5.2 Framework Overview 85

5.3 Network Training 88

 5.3.1 Dropout 90

 5.3.2 Computing the Error Gradients 91

5.4 Experimental Evaluation 93

CHAPTER 6: CONCLUSION AND FUTURE WORK 96

6.1 Summary of Contributions 96

6.2 Future Work 97

LIST OF REFERENCES 99

LIST OF FIGURES

Figure 1.1: Sample video frames from UCF Sports Action Dataset.	2
Figure 2.1: Examples of global action representations. (a) Motion Energy and Motion History Images for handwaving action (figure is taken from [22]). (b) A sequence of tracked object contours for falling action and STV for falling action generated by applying the point correspondence method (figure is taken from [23]). (c) Spacetime volume of stacked silhouettes and corresponding solutions to the Poisson equation on space-time shapes (figure is taken from [24])	13
Figure 2.2: Bag of visual words framework (figure is taken from [34]).	15
Figure 2.3: Space-time-interest-points. (a) Harris3D detector (figure is taken from [35]). (b) Dollar’s interest point detector (figure is taken from [26]).	16
Figure 3.1: Space-time interest points (STIP) are extracted in a video sequence using Harris3D detector and HOG & HOF descriptors are computed for each STIP.	20
Figure 3.2: We represent a video sequence as a collection of overlapping video chunks and each video chunk by a collection of STIPs. The goal of action detection is to localize subvolumes that contain the action of interest.	22
Figure 3.3: We need only consider subvolumes in a video chunk that touch a STIP on each face; through video projection, we model subvolumes as 2D rectangles.	23

Figure 3.4: Video projection converts the subvolume localization problem to a 2D search problem. Blue (Red) points correspond to STIPs with positive (negative) SVM weights.	25
Figure 3.5: A candidate subset of rectangles in the parameter space is represented as tuples $[T, B, L, R]$, where T, B, L and R define the maximum and minimum values for t, b, l and r , respectively. Figure is courtesy of Lampert et al.s CVPR 2008 talk on ESS.	26
Figure 3.6: (a) Two STIPs with positive SVM weights define a rectangle with STIPs at opposite corners. We search over all such possible rectangles followed by non-maximal suppression. (b) Integral image [57] is used for efficiently computing the SVM score of any rectangle using only four operations. The score of area D is computed as $\bar{4} - B - C + A$, where $\bar{4}$ is the accumulated sum at point 4.	29
Figure 3.7: (a) Action segments detected in subsequent video chunks. (b) Two action segments belong to the same action instance, i.e. connected, if intersection over union volume of the action segments is greater than a threshold.	31
Figure 3.8: (a) Sample video frames from KTH dataset. (b) Sample video frames from MSR dataset. Different than KTH dataset, there are other activities happening in the background.	34
Figure 3.9: Summary of our training and test methodology.	36
Figure 3.10: Direct comparison on MSR Action II, trained using KTH+4 clips of MSR. Both variants of proposed method, VP+ESS and VP+TPSS, outperform Cao <i>et al.</i> [5] despite their simplicity.	37

Figure 3.11: VP+ESS on the waving action for different video chunk sizes. The system performs best when longer chunks are used, but performance is acceptable if smaller chunks are needed, such as for on-line recognition.	38
Figure 3.12: Examples of action detections on MSR dataset using VP+TPSS. Colored boxes denote detected actions: boxing (blue), clapping (green), and waving (red). Qualitatively, our results are comparable to Cao <i>et al.</i> [5].	39
Figure 4.1: Interest points (STIPs) are sampled at dense space time intervals and a single 2D image can represent the quantized STIP features for a given range of the action video frames. We use the term ‘frame’ to represent the descriptor images.	42
Figure 4.2: Each sub-region, r , in a descriptor frame is represented as histogram of quantized features contained in the sub-region and the joint probability distribution between sub-regions and action labels are computed.	45
Figure 4.3: Action detection on MSR-II compared with [5]. Black curves show our result trained with randomly selected 4 videos and Red curves show our detection result trained with 3-fold cross validation. Training using just 4 videos, our performance is acceptable. With more training videos, the localization improves significantly and outperforms [5].	48
Figure 4.4: Examples of action detections on MSR dataset. Our method gives reasonable localization results on most of the videos.	49

Figure 4.5: This diagram illustrates our approach for recognizing actions. First, a set of localizers estimate which sub-regions in the video frames are likely to contain the discriminative information necessary to classify the action. These candidate sub-regions are represented by histograms of visual words. Next, the feature vectors generated by the histograms are used to compute the probability of any particular combination of candidate sub-region and action class. An important, unique aspect of this approach is that the system is designed so that localization parameters (Φ) and recognition parameters (Θ) are trained to maximize the probability of the correct action class. This makes it possible to train this system in a weakly-supervised fashion. 51

Figure 4.6: Our objective is to compute a video representation by selecting the sub-region in each video frame that maximizes the probability in Equation (4.6) and accumulating the histograms of sub-regions across all video frames. 53

Figure 4.7: Sample video frames from UCF Sports dataset for each category. 59

Figure 4.8: Confusion matrix for the UCF Sports dataset using train/test split. 61

Figure 4.9: We show localization results obtained using our method on the UCF Sports action dataset. We can see that our model is able to correctly localize action specific sub-regions as the best possible representation of the action being conducted in the video. 65

Figure 4.10: Comparison of action localization performance against Lan et al. [8]. (a) ROC curves for $\sigma = 0.2$. (b) Area Under ROC for different σ . σ is the threshold that determines if a video is correctly localized. Compared with [8], which requires the action be manually located in the training data, our system produces comparable or improved results.	66
Figure 4.11: Sample video frames from HMDB dataset.	68
Figure 4.12: Confusion matrix for HMDB dataset using STIP features. (a) global BoW (b) our method	72
Figure 4.13: Comparison of action class accuracies for HMDB dataset using STIP features. The bars show the percentage of the videos that are correctly classified in each action class.	73
Figure 4.14: Confusion matrix for HMDB dataset using MBH features. (a) global BoW (b) our method	74
Figure 4.15: Comparison of action class accuracies for HMDB dataset using MBH features. The bars show the percentage of the videos that are correctly classified in each action class.	75
Figure 4.16: Confusion matrix for UCF101 dataset using STIP features. (a) global BoW (b) our method	78
Figure 4.17: Comparison of action class accuracies for UCF101 dataset using STIP features. The bars show the percentage of the videos that are correctly classified in each action class.	79

Figure 4.18: Confusion matrix for UCF101 dataset using MBH features. (a) global BoW (b) our method	80
Figure 4.19: Comparison of action class accuracies for UCF101 dataset using MBH fea- tures. The bars show the percentage of the videos that are correctly classi- fied in each action class.	81
Figure 5.1: Action recognition framework using neural network with dropout. The first stage of the framework is kept same as the one in Figure 4.5 and the second stage is replaced with a two-layer neural network.	83
Figure 5.2: Feature points are sampled densely at each spatial scale and tracked for $L = 15$ frames. MBH features are computed within a space-time volume aligned with the trajectories. (Figure is taken from [44])	84
Figure 5.3: Network diagram for a two-layer neural network with B inputs, M hidden units and C outputs. The output of the kernel map are used as input to the neural network.	86
Figure 5.4: Two different network structures are used for multiple localizers: (a) the localizer parameters are only connected to a subset of the hidden units (b) fully connected neural network where all localizer parameters are con- nected to all hidden units.	88
Figure 5.5: (a) In a fully connected neural network, all the hidden units contribute to the forward pass and backpropagation. (b) When there is dropout, the output of each hidden unit is set to zero with probability 0.5 and the hidden units that are dropped out do not contribute to the forward pass and backpropagation.	90

LIST OF TABLES

Table 3.1:	Worst case complexity comparison of our method and naive 3D branch-and-bound search for detecting actions in a given video sequence.	33
Table 4.1:	Mean per-class action recognition accuracies (split) on the UCF Sports dataset. We show that our method outperforms both global and results reported in [8–10]. * Both [8] and [9] use ground truth annotations during training where as our model is weakly supervised and does not require ground truth annotations.	60
Table 4.2:	Mean per-class action recognition accuracies (LOOCV) on the UCF Sports dataset. Our LOOCV results are comparable to the state-of-the-art.	62
Table 4.3:	Classification accuracy w.r.t. the number of discriminative sub-region detectors. We observe that increasing localizers improves recognition performance up until $D = 10$. Beyond that, there is a drop in performance, which can be attributed to overfitting.	63
Table 4.4:	Mean per-class action classification accuracies for the first 10 action classes from HMDB dataset using STIP features. The number of localizers is set to 10 in a similar setting to the UCF Sports Dataset.	69
Table 4.5:	Mean per-class action classification accuracies on HMDB dataset using STIP features for different number of localizers.	69
Table 4.6:	Comparison of our method with global BoW and other methods that use STIP (HOG/HOF) and MBH features.	70

Table 4.7:	Comparison of our method with global BoW using STIP features.	77
Table 4.8:	Comparison of our method with global BoW using MBH features.	77
Table 5.1:	Comparison of the action classification accuracies for STIP features on HMDB dataset with and without using the dropout.	94
Table 5.2:	Comparison of our method with other methods that use STIP (HOG/HOF) and MBH features on HMDB dataset.	94
Table 5.3:	Mean per-class action classification accuracies for UCF101 dataset using STIP and MBH features.	95

CHAPTER 1: INTRODUCTION

Human action recognition and detection are very popular research areas in computer vision. The goal of human action recognition is to automatically classify a given video sequence based on the action performed in the video. In case of action recognition, the video is generally segmented to contain only one execution of a human action. In more general cases, the video may contain multiple actions and the goal of action detection is not just to recognize the actions being performed in the video but also determine the spatio-temporal boundaries of them.

While the terms ‘action’ and ‘activity’ are used interchangeably in the computer vision literature, the human actions can be considered as simple motion patterns generally performed by a single person for a short period of time whereas the activities are more complex sequence of actions performed by several people. Figure 1.1 shows sample human actions from UCF Sports Action dataset.

1.1 Applications

Applications of human action recognition and detection include content based video analysis and retrieval, security and surveillance systems, behavioral biometrics, human-computer interactions and robotics.

As video sharing websites like YouTube become popular, it has become necessary to develop efficient and reliable video indexing and retrieval algorithms to deal with the vast amount of amateur videos captured by hand-held cameras or cell phones. Indexing and searching the videos based on the content of the video, instead of user attached keywords, is one of the important applications of human action recognition.



Figure 1.1: Sample video frames from UCF Sports Action Dataset.

Automatic recognition of human anomalies is another application of human action recognition systems. Traditional security and surveillance systems contain a network of video cameras and human operators need to be aware of the activity in the camera field of view. Vision based solutions can replace or assist human operators.

Another application of human action recognition is behavioral biometrics. Biometrics refers to identification of humans by their characteristics. Traditional biometric identification approaches rely on physiological characteristics of people such as fingerprint, face recognition and palm print. Most of those methods require cooperation from the subject for collection of the biometric. On the other hand, behavioral biometric identification methods rely on the behavior of a person such as gait and do not require cooperation from the subject for collection of the data.

Gesture-based human computer interfaces such as Kinect enable users to control and interact with

the game consoles without need to touch a physical game controller. Other applications of human action recognition systems include, but not limited to, monitoring of elderly people and children, intelligent environments, and sports play analysis.

1.2 Challenges and Datasets

There are several challenges involved with action recognition systems such as variations in appearance of actors performing the actions, background clutter, occlusions, camera motion, different camera view points, variations in illumination, etc.

Several standard action datasets have been used by computer vision community for developing robust action recognition and detection algorithms. In earlier datasets such as KTH [1], actions were performed by different actors and the videos were captured in controlled environments where there was no background clutter or camera motion. Even though those datasets were helpful to develop initial action recognition algorithms, they did not provide the challenges necessary to develop more realistic and robust action recognition algorithms.

In recent years, more challenging datasets have been released such as UCF Sports Action [2], Human Motion DataBase (HMDB) [3] and UCF101 [4]. The UCF Sports Action dataset consists of set of actions collected from various sports which are typically featured on broadcast television channels. Compared with the KTH dataset, it is more challenging, because the videos were captured in more realistic environments where there were background clutter and camera motion. However, there are only 11 action categories and 200 videos contained in UCF Sports Action dataset which makes it relatively simple dataset for action recognition tasks when compared to larger datasets such as HMDB and UCF101. The UCF Sports Action dataset is mostly used for action detection tasks since the ground truth locations of the actions in each frame are provided.

The HMDB dataset contains 51 action classes and 6849 video clips which were collected from various sources, mostly from movies, and a small proportion from public databases such as YouTube and Google videos. The UCF101 is a recently introduced dataset with 13320 videos from 101 action categories. UCF101 gives the largest diversity in terms of actions and with the presence of large variations in camera motion, object appearance and pose, object scale, viewpoint, cluttered background, and illumination conditions.

Videos in the above datasets are segmented temporally to contain only one instance of an action and therefore they are mostly used for action recognition tasks. In contrast with the previous datasets, the Microsoft Research Action Dataset II (MSR) [5] consists of 54 video sequences recorded in a crowded environment and each video sequence consists of multiple actions. There are in total 203 action instances and three types of action classes. Since videos contain multiple action instances, it is commonly used for action detection tasks.

1.3 Our Contributions

In this dissertation, we focus on developing action localization and recognition algorithms which are efficient for detecting actions, do not rely on ground truth action location, do not require pre-processing with a human detector, and perform better than the baseline and the state-of-the-art action recognition algorithms. The major contributions are:

- We propose an efficient action detection algorithm that reduces the 3D action detection problem into a series of 2D search problems. In this method, each video is represented as a series of overlapping video chunks and each video chunk is searched independently for actions. Our novel 2D projection method makes it convenient to apply best techniques from object detection to action detection problem. Since our method does not rely on the whole video

content for detecting actions, it can be used in real time applications. If the whole video content is available, the video chunks can be searched in parallel for a faster action detection system.

- We introduce a unified, weakly-supervised probabilistic model for recognizing actions through localizing discriminative regions in video frames that correspond well to the regions containing the actual actions. Unlike our first method and most action localization systems, this second method does not rely on ground truth action locations during training or any saliency/human detectors for keeping the system computationally tractable. The localization and recognition parameters of the system are trained concurrently using a single learning criteria.
- Finally, we extend our proposed framework by replacing the recognition stage with a two-stage neural network and apply dropout for preventing overfitting of model parameters on the training data. To our knowledge, this is the first system to apply dropout in action recognition problem in order to prevent overfitting.

The following subsections provide a brief introduction to the action localization and recognition algorithms developed in this dissertation.

1.3.1 Localizing Actions through Sequential 2D Video Projections

As imaging systems become ubiquitous, the ability to recognize human actions is becoming increasingly important. Just as in the object detection and recognition literature, action recognition can be roughly divided into classification tasks, where the goal is to classify a video according to the action depicted in the video, and detection tasks, where the goal is to detect and localize a human performing a particular action.

The detection task is particularly challenging because the action must be detected and localized in a spatio-temporal volume. In the worst case, the system must search a six-dimensional space to locate the video volume. Recent work has built on Lampert *et al.*'s Efficient Subwindow Search method (ESS) [6] to make such searches efficient [5, 7]. While successful, these action detection methods are distinct from the popular techniques currently used for object detection, localization and classification in images, since the former employ mutual information or generative models rather than discriminative classifiers over feature descriptors.

In this work, we show that actions can be localized without explicitly searching through time. In our proposed method, a video sequence is represented as a series of overlapping video chunks and each video chunk as a collection of features. Actions can be detected by projecting chunks of the 3D spatio-temporal volume into a 2D representation, then performing a 2D search. The advantage of the proposed approach is that this 2D search can be performed using the same techniques that have proven successful in object detection. As an example, the Efficient Sub-Window Search algorithm [6] for object detection can be directly applied to action detection using this technique. We also introduce a novel, straightforward method for searching the 2D projection to localize actions, termed Two Point Sub-Window Search (TPSS). As shown in the experiments in Section 3.6, this approach leads to improved results over the ESS algorithm.

Another advantage of our method is that since the video chunks are searched independently for action segments, it can be used in real-time action detection systems. If the whole video sequence is available at the time of detection, the video chunks can be searched in parallel making it faster for action detection. Finally, we introduce a greedy algorithm in Section 3.4 showing how the action segments detected in different video chunks can be chained together to identify the entire extent of the action.

1.3.2 A Weakly-Supervised Probabilistic Model for Action Localization and Recognition

With the recent introduction of large new datasets [3] and continued development of new algorithms [8–12], action recognition continues to attract significant research interest. While action recognition research tends to focus on whole-clip classification, where the system processes an entire clip and assigns a single label to the clip, there is growing interest in localizing or detecting the regions in the video where the action is occurring [8–10, 13]. Several key challenges can be identified with the recent work on localization:

1. **Locating the Actions in Training Data** – While recent work has begun exploring weakly-supervised localization [10], the reliance on object saliency detectors for keeping the system computationally feasible becomes a bottleneck. Other previous systems have required that the location of the action at each frame be manually annotated in the training data [8, 9, 14]. This increases the cost and effort required to apply these methods to novel datasets.
2. **Incorporating Effective Non-Linear Models** – Linear models are convenient because they facilitate efficient computation, but as pointed out in [8], recognizing actions in sub-regions can require non-linear classifiers that are better able to discriminate between classes.
3. **Efficiently Applying Non-Linear Models** – While non-linear models may be necessary to properly discriminate between classes, finding non-linear models that are both descriptive and efficient can be difficult. Previous approaches, such as adding latent variables to eliminate certain features [8] or extensions of kernel methods [15], add complexity to inference. This complexity is compounded by the difficulty of working with multiple frames in the video.

To address the above concerns, we present a novel approach to learning how to localize discriminative sub-regions in the video from *weakly-supervised* data, where clips have only been annotated

with a single tag, as is common in most video datasets. While this approach cannot be guaranteed to find the region that matches the semantics of the label, we show experiments on two different datasets, MSR Action II and UCF Sports, demonstrating that this weakly-supervised approach is able to localize the action as well as methods trained on *hand-annotated* data. To our knowledge, this is the first system that learns to localize actions from high-level tags on videos. To incorporate efficient and effective non-linear models, we use kernel map transformation to approximate non-linear kernels [16].

While the focus of this work is on creating a localization system, experiments in Section 4.3.3 will also show that our approach produces recognition results on the UCF Sports database that out-perform previous localization-based methods, such as [8–10, 13]. We also provide recognition results on HMDB and UCF101 which are much larger datasets when compared to the UCF Sports and our results are better than or comparable to state-of-the-art recognition systems on those datasets.

The contribution of this work lies on its ability to harness large amounts of video, labeled with basic tags, to learn localizers that perform comparably to systems trained from manually annotated data. While crowd-sourcing has made it possible to manually annotate much larger image datasets, video poses a particular problem because a full annotation with localization requires manually annotating every frame. Speaking generally, this means that fully annotating a single, several-second clip in a video dataset requires annotating hundreds of frames. An effective weakly-supervised algorithm that can be trained from clip-level tags would significantly reduce the effort in creating training data and open the possibility to train from larger, more diverse training databases.

1.3.3 Action Recognition using Neural Networks with Dropout

One common problem with datasets containing large number of classes is the overfitting of the parameters on the training dataset. Most commonly used approach to avoid overfitting is introducing a regularization term to the cost function.

A recently introduced technique for feed-forward deep neural networks, called dropout [17], has been applied successfully to object recognition problem by preventing the overfitting substantially [18]. The dropout procedure can be viewed as a very efficient way of performing model averaging with neural networks. The overfitting on the data is reduced by setting the output of hidden units to zero with a probability of 0.5. Those units that are dropped out do not contribute to the forward passing and back-propagation. Every time a training input is presented, the architecture of the network changes, but still sharing the same weights and this prevents complex co-adaptations on the training data. Even though the success of dropout was demonstrated on large scale object recognition datasets, to our knowledge it has not been used for action recognition problem.

We extend our action recognition framework introduced in previous section by incorporating a two stage neural network with dropout into our framework. We provide accuracy results on HMDB and UCF101 datasets with and without dropout. Our results shows that using neural network instead of a multinomial logistic regression classifier improves the classification accuracies and incorporating dropout improves the results further on large datasets.

1.4 Organization of Dissertation

The rest of the dissertation is organized as follows. Chapter 2 provides an overview of the literature on human action recognition and localization. Chapter 3 introduces the proposed action detection method using the 2D video projections. Chapter 4 proposes a weakly-supervised prob-

abilistic model for action recognition and localization in unconstrained videos. Chapter 5 extends the framework proposed in Chapter 4 by replacing the recognition stage of the framework with a two-stage neural network and applies dropout for preventing overfitting of the parameters. Finally, Chapter 6 concludes the dissertation with a summary of contributions and possible future work.

CHAPTER 2: LITERATURE REVIEW

This dissertation is related to previous work in action recognition, action localization and weakly-supervised methods in object detection.

2.1 Action Recognition

A large amount of literature on the problem of recognizing actions in videos has developed over the past decade. Turaga et al. [19], Weinland et al. [20] and Poppe [21] provide good overviews of various action recognition methods and datasets. A general action recognition system consists of feature extraction, action learning and classification steps.

- **Feature Extraction and Video Representation:** Videos contain massive amount of raw pixel data and most of this data is not directly relevant to the actions. In feature extraction step, the appearance and motion cues that are discriminative with respect to human actions are extracted from raw pixels and those features are used to construct higher level representations for videos. It is important that the selected features are invariant with respect to scaling, rotation, different illuminations, etc.
- **Action Learning and Classification:** During action learning, statistical models are learned using the features extracted from training data and during action classification, learned models are used to classify the new observations. Support vector machines (SVM) are supervised learning models that are commonly used in action recognition problems.

In [21], Poppe classifies the action recognition approaches into two groups based on the video representation: global and local representations.

2.1.1 Global Representation

In global representation, a person is localized first in each video frame using background subtraction or tracking and the region of interest (ROI) around the person is encoded as a whole to construct the feature representation. Global representations can be extracted from silhouettes, shape counters or optical flow.

Bobick and Davis [22] use background subtracted blobs to construct a 2D temporal template. In their method, they first extract the blobs in each frame using background subtraction and then aggregate them into a single image. They propose two different approaches for aggregation. In the first approach, all blobs are equally weighted and the resulting image is called Motion Energy Image (MEI). In the second approach, the higher weights are given to new frames and the resulting image is called Motion History Image (MHI) (see Figure 2.1(a)). Hu-moments are extracted from those templates and used for action recognition. This method would not work well for complex actions since the motion history would be overwritten.

Instead of using a 2D template, Yilmaz and Shah [23] represent actions as 3D objects by stacking together the shape counters detected in each frame as shown in Figure 2.1(b). Sequence of counters with respect to time generate spatio-temporal volume (STV) in (x, y, t) . The STV can be treated as a 3D object and the descriptors are extracted from the objects surface corresponding to geometric features such as peaks, valleys, and ridges. One drawback of this method is that the point correspondence needs to be computed between each frame.

Blank et al. [24] use background subtracted blobs instead of counters to represent actions. Blobs are stacked together to create a binary space-time volume and shape descriptors are extracted from the 3D template as shown in Figure 2.1(c).

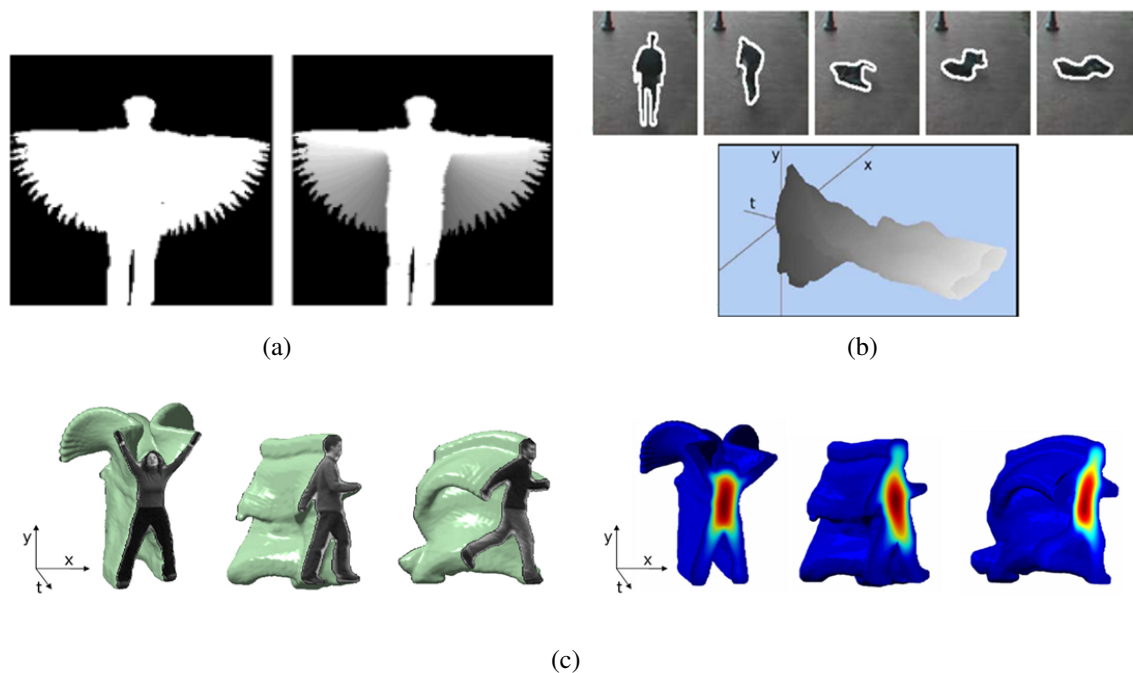


Figure 2.1: Examples of global action representations. (a) Motion Energy and Motion History Images for handwaving action (figure is taken from [22]). (b) A sequence of tracked object contours for falling action and STV for falling action generated by applying the point correspondence method (figure is taken from [23]). (c) Spacetime volume of stacked silhouettes and corresponding solutions to the Poisson equation on space-time shapes (figure is taken from [24])

Ke et al. [25] consider a video as a collection of sub-volumes of arbitrary shape. The sub-volumes are extracted by clustering the pixels based on appearance and spatial proximity. Actions are classified by correlating the action templates with the volumes using shape and flow features (volumetric region matching). Different from previous methods, it does not require background subtraction for silhouette extraction.

2.1.2 Local Representation

Local representations consider a video volume as a collection of local descriptors or parts. Different from global representations they do not depend on a robust background subtraction or tracking and they are more invariant to changes in viewpoint, person appearance and partial occlusions. Local features are extracted at densely sampled points or at space-time interest points. Once the features are extracted, they are combined into a final video representation. Bag of visual words (BoW) representations of actions in videos have proven to be remarkably powerful and robust [1, 26–28]. Using these visual codebooks, some have suggested codebook refinement techniques for improved recognition results [29, 30] while others employ higher-order relations between visual words [31–33].

Figure 2.2 shows the components of a generic bag of words framework. During learning, local spatio-temporal feature descriptors are extracted from the training videos. Subset of those features are clustered into a set of video codewords, called codebook using K-means algorithm. Then, each feature in the video is mapped to the closest codeword and the video is represented by the histogram of the codewords. Those histograms are used in conjunction with machine learning techniques such as SVMs and graphical models. During recognition, the new observations are represented as histogram of the codewords and the classifier model learned during training is used for deciding the best action label for the video.

Bag of word models do not take the temporal information into consideration. They summarize all frames of an observed sequence into a single representation or perform action recognition for each frame individually. Temporal state-space models such as Hidden Markov Model (HMM) and Conditional Random Field (CRF), on the other hand, represent an action as sequence of moments (states) connected by edges. The edges model the probabilities among states and between states and observations.

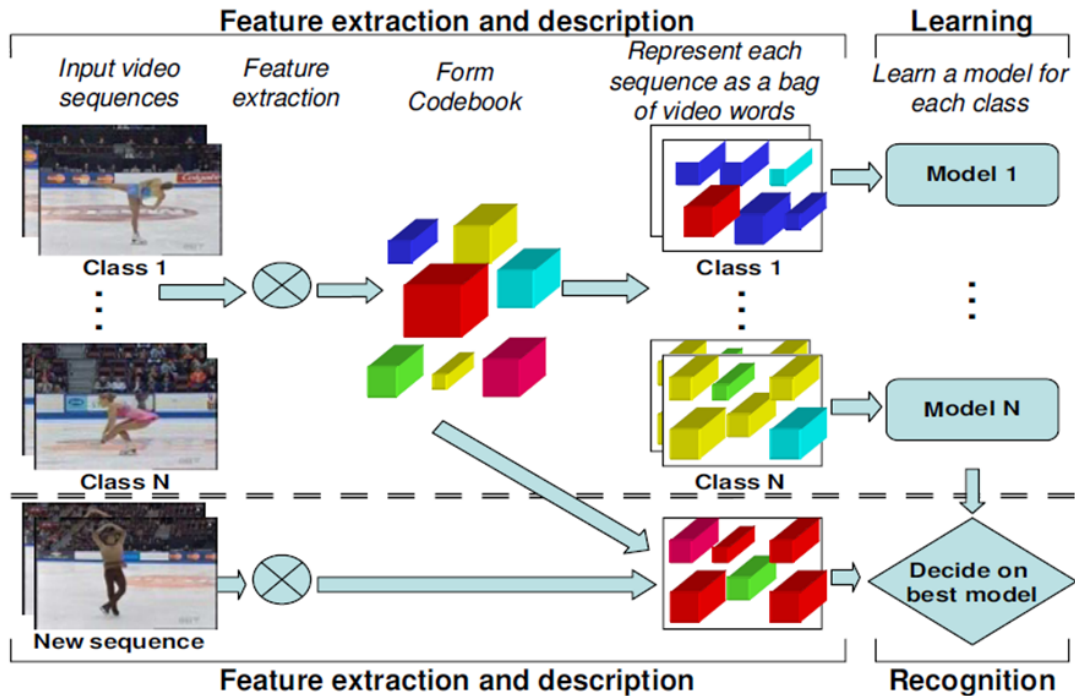


Figure 2.2: Bag of visual words framework (figure is taken from [34]).

Several methods have been proposed for interest point detection from videos. Laptev and Lindeberg [35] use a space-time generalization of Harris corner detector, which is widely used in object recognition tasks. This work is extended to compensate for relative camera motions in [36]. The drawback of these methods is that they only extract small number of stable interest points. This issue was addressed by Dollar et al. in [26]. In their method, they extract distinctive periodic motion-based interest points in a given video using a Gaussian kernel in space and a Gabor function in time.

Once the interest points are detected, the local feature descriptors are computed in the neighborhood of those interest points. In [37], space-time volumes are extracted at detected interest points and then each volume is subdivided into grid of cuboids. Histograms of oriented gradient (HOG) [38] and optical flow (HOF) are computed for each cuboid. Finally, the histograms are

normalized and concatenated into HOG and HOF descriptor vectors. In [39], Kläser et al. extend the HOG to 3D where 3D spatio temporal gradients are quantized using regular polyhedrons. They also extend integral images to integral videos for efficient 3D gradient computation. In [40], Sco-vanner et al. extend the SIFT descriptor [41] to 3D.

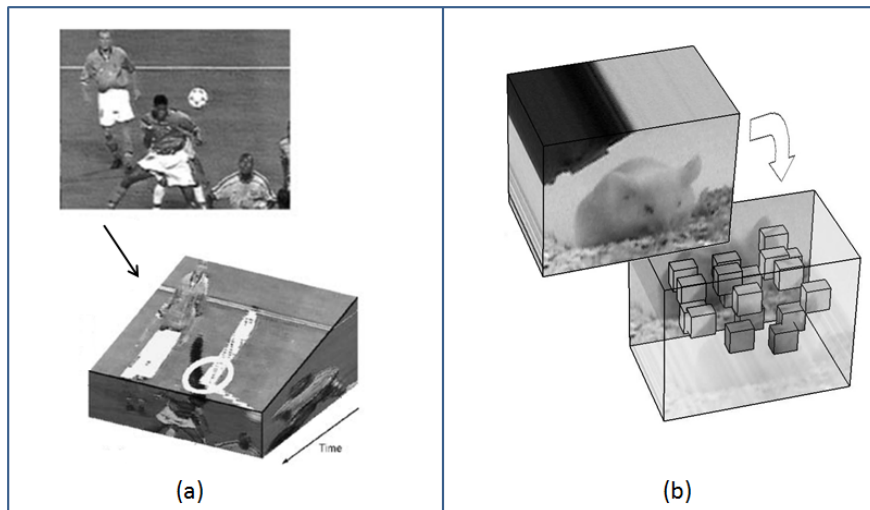


Figure 2.3: Space-time-interest-points. (a) Harris3D detector (figure is taken from [35]). (b) Dollar's interest point detector (figure is taken from [26]).

Wang et al. [42] show comparisons of different interest point detectors on a variety of available well-known complex datasets. In their paper, they found that dense sampling outperformed the interest point detectors of Dollar et al. [26], Laptev and Lindeberg [35], and Willems et al. [43]. They also compared local feature descriptors and found that combination of image gradients and optical information resulted in the best performance in general.

In [44], Wang et al. introduce a video representation based on dense trajectories and motion boundary histograms (MBH). In their method, they sample dense points from each frame and track them based on displacement information from optical flow. The histogram of gradient and histogram

of optical flow features are extracted along the trajectories. They introduce a new feature descriptor based on motion boundary histograms which rely on derivative of optical flow and therefore it is more robust to camera motion. This descriptor outperforms other descriptors, particularly in uncontrolled realistic videos such as HMDB.

2.2 Action Localization

Recent work has begun showing the benefits of localizing actions. Studies have focused on showing the importance of action localization for complex datasets [45,46] by utilizing person-location information or action detection prior to the task of recognition. Lan et al. [8] propose a figure-centric representation for action localization and recognition by treating person location as a latent variable and inferring it while simultaneously recognizing the action. Yao et al. [47] classify and localize human actions in videos using a Hough transform voting framework. Amer et al. [48] formulate a generative chain model of group activities to localize and recognize group activities. Yuan et al. [7] propose a discriminative pattern matching technique to locate the action in the 3D video space using a branch-and-bound search mechanism. Boyraz et al. [49] propose a technique that transforms the 3D action localization problem into a series of 2D detection tasks. Lu et al. [50] propose a generative probabilistic model for concurrent action tracking and recognition. Ikizler et al. [51] employ a “tracking-by-detection” method in association with Felzenszwalb’s human detector [52] for action detection. Raptis et al. [9] use trajectory clusters as salient spatio-temporal structures for parts of an action. These parts are then represented using a graphical model that incorporates individual as well as pairwise constraints. Cao et al. [5] propose to use an adaptive cross-dataset action detection approach by exploring the spatio-temporal coherence of actions.

Those systems are similar in that they require access to manually annotated localization in the training data. Shapalova et al. [10] present a weakly supervised method to localize action discrim-

inactive regions in video. However, our approach is superior in that not only do we eliminate any pre-processing but we also perform better than their results on UCF Sports dataset. We will show how our approach looks for most discriminative regions automatically and that these discriminative regions tend to correspond to the action of interest.

We also show that our recognition framework looks for actions within segments of the raw video sequence. Although previous work by Schindler et al. [53] show similar recognition techniques for short frame snippets, experiments are not only conducted on not-cluttered action datasets (e.g. KTH and Weizmann) but also rely heavily on pre-processing person detection techniques. In contrast, we show experiments on cluttered MSR II action dataset without the need for any person detection pre-processing.

2.3 Learning to Detect Objects from Weakly-Supervised Data

The task of learning to locate objects from weakly-tagged data has been considered in both [54] and [55]. In [54], a conditional random field model is constructed between images to identify regions that could contain the same object. Extending this approach to video would be complicated by the need to link frames within the video. This would lead to a large increase in the number of edges and nodes in that the CRF would be significantly increased. In [55], Pandey and Lazebnik propose adapting the deformable parts model from [52] and modifying the training strategy to locate the object. Because this approach is closely tied to the original implementation in [52], it is not clear how this would be generalized to video. Our approach also has the advantage of being a much simpler, bag-of-words based model that can be computed very efficiently. This is important for coping with the large amounts of data in video.

CHAPTER 3: LOCALIZING ACTIONS THROUGH SEQUENTIAL 2D VIDEO PROJECTIONS

In this chapter, we introduce a technique that transforms the 3D localization problem into a series of 2D detection tasks. This is accomplished by dividing the video into overlapping segments, then representing each segment with a 2D video projection. The advantage of the 2D projection is that it makes it convenient to apply the best techniques from object detection to the action detection problem. Our experiments show that video projection outperforms the latest results on action detection in a direct comparison.

Our work is strongly influenced by Yuan *et al.* [7] and Cao *et al.* [5], where action detection is performed using an efficient branch-and-bound search. Although we propose a fundamentally different representation and do not explicitly support cross-dataset training, the goals of our work are sufficiently close as to enable direct comparison. Philosophically, our notion of a 2D video projection is also related to concepts such as motion-history images [22] and spin images [56], where 3D spatio-temporal or volumetric data is transformed into a 2D space that enables efficient search or recognition. Finally, one can view video projections as enabling spatial and temporal localization with bag of visual words models by efficiently classifying spatio-temporal subregions.

The basic idea behind the proposed method is to treat the action detection task as a series of parallel localization subtasks, each examining a short chunk of the video. Each subtask is transformed into a 2D problem using video projections and solved efficiently. Finally, we connect the local detections in time using a chaining algorithm. The following subsections detail each of these steps.

3.1 Video Representation

Following Laptev *et al.* [27], we compute spatio-temporal interest points (STIPs) in each video using Harris3D detector. A subvolume $(\Delta_x, \Delta_y, \Delta_t)$ is extracted at each interest point as shown in Figure 3.1. The size of the subvolume depends on the detection scales, i.e. $\Delta_x, \Delta_y = 18\sigma$ and $\Delta_t = 8\tau$, where the σ is spatial scale and τ is the temporal scale used for computing the descriptor. Each subvolume is divided into (n_x, n_y, n_t) grid of cuboids and for each cuboid 4-bins histogram of oriented gradients (HOG) and 5-bins histogram of optical flow (HOF) are computed. Normalized histograms are concatenated into HOG and HOF descriptors. We use parameter values $n_x, n_y = 3, n_t = 2$ as suggested by authors. Eight spatial and two temporal scales are used for computing the feature descriptors.

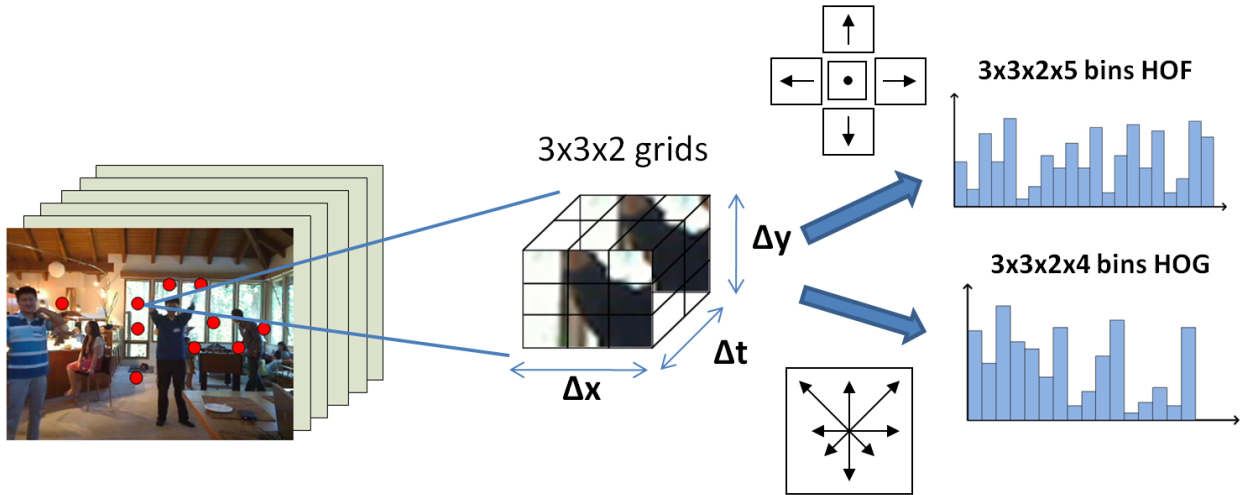


Figure 3.1: Space-time interest points (STIP) are extracted in a video sequence using Harris3D detector and HOG & HOF descriptors are computed for each STIP.

Those descriptors are quantized using a visual codebook constructed over the training set, enabling

us to represent each STIP p_j as the tuple (x_j, y_j, t_j, c_j) , denoting that a STIP was observed at (x_j, y_j) in the t_j 'th frame of video; the discrete label c_j corresponds to the codebook word nearest in feature space to p_j 's descriptor. The core assumption behind our approach (similar to that in [5, 7, 27]) is that one can recognize whether a collection of STIPs corresponds to the action of interest using a classifier that takes as its input a histogram over these discrete labels. However, we do not accumulate the features into a single histogram, as would be typical in a bag of visual words model, but rather we aggregate them in localized spatio-temporal subvolumes, as described below.

We divide a given video sequence V into a series of overlapping video chunks $\{V_1, V_2, \dots, V_N\}$ each with a temporal duration of F frames, as shown in Figure 3.2. Each chunk is represented by the collection of STIPs contained within:

$$V_m = \{(x, y, t, c) : t_s^m \leq t < t_e^m\}, \quad (3.1)$$

where $t_s^m = (m - 1) \times F/2 + 1$ and $t_e^m = t_s^m + F$. Since each STIP retains its spatio-temporal location, the goal of action detection is to find those subvolumes that contain STIPs corresponding to the action of interest. More accurately, since a given action of interest is likely to be span several chunks, we aim to identify subvolumes within each chunk that are likely to be parts of the action. Naively testing whether every possible subvolume within a chunk matches the action using an exhaustive scanning window is clearly inefficient. We propose a better approach next.

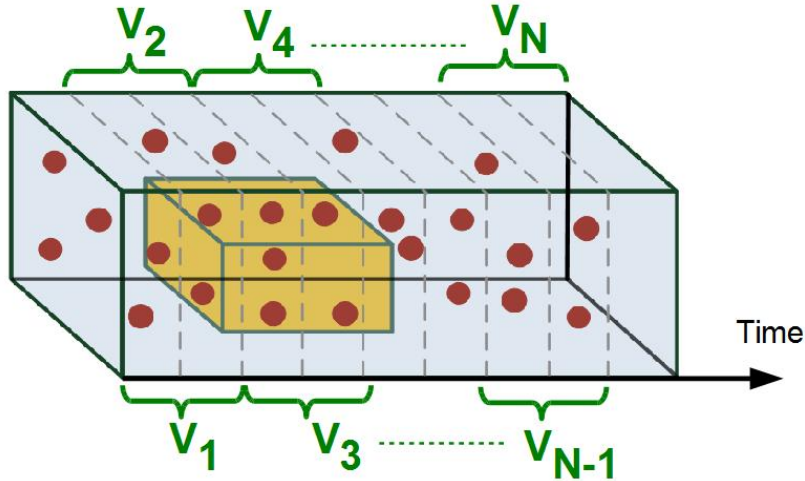


Figure 3.2: We represent a video sequence as a collection of overlapping video chunks and each video chunk by a collection of STIPs. The goal of action detection is to localize subvolumes that contain the action of interest.

3.2 Reducing Action Localization to 2D Search

An action can be modeled as a spatio-temporal bounding box (yellow volume in Figure 3.2). We refer to the subvolume of the action that is contained within a single video chunk as an action segment. Hence, we can consider an action instance as a chain of action segments contained in consecutive video chunks,

We analyze each video chunk independently to determine whether it contains an action segment. Since a chunk consists of only a small number of frames, we seek to localize the action segment only spatially within the chunk by assuming that it extends temporally throughout the chunk. Specifically, as shown in Figure 3.3, for each action segment in the chunk, we seek a subvolume cuboid of duration F frames that covers it. Since the classifier score of any cuboid in the chunk is determined solely by the STIPs contained within, we observe that rather than exhaustively considering every cuboid, we only need to consider cuboids that touch STIPs on each face. And since all

subvolumes are of duration F , it can be modeled as a 2D rectangle $[x_{\min}, y_{\min}, x_{\max}, y_{\max}]$.

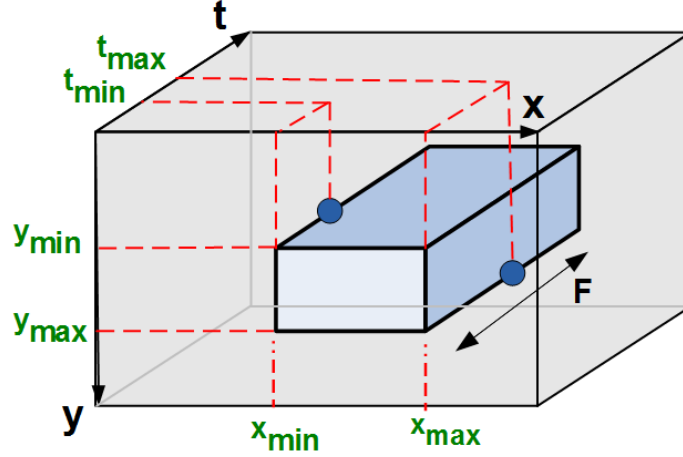


Figure 3.3: We need only consider subvolumes in a video chunk that touch a STIP on each face; through video projection, we model subvolumes as 2D rectangles.

Each subvolume is represented in the classifier by a histogram of the codebook counts for the STIPs contained within, $\mathbf{h} = [h_1, \dots, h_K]$, where h_i is the number of STIPs within the subvolume assigned to cluster center i .

$$h_i = \sum_{j=1}^N l_{ij}, \quad (3.2)$$

where

$$l_{ij} = \begin{cases} 1 & \text{if } c_j = i, \\ 0 & \text{if } c_j \neq i, \end{cases} \quad (3.3)$$

N is the number of STIPs within the subvolume and the c_j is the cluster index of the j th STIP in the subvolume. We compute the histograms for ground truth action segments that are extracted from the training videos and train a linear support vector machine (SVM) using the resulting histograms.

We define the action detection and localization problem within a video chunk as finding a subvol-

ume that would maximize the SVM classifier score, f , given by:

$$f = \beta + \sum_{i=1}^K w_i h_i, \quad (3.4)$$

where K is the number of cluster centers, h_i is the count of STIPs within the subvolume that belong to cluster center i , and w_i is the SVM weight corresponding to cluster center i . Using Equations 3.2 and 3.3 and the linearity of the scalar product, we can rewrite Equation 3.4 as follows:

$$f = \beta + \sum_{i=1}^K w_i h_i = \beta + \sum_{i=1}^K w_i \sum_{j=1}^N l_{ij} \quad (3.5)$$

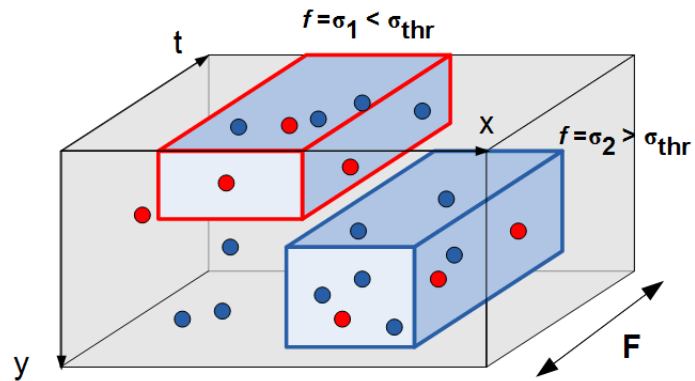
$$= \beta + \sum_{j=1}^N \sum_{i=1}^K w_i l_{ij} \quad (3.6)$$

$$= \beta + \sum_{j=1}^N w_{c_j}, \quad (3.7)$$

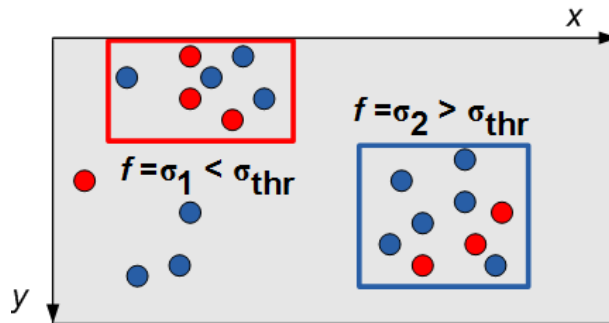
where N is the number of STIPs contained in the subvolume and w_{c_j} is the SVM weight corresponding to the cluster index c_j . Equation 3.7 says that each STIP contributes to the SVM score by its corresponding SVM weight w_{c_j} . Intuitively, some STIPs are positively associated with a given action (their SVM scores are positive) while others are negatively associated with the action (negative SVM score). Thus, the goal is to identify subvolumes containing a high sum of weights.

Since we are using fixed depth subvolumes with same duration as the video chunk and the SVM score of the subvolume depends only on the weight of each STIP within the subvolume (Equation 3.7), we can redefine the three-dimensional subvolume search problem as a two-dimensional search problem by projecting the data along the temporal dimension, as shown in Figure 3.4. Figure 3.4(a) shows two candidate subvolumes in a video chunk: the blue one has an SVM score greater than the threshold and red one has an SVM score less than the threshold. Figure 3.4(b) shows the corresponding subwindows with the same SVM scores in the projected representation.

Thus, the subvolume search problem in a video chunk V_m reduces to a subwindow (rectangle) search problem in its 2D projection.



(a) Video chunk as spatio-temporal volume.



(b) 2-D representation after video projection

Figure 3.4: Video projection converts the subvolume localization problem to a 2D search problem. Blue (Red) points correspond to STIPs with positive (negative) SVM weights.

3.3 Subvolume Search in 2D Video Projection

In the previous section, we show how the subvolume search problem in a video chunk could be reduced to a subwindow search problem in the 2D projection of the video chunk. We can use any two dimensional search method to find rectangular regions of interest whose SVM scores, as given

by Equation 3.7, would exceed a specified threshold. One obvious candidate is Lampert *et al.*'s Efficient Subwindow Search (ESS) algorithm [6]. However, given the sparsity of STIP features in our chunks, we propose a new fast method, Two-Point Subwindow Search (TPSS) that is well-suited for this task. Both are detailed below.

3.3.1 Efficient Subwindow Search

Efficient Subwindow Search (ESS), as proposed by Lampert *et al.* [6], was designed for efficient object localization in images. ESS uses branch-and-bound algorithm to find a region of interest with global maximum for a given quality function f . The parameter space is the set of all possible rectangles in an image. Each rectangle is represented by its top, left, bottom, right coordinates: (t, b, l, r) . A candidate subset of rectangles in the parameter space is represented as tuples $[T, B, L, R]$, where T , B , L and R define the maximum and minimum values for t , b , l and r , respectively as shown in Figure 3.5.

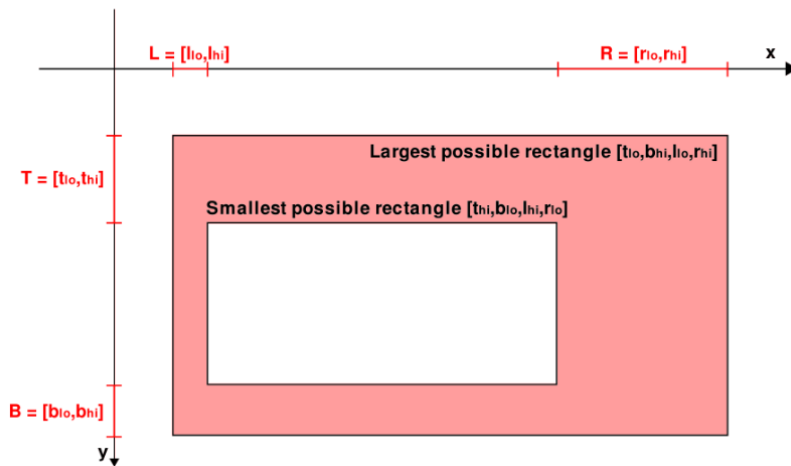


Figure 3.5: A candidate subset of rectangles in the parameter space is represented as tuples $[T, B, L, R]$, where T , B , L and R define the maximum and minimum values for t , b , l and r , respectively. Figure is courtesy of Lampert *et al.*'s CVPR 2008 talk on ESS.

In order to use the ESS algorithm, we need to define a quality function f that we seek to maximize and a function \hat{f} such that:

$$\hat{f}(\mathbf{R}) \geq \max_{R \in \mathbf{R}} f(R) \quad (3.8a)$$

$$\hat{f}(\mathbf{R}) = f(R), \text{ if } R \text{ is the only rectangle in } \mathbf{R} \quad (3.8b)$$

In our problem, f as defined in Equation 3.7 serves as the quality function for ESS to maximize; we also define the bounding function \hat{f} as follows:

$$\hat{f}(\mathbf{R}) = f^+(R_{\max}) + f^-(R_{\min}), \quad (3.9)$$

where R_{\max} is the largest rectangle and R_{\min} is the smallest rectangle contained in \mathbf{R} , f^+ is computed using only those points with positives SVM weights and correspondingly, f^- using only negative weights in R .

Once we define the quality and bounding functions we can use ESS to search a rectangular region with the highest SVM score. Algorithm 1 (adapted from [6]) provides pseudo-code for ESS. For each rectangle set, an upper bound is computed for the highest score that the quality function f could take on any of the rectangles in the set. ESS terminates when it has identified a rectangle with a quality score that is at least as good as the upper bound of all remaining candidate regions. In order to detect multiple action instances within the same video chunk, we run Algorithm 1 multiple times, removing the detected rectangle from I at every iteration until the SVM score of the detected rectangle falls below the threshold.

Algorithm 1 Efficient Subwindow Search

```
//  $I_{m \times n}$  : 2D video projection
//  $\hat{f}$ : Quality bounding function

Initialize  $P$  as empty priority queue
set  $[T, B, L, R] = [0, n] \times [0, n] \times [0, m] \times [0, m]$ 
repeat
   $[T, B, L, R] \rightarrow [T_1, B_1, L_1, R_1] \cup [T_2, B_2, L_2, R_2]$ 
  push  $([T_1, B_1, L_1, R_1], \hat{f}([T_1, B_1, L_1, R_1]))$  into  $P$ 
  push  $([T_2, B_2, L_2, R_2], \hat{f}([T_2, B_2, L_2, R_2]))$  into  $P$ 
  retrieve top state  $[T, B, L, R]$  from  $P$ 
until  $[T, B, L, R]$  consists of only one rectangle
set  $R_{\max} = [T, B, L, R]$ 
```

3.3.2 Two Point Subwindow Search

Even though, theoretically, the search space for an $M \times N$ image using the sliding window approach is M^2N^2 , in practice there are only sparse number of STIPs with non-zero SVM weight and only a small fraction of those are positive. Using this observation, we propose a new search method called Two-Point Subwindow Search (TPSS). In this search method, two STIPs with positive SVM weights define a rectangle, with STIPs at opposite corners as shown in Figure 3.6. The SVM score of such rectangular regions can be efficiently computed using an integral image [57]. Our proposed search method has two stages, summarized in Algorithm 2: (1) we compute the SVM score for all candidate rectangles that are defined by pairs of STIPs, each with positive SVM weight, if the SVM score is over a threshold, then the rectangle is considered as a detection; (2) we perform a non-maximum suppression algorithm on the detection set to eliminate overlapping rectangles.

The primary benefit of the first stage is that the set of candidates (rectangles defined by two STIPs with positive SVM weights) is much smaller than the number of sliding windows (or even rectangles defined by all STIPs). Intuitively, the TPSS algorithm restricts its search to promising

rectangles. For instance, consider a rectangle bounded by a STIP with a negative score; shrinking such a rectangle so as to exclude this STIP should result in improving the score.

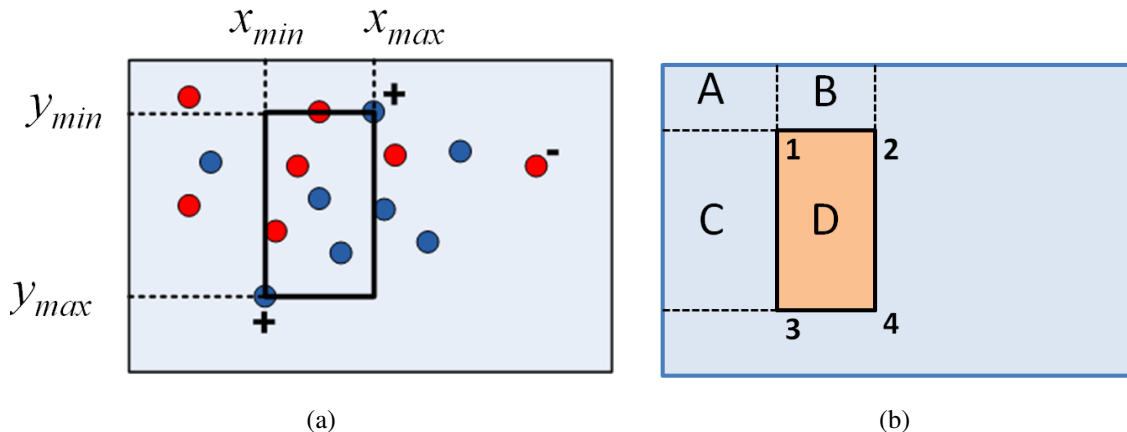


Figure 3.6: (a) Two STIPs with positive SVM weights define a rectangle with STIPs at opposite corners. We search over all such possible rectangles followed by non-maximal suppression. (b) Integral image [57] is used for efficiently computing the SVM score of any rectangle using only four operations. The score of area D is computed as $\bar{4} - B - C + A$, where $\bar{4}$ is the accumulated sum at point 4.

The second stage performs non-maximum suppression to reduce the number of redundant detections. During non-maximum suppression, we first select the rectangle with the highest SVM score, i.e., R_{\max} , from the detection set. Then, we identify and remove all the rectangles from the detection set that are connected to R_{\max} . Two rectangles R_1 and R_2 are treated as connected if both $(R_1 \cap R_2)/R_1$ and $(R_1 \cap R_2)/R_2$ are greater than a threshold ρ , where \cap denotes the operator used to compute the overlap volume. Once we find the set of rectangles that are connected to R_{\max} , we replace R_{\max} with a bounding box that contains all rectangles that are connected to R_{\max} , and push the updated R_{\max} to the final detection list. We repeat the process by selecting the next rectangle with the maximum SVM score from the remaining rectangles until no rectangles remain in the

detection set. Thus, unlike ESS, TPSS does not require multiple reruns to detect multiple action instances in a video chunk.

Algorithm 2 Two-Point Subwindow Search

// p^+ : STIP with positive SVM weight
// R_{ij} : The subwindow defined by STIPs p_i^+ and p_j^+
// $f(R)$: The SVM score of the subvolume R
// R_{\max} : Subwindow which maximizes f
// D_n : List of detections in I_n
// I_n : 2D projection of video chunk V_n

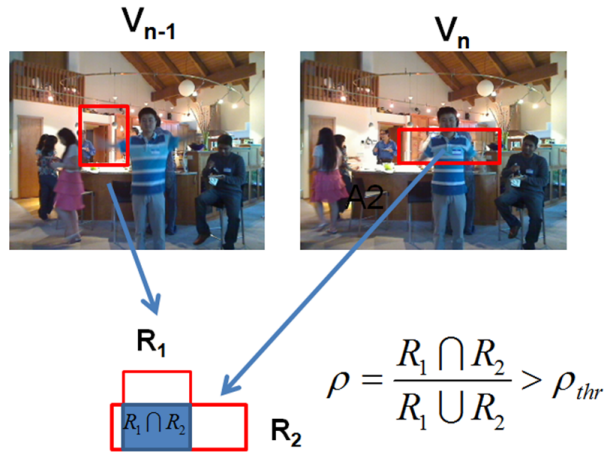
for $\forall (p_i^+, p_j^+) \in I_n$ where $i \neq j$ **do**
 $R_{ij} \leftarrow (p_i, p_j)$
 Compute the SVM score f for R_{ij} using Eqn. 3.7
 if $f >$ threshold **then**
 Add R_{ij} to the detection list D_n
 end if
end for

Non-maximum Suppression:

while D_n is not empty **do**
 $R_{\max} \leftarrow \arg \max_R f(R)$
 Move R_{\max} to the local maximum list L_n
 for $\forall R \in D_n$ **do**
 $\rho_1 \leftarrow \frac{|R \cap R_{\max}|}{|R_{\max}|}$
 $\rho_2 \leftarrow \frac{|R \cap R_{\max}|}{|R|}$
 if ρ_1 and $\rho_2 \geq 0.5$ **then**
 $R_{\max} \leftarrow R_{\max} \cup R$
 Remove R from the detection list, D_n
 end if
 end for
end while



(a)



(b)

Figure 3.7: (a) Action segments detected in subsequent video chunks. (b) Two action segments belong to the same action instance, i.e. connected, if intersection over union volume of the action segments is greater than a threshold.

3.4 Chaining Subvolumes Across Time

Once subvolumes are detected in each video chunk, either using ESS or TPSS, a greedy chaining strategy is used to connect the detections in consecutive video chunks together to form a complete action. We start the search with the first video chunk and select the detection with the highest SVM score. Then, we find all the detections C from the next video chunk that are connected to the currently selected detection. We treat two detections in adjacent video chunks as connected if their

volume overlaps by a certain threshold (see Figure 3.7). We select the detection with the highest SVM score from the connected detections list and continue our search with the next video chunk until there are no more connected detections left in the following video chunks (Algorithm 3).

Algorithm 3 Chaining Subvolumes Across Video Chunks

// ActionSet : The set of action detections
// Action: List of connected action segments
// D_n: List of subvolume detections in V_n
// R_{max}: Subvolume detection maximizing SVM score
// C: List of rectangles from D_{n+1} connected to R_{\max}

Initialize *ActionSet* as an empty set

```

for  $n = 1$  to  $N$  do
  if  $D_n \neq \emptyset$  then
    Action := {}
     $R_{\max} \leftarrow \arg \max_{R \in D_n} f(R)$ 
    Delete  $R_{\max}$  from  $D_n$ 
    Add  $R_{\max}$  to ActionTrace
     $i \leftarrow 1$ 
     $C \leftarrow \{R : R \in D_{n+i}, \rho_1 \text{ and } \rho_2 \geq \rho\}$ 
    while  $C \neq \emptyset$  do
       $R_{\max} \leftarrow \arg \max_{R \in C} f(R)$ 
      Delete  $R_{\max}$  from  $D_{n+i}$ 
      Add  $R_{\max}$  to Action
       $i \leftarrow i + 1$ 
       $C \leftarrow \{R : R \in D_{n+i}, \rho_1 \text{ and } \rho_2 \geq \rho\}$ 
    end while
    Add Action to ActionSet
  end if
end for

```

3.5 Computational Complexity

The worst case complexity of the naive 3D branch-and-bound search for detecting a single instance of an action is given by $O(m^2n^2t^2)$ for a $m \times n \times t$ video. Overall complexity of our proposed

method for detecting a single instance of an action using the TPSS is given by $O(N_p^2 \times t/F) + O(t/F)$, where N_p is the number of STIPs contained in the video chunk with positive SVM weight and F is the duration of the video chunks. The first term in this equation is the complexity of the TPSS algorithm and the second term is the complexity of chaining the subvolumes across time. Since $N_p \ll m \times n$, the upper bound of our algorithm is given as $O(m^2n^2t) + O(t) = O(m^2n^2t)$.

In order to detect multiple instances of actions, the 3D branch-and-bound search needs to be run multiple times. Assuming that there are k instances of actions in a given video, the worst case complexity of 3D branch-and-bound search becomes $O(m^2n^2t^2k)$. On the other hand, the TPSS does not require reruns to detect multiple instances of actions and the complexity of the first part of our algorithm does not change. However, the worst case complexity of chaining the subvolumes becomes $O(k^2t)$ since there can be at most k action segments in each video chunk. Hence, the overall complexity of our method for detecting k instances of actions is $O(m^2n^2t) + O(k^2t)$. Realistically, the number of instances of actions in a video is much less than $m \times n$ and the overall complexity of our method is given by $O(m^2n^2t)$ when $k < m \times n$. Moreover, since our method searches each video chunk separately, the 2D search can be performed in parallel. The Table 3.1 summarizes the worst case complexity comparison of our method and the 3D branch-and-bound search.

Table 3.1: Worst case complexity comparison of our method and naive 3D branch-and-bound search for detecting actions in a given video sequence.

	Naive 3D B&B	Our Method
1 action	$O(m^2n^2t^2)$	$O(m^2n^2t)$
k actions	$O(m^2n^2t^2k)$	$O(m^2n^2t)$

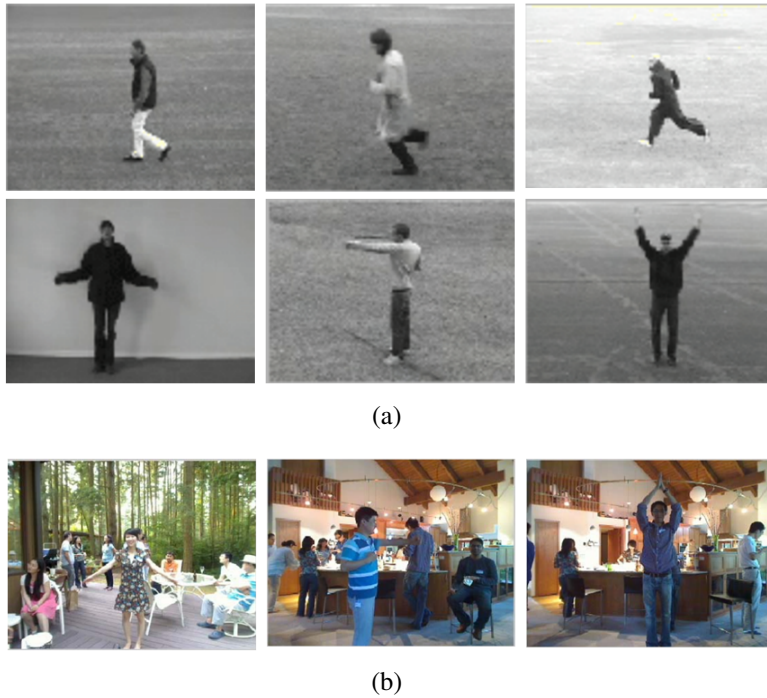


Figure 3.8: (a) Sample video frames from KTH dataset. (b) Sample video frames from MSR dataset. Different than KTH dataset, there are other activities happening in the background.

3.6 Experimental Results

Our experimental methodology follows that of Cao *et al.* [5] to facilitate direct comparisons. We use the KTH [1] and Microsoft Research Action Dataset II (MSR) [5] in our experiments. KTH contains videos of six action types: boxing, hand waving, hand clapping, walking, jogging and running, performed by a single actor against a relatively fixed background. MSR contains three action types: hand waving, hand clapping and boxing, performed in a more challenging setting with multiple users in a cluttered and dynamic scene. By design, the three actions in MSR are the same as those in KTH, to explore cross-dataset performance of action detection algorithm. The MSR dataset contains 54 video clips, with each clip exhibiting several instances of each action type (71 waving, 51 clapping and 81 boxing action instances). Figure 3.8 shows sample video

frames from MSR and KTH datasets.

Following [5], our training set consists of the waving, clapping and boxing clips from KTH augmented with four randomly-selected video clips from the MSR dataset. The testing dataset consists of the remaining videos in MSR. We first construct a standard vocabulary using K-means clustering ($K = 1000$) on HNF descriptors computed at space-time interest points (STIP) [35] extracted on the training set, where the descriptors are a compound of HOG and HOF [27] features.

Next, we train a set of linear one-vs-all SVM classifiers using videos from KTH and the ground truth volumes from the MSR training subset. For each, we extract overlapping video chunks with durations of F frames ($F = 32$ is typical) and compute a bag-of-video-words histogram by accumulating the counts of HNF descriptors in the volume, quantized using the above dictionary. We employ randomly-extracted video chunks of the same size as negative examples and expand this negative sample set using bootstrapping [58]. Figure 3.9 summarizes the training and test of our proposed action detection model.

For comparing our results to the ones in Cao *et al.* [5] we use the same precision and recall criteria. For the precision score, a detection is regarded as a true positive if at least 1/8 of its volume overlaps with that of the ground truth. For recall, the ground truth label is regarded as retrieved if at least 1/8 of its volume is covered by one of the detected volumes. The precision and recall values are computed using Equation 3.10 for different SVM thresholds to generate P-R curves, as shown below.

$$\text{Precision} = \frac{\text{Number of True Positives}}{\text{Number of Detections}} \quad (3.10a)$$

$$\text{Recall} = \frac{\text{Number of Retrieved Labels}}{\text{Number of Labels}} \quad (3.10b)$$

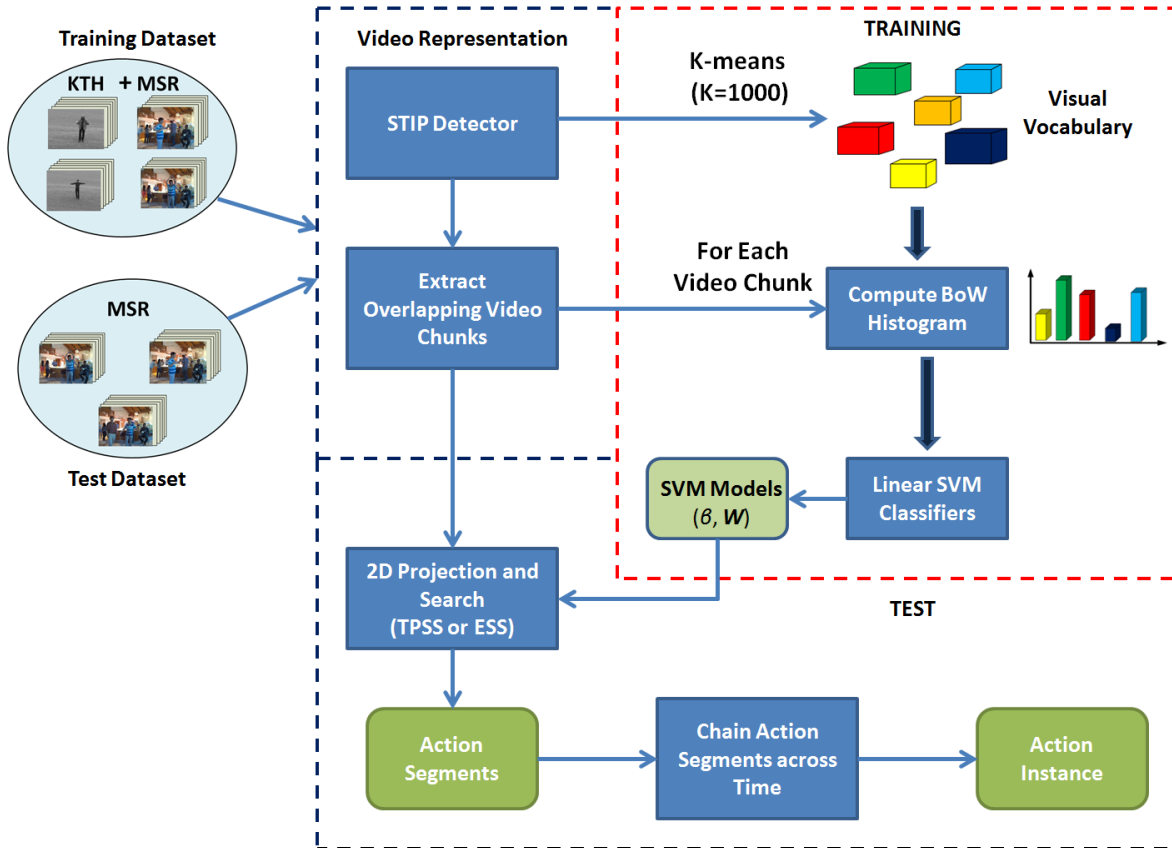


Figure 3.9: Summary of our training and test methodology.

3.6.1 Cross-Dataset Action Detection Comparisons

Figure 3.10 shows the precision-recall curves for Cao *et al.* [5] and the two search variants of the proposed video projection (VP) method: VP+ESS and VP+TPSS. We make several observations. First, we see that in general, both VP methods outperform [5], particularly in the high recall regime. Second, we observe that the proposed two-point subwindow search (TPSS) appears to be slightly better than ESS [6] on this task, in spite of its simplicity. Finally, we note that even though our

approach is not explicitly designed with cross-dataset action detection in mind, it surprisingly outperforms [5] on the task.

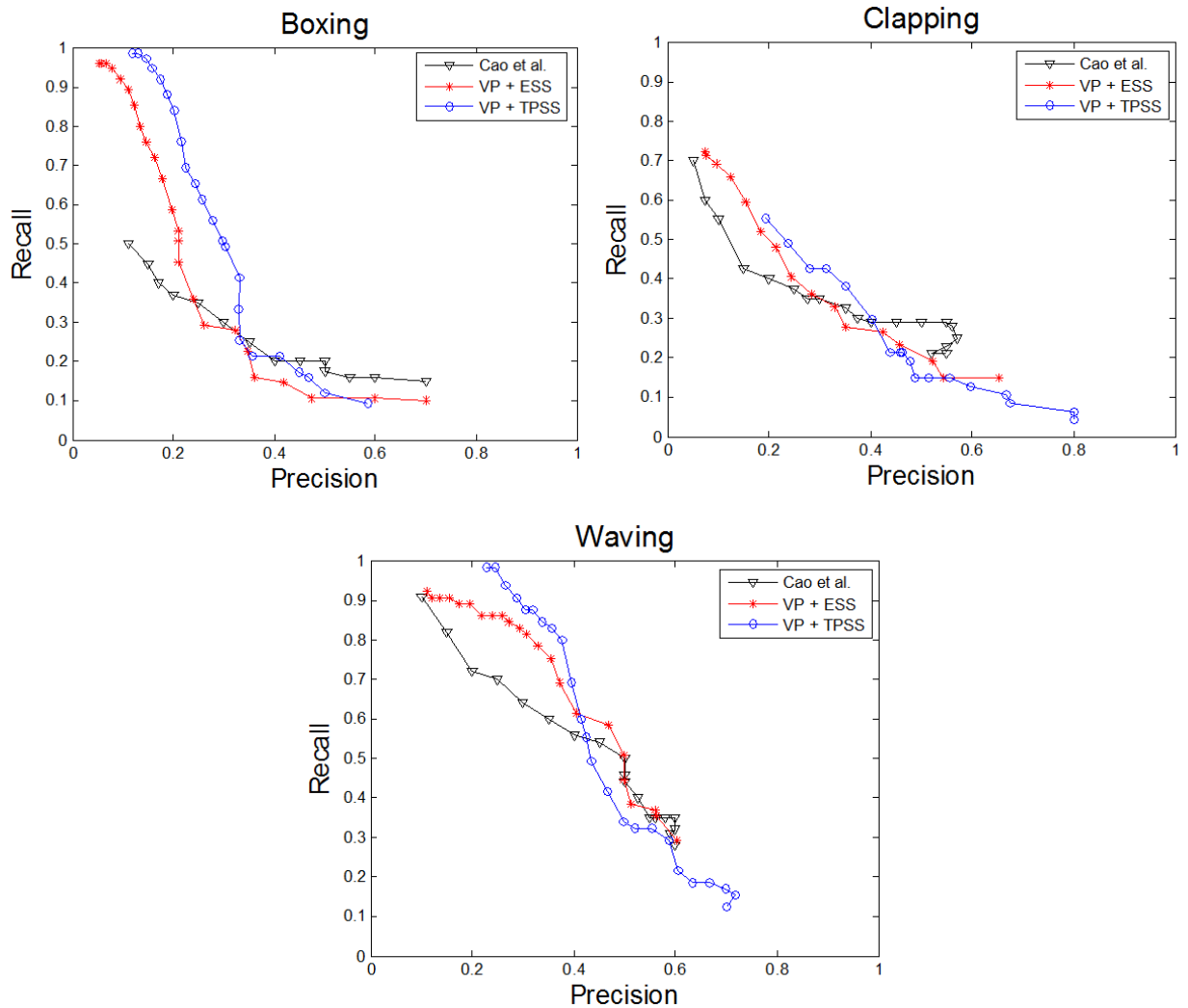


Figure 3.10: Direct comparison on MSR Action II, trained using KTH+4 clips of MSR. Both variants of proposed method, VP+ESS and VP+TPSS, outperform Cao *et al.* [5] despite their simplicity.

3.6.2 Additional Results

Our last set of experiments explores a few remaining questions, such as: how well does our method perform with shorter-duration video chunks (smaller values of F)? Figure 3.11 shows the precision/recall curves for the waving action (trained as in Section 3.6.1) for different chunk lengths. We observe that while the performance is reduced for smaller window sizes, it is still reasonable. This indicates that the proposed approach should be suitable for online recognition systems, where low latency is essential; in such settings, a detection can be flagged using only a portion of the action of interest. Finally, Figure 3.12 shows examples of action detections on the MSR Actions II dataset using VP+TPSS.

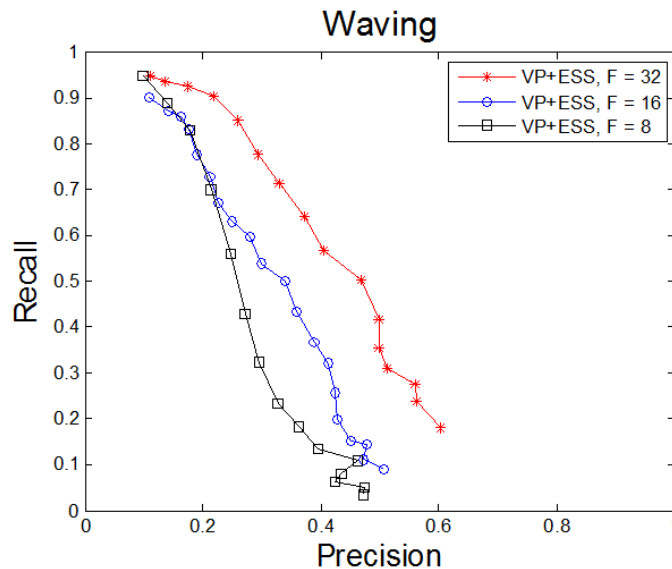


Figure 3.11: VP+ESS on the waving action for different video chunk sizes. The system performs best when longer chunks are used, but performance is acceptable if smaller chunks are needed, such as for on-line recognition.



Figure 3.12: Examples of action detections on MSR dataset using VP+TPSS. Colored boxes denote detected actions: boxing (blue), clapping (green), and waving (red). Qualitatively, our results are comparable to Cao *et al.* [5].

CHAPTER 4: A WEAKLY-SUPERVISED PROBABILISTIC MODEL FOR ACTION LOCALIZATION AND RECOGNITION

In the previous chapter, we introduced an efficient method for detecting actions by transforming the 3D localization problem into a series of 2D detection tasks. The drawback of this method and other supervised detection methods [5,7–9,47,52] is that they require access to manually annotated action locations in the training data. This is feasible on small datasets like UCF Sports and MSR II, however it is highly costly and cumbersome on larger, more complex datasets such as UCF101 and HMDB.

In this chapter, we first present a weakly-supervised probabilistic model for detecting actions by localizing discriminative sub-regions in datasets where each video may contain multiple instances of actions. While our first system introduced in the previous chapter requires every action to be manually outlined in every frame of the video, this second system only requires that the video be given a single high-level tag. From this data, the system is able to identify discriminative sub-regions that correspond well to the regions containing the actual actions.

Next, we extend the above probabilistic model for finding the discriminative sub-regions to the action recognition problem where the video clips have been tightly trimmed to contain only one instance of an action. The goal of our proposed model is to extract most discriminative sub-regions within a video sequence and then aggregate them for the final action classification. This is different than standard BoW model where all the features contained in a video are accumulated into a single histogram. Our model is trained in a weakly-supervised manner: training videos are annotated only with training label without any action location information within the video. Our localization experiments on UCF Sports dataset show that the discriminative sub-regions produced by this weakly supervised system are comparable in quality to action locations produced by systems that require

training on datasets with fully annotated location information. Furthermore, our classification experiments on UCF Sports and two other major action recognition benchmark datasets, HMDB and UCF101, show that our recognition system significantly outperforms the baseline BoW models and is better than or comparable to the state-of-the-art.

The rest of this chapter is organized as follows. Section 4.1 provides the overview of the video representation. Section 4.2 introduces our weakly-supervised model for detecting actions in datasets where each video clip may contain multiple instances of actions. Section 4.3 extends the localization model to recognize actions in datasets where each video has been tightly trimmed to contain only one instance of an action.

4.1 Overview of Video Representation

Video data is represented using HOG-HOF descriptors computed at densely sampled interest points using Laptev’s STIP detector [37]. Building on the success of bag-of-words approaches, the descriptors are represented in a standard vector quantization representation and then clustered to construct a visual codebook of size 4000 using K -means clustering. Each STIP p_j is represented as the tuple (x_j, y_j, t_j, c_j) , denoting that a STIP was observed at (x_j, y_j) in frame t_j of the video; the label c_j corresponds to the index of the visual word in the codebook that is closest in feature space to p_j ’s descriptor.

The densely sampled interest point option for the STIP detector returns interest point locations based on the spatial and temporal scale sizes used. These locations are highly sparse, making it possible to compact the images and significantly reduce the amount of computation necessary to localize discriminative sub-regions. Each feature descriptor captures the information contained in a rectangular cuboid centered at the interest points and a single 2D image can represent the quan-

tized STIP features for a given range of actual video frames as shown in Figure 4.1. For brevity, we will use the term frame to denote this 2D image.

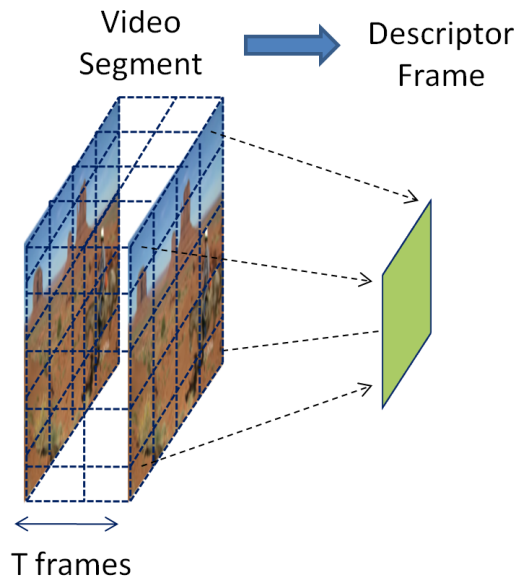


Figure 4.1: Interest points (STIPs) are sampled at dense space time intervals and a single 2D image can represent the quantized STIP features for a given range of the action video frames. We use the term ‘frame’ to represent the descriptor images.

4.2 A Weakly-Supervised Model for Action Localization

In this section, we introduce a probabilistic model for detecting actions by localizing discriminative sub-regions in videos where each video may contain multiple instances of actions. While our first detection system introduced in the previous chapter requires every action to be manually outlined in every frame of the video, this second system only requires that the video be given a single high-level tag. Our experiments on MSR Action Dataset II show that the localizations produced by this weakly supervised system are comparable in quality to localizations produced by systems that

require each frame to be manually annotated.

4.2.1 Probabilistic Model for Localizing Discriminative Sub-Regions

A localizer identifies the sub-regions in the frame of quantized descriptors that are the most discriminative for classifying an action. Building on the success [6], the discriminative power of each sub-region is computed with a linear discriminator based on the histogram of quantized features, or bag-of-words, in the sub-region. This can be implemented probabilistically with a distribution similar to the softmax activation function. For now, we will focus on localizing within a single frame of descriptors, which corresponds to several frames of raw video data, then expand to complete videos below in Section 4.2.3. If R_f denotes the set of all possible sub-regions in frame f , then the probability that the sub-region r is the most discriminative region, $p^f(r; \phi)$ is defined as:

$$p^f(r; \phi) = \frac{\exp(\phi^\top \mathbf{h}_{f,r})}{\sum_{r' \in R_f} \exp(\phi^\top \mathbf{h}_{f,r'})}. \quad (4.1)$$

In this distribution, $\mathbf{h}_{f,r}$ denotes the histogram describing the frequency of visual words in the sub-region r contained in frame f and ϕ contains localization parameters. The R_f denotes the set of all possible sub-regions in the frame, however, we only used full size, three-quarter-sized, half-sized and quarter-sized sub-regions w.r.t. the frame size in our computations. We have experimented with including smoothness terms between frames and computing the marginal distribution for each frame, but this did not improve accuracy. A significant advantage of this linear scoring function is that it can be computed efficiently using the integral image representation, similar to [6].

4.2.2 Incorporating Weakly Supervised Tags

Because this system is weakly supervised, the only source of localization information is the label assigned to each clip. Thus, the localization model must be extended to include a class label. This will make it possible to learn the localizer by optimizing a recognition criterion. This can be done by changing the localization model in Equation (4.1) from a distribution over the most discriminative sub-regions in each frame into a joint distribution over sub-region locations and a video label, c . We thus introduce one localizer per label and altering the distribution so that the parameters used depend on the label, c . Again, letting r denote the most discriminative sub-region in a frame, the joint probability over r and c is defined as

$$p^f(r, c; \phi_1, \dots, \phi_C) = \frac{\exp(\phi_c^\top \mathbf{h}_{f,r})}{\sum_{c' \in C} \sum_{r' \in R_f} \exp(\phi_{c'}^\top \mathbf{h}_{f,r'})} \quad (4.2)$$

where there are C possible labels, and ϕ_1, \dots, ϕ_C are the localizers associated with each of the C classes (Figure 4.2). This distribution can be interpreted as specifying the probability that the label c and sub-region r are the best combination of sub-region and label that describes the action in the frame.

4.2.3 Detecting Actions

With this basic formulation, it is possible to create a weakly-supervised action detection system for video data that may contain multiple simultaneous actions. Building on the single frame model described in the previous section, this detection method will detect the action using just the descriptors from one segment of video. Depending on the number of frames in the descriptor, this means that each frame of descriptors will represent T frames of raw video data.

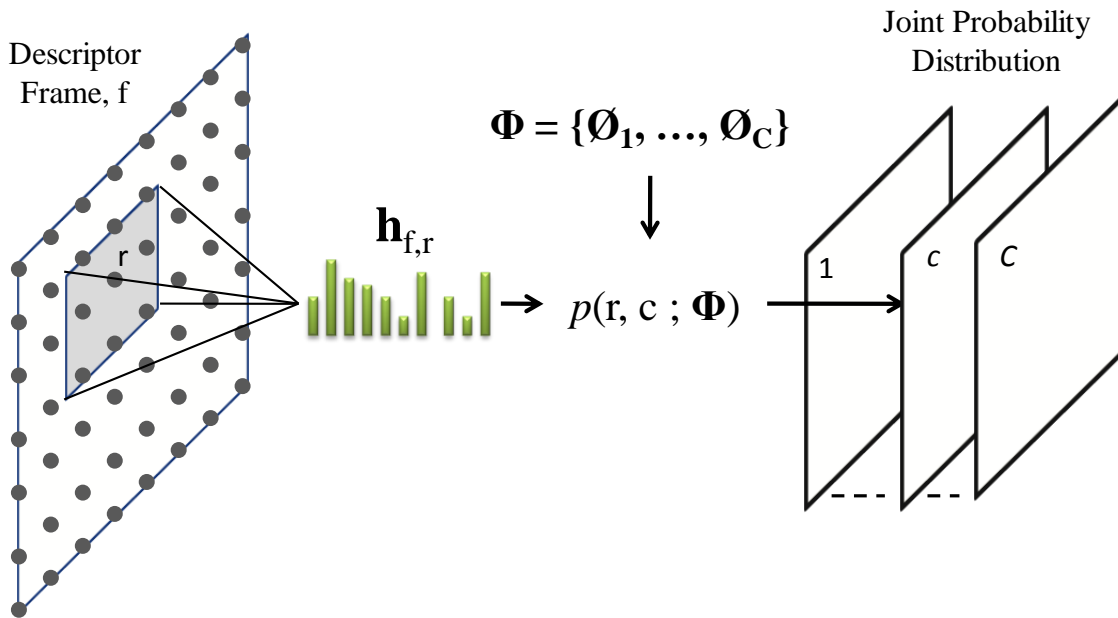


Figure 4.2: Each sub-region, r , in a descriptor frame is represented as histogram of quantized features contained in the sub-region and the joint probability distribution between sub-regions and action labels are computed.

While previous work, such as [5], searches over variable length sub-volumes in the video, detecting from a fixed set of frames makes it possible to apply this method to streaming video. In addition, both [53] and our experiments below show that competitive detection results can be achieved from short snippets of data.

Using a dataset, like the MSR Action Dataset II used in [5], this system can be trained by processing the video into frames of descriptors, then treating the frames as independent samples. Each frame is tagged with a label specifying which actions is present. Frames that includes multiple actions are duplicated for each of the actions present in the frame. Treating the frames as independent samples does lose temporal coherence, but greatly simplifies the application to streaming

data. We have experimented with the incorporation of CRF models to capture temporal coherence, but did not find significant differences in performance.

The learning criterion is created by taking the negative log of the marginal distribution over c , then summing this negative log over the frames. For a set of N_f training samples and associated labels l_1, \dots, l_{N_f} , the loss function is defined as:

$$L(\Phi) = - \sum_{f=0}^{N_f} \log(P(T_f = l_f | \Phi)) + \frac{1}{2} \epsilon \sum_{c=1}^C \|\phi_c\|^2 \quad (4.3)$$

where $\Phi = \{\phi_1, \dots, \phi_C\}$ are the sub-region localizer weights, T_f is the predicted action label and the l_f is the ground truth action label of sample frame f . The probability $P(T_f = l_f | \Phi)$ is computed by marginalizing the joint distribution in Equation 4.2 over r :

$$P(T_f = l_f | \Phi) = \frac{\sum_{r \in R_f} \exp(\phi_{l_f}^\top \mathbf{h}_{f,r})}{\sum_{c \in C} \sum_{r \in R_f} \exp(\phi_c^\top \mathbf{h}_{f,r})} \quad (4.4)$$

Using Equation 4.4 in Equation 4.3, the final loss for N_f examples becomes:

$$L(\Phi) = - \sum_{f=0}^{N_f} \left(- \log \sum_{r \in R_f} \exp(\phi_{l_f}^\top \mathbf{h}_{f,r}) + \log \sum_{c=1}^C \sum_{r \in R_f} \exp(\phi_c^\top \mathbf{h}_{f,r}) \right) + \frac{1}{2} \epsilon \sum_{c=1}^C \|\phi_c\|^2 \quad (4.5)$$

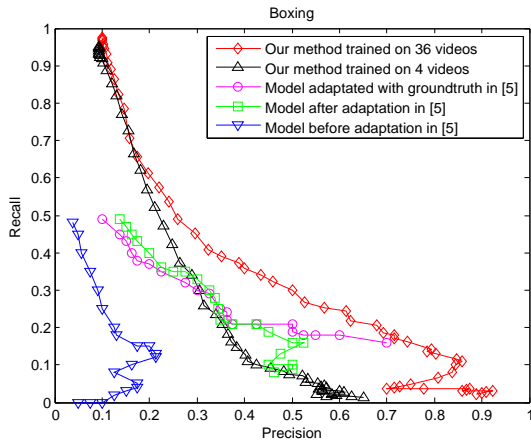
A significant advantage of this straightforward probabilistic formulation is that is easily implemented using standard optimization packages. We train the localization parameters, $\Phi = \{\phi_c\}$, using conjugate gradient descent optimization package.

4.2.4 Action Detection Experiments

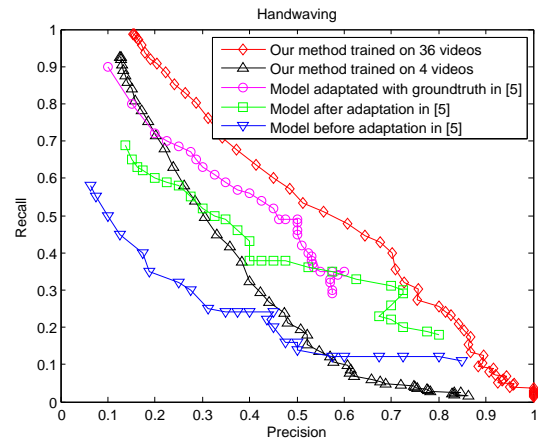
Tested on the MSR Action Dataset II [5], the discriminative sub-regions chosen by our weakly-supervised learning approach match the ground-truth labeling of the videos well. The MSR Action Dataset II consists of 54 video sequences recorded in crowded scene. There are 3 type of actions (boxing, handclapping and handwaving) in this dataset and each video contains multiple instances of actions. There are 203 action instances in total.

While [5] began by training from the KTH Action dataset, then used videos from the MSR-II dataset to adapt the detectors. We found that beginning with the KTH dataset did not help our system, so instead we trained directly on four videos from MSR-II as [5] did for domain adaptation, so that both our system and [5] both used the same amount of data from MSR-II. To examine how more data benefits our localization performance, we also randomly split the 54 video sequences into 3 groups and measure our detection performance with a 3-fold cross validation test.

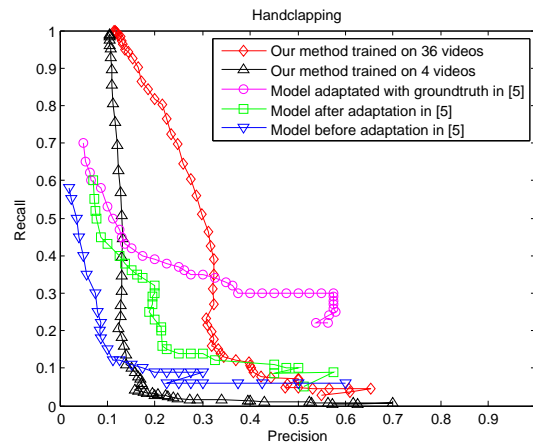
In detection stage, we perform a per-frame localization and obtain a bounding box for each frame. Following [5]’s evaluation setup, we calculate Precision-Recall curves based on different probability thresholds. For the precision score, a detection is regarded as a true positive if at least 1/8 of its volume overlaps with that of the ground truth. For recall, the ground truth label is regarded as retrieved if at least 1/8 of its volume is covered by one of the detected volumes. As Figure 4.3 shows, our system produces acceptable results when trained with just four videos, but produces results that improve on [5] when given more access to training data. Our method gives reasonable localization results on most of the videos as shown in Figure 4.4.



(a)

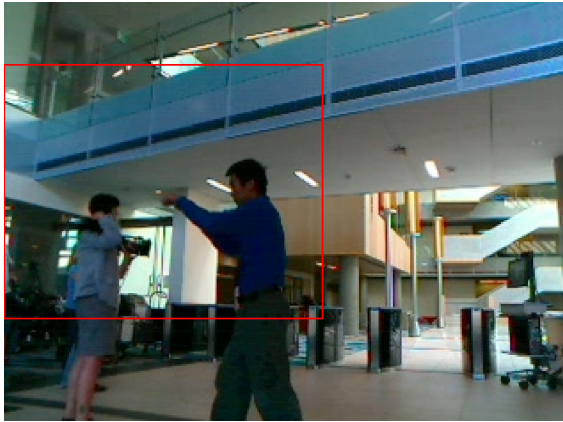


(b)

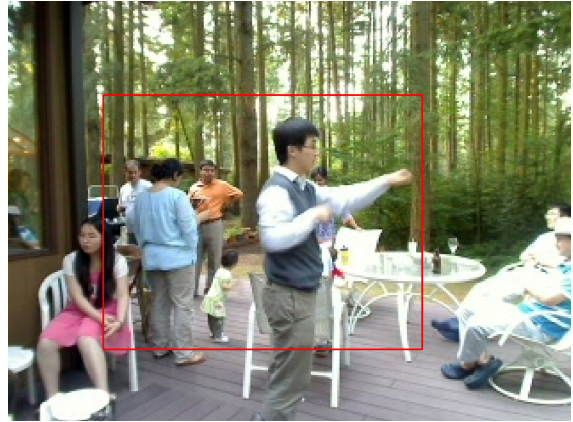


(c)

Figure 4.3: Action detection on MSR-II compared with [5]. Black curves show our result trained with randomly selected 4 videos and Red curves show our detection result trained with 3-fold cross validation. Training using just 4 videos, our performance is acceptable. With more training videos, the localization improves significantly and outperforms [5].



(a)



(b)



(c)



(d)

Figure 4.4: Examples of action detections on MSR dataset. Our method gives reasonable localization results on most of the videos.

4.3 Action Recognition by Discriminative Sub-Region Localization

The model in Section 4.2 is designed for videos where the action in each frame may be different. In current research, whole-clip classification is a popular alternative. In these datasets, such as UCF Sports [2], the training and test data are made up of short clips that have been tightly trimmed temporally to contain only the action. Each of these clips are assigned a single label that represents the action in that clip.

In this section, we show how the probabilistic model for finding discriminative sub-regions can be extended to video databases with one label assigned to short clips. Because a single label is applied to multiple frames in the clip, the role of the localizer must be adapted. Our goal in this new framework is to build on techniques that have proven to be robust for action recognition. If the localizers are used to produce global histograms that represent different sub-volumes, the recognition system can be built on the bag-of-words recognition model that has proven successful in action recognition.

4.3.1 Model Implementation

Figure 4.5 shows our proposed weakly-supervised framework for localizing discriminative sub-regions and recognizing actions. The goal of our system is to extract most discriminative sub-regions within a video sequence and then aggregate them for the final action classification.

Given a video sequence, the classification system uses a set of localizers to find the most discriminative sub-regions in the video necessary to classify the action. Each of these sub-regions is represented via a visual words histogram. These histograms are then aggregated across frames to construct a video level histogram for each localizer. In order to incorporate nonlinearity into our model, we use Kernel Map technique [16] to transform the histograms to a high-dimensional

feature space, where linear dot product approximates Histogram Intersection Kernel (HIK). Using these high-dimensional features, we then compute the joint probability for all combinations of discriminative sub-region localizers and action classes. Marginalizing this probability over localizers and picking the class corresponding to the highest probability gives us our final classification.

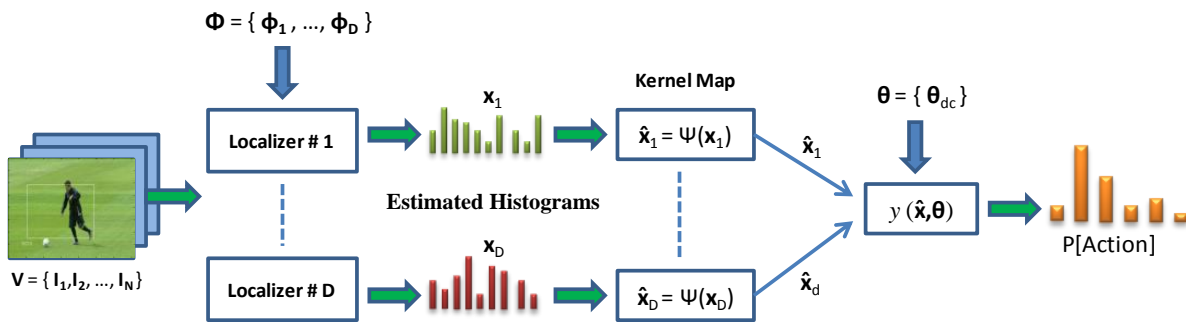


Figure 4.5: This diagram illustrates our approach for recognizing actions. First, a set of localizers estimate which sub-regions in the video frames are likely to contain the discriminative information necessary to classify the action. These candidate sub-regions are represented by histograms of visual words. Next, the feature vectors generated by the histograms are used to compute the probability of any particular combination of candidate sub-region and action class. An important, unique aspect of this approach is that the system is designed so that localization parameters (Φ) and recognition parameters (Θ) are trained to maximize the probability of the correct action class. This makes it possible to train this system in a weakly-supervised fashion.

The following are the summary of the differences between the action recognition model presented in this section and the detection model presented in Section 4.2:

- Action recognition model is used for localizing and recognizing actions from video clips where each video clip is segmented to contain only one instance of an action while the previous model is used for detecting actions (localizing in both time and space) in videos where the video may contain multiple action instances and the duration of the video sequence is longer than the duration of the action instance.

- In the detection model, each descriptor frame is considered as a separate example during training. In recognition model, the histograms extracted from each frame are accumulated to have a video level representation.
- In recognition model, the number of localizers can be different than the number of action labels. Each localizer is considered as a latent variable.
- The kernel map [16] is integrated into the action recognition framework to introduce non-linearity and therefore improve the discriminative power of the classifier.

We will provide a detailed explanation of our proposed framework shown in Figure 4.5 in the following sections.

4.3.1.1 Localizing Discriminative Sub-Regions

As shown in Figure 4.5, the first step in recognizing the action is localizing discriminative sub-regions that best describe the action. These candidates are selected using a set of D discriminative sub-region localizers. A localizer ϕ_d , learned during training (as explained in Section 4.3.2), is a vector of parameters describing the probability distribution of a latent location variable. For every sub-region in each frame in the video, each localizer computes the probability of that sub-region being the most discriminative in that frame. Each localizer has the same form as the localizer shown in Equation (4.1), i.e.:

$$p^f(r; \phi_d) = \frac{\exp(\phi_d^\top \mathbf{h}_{f,r})}{\sum_{r' \in R_f} \exp(\phi_d^\top \mathbf{h}_{f,r'})}. \quad (4.6)$$

In this distribution, $\mathbf{h}_{f,r}$ denotes the histogram describing the frequency of visual words in the sub-

region r contained in frame f and ϕ_d are the parameters for localizer d , which are learned during training.

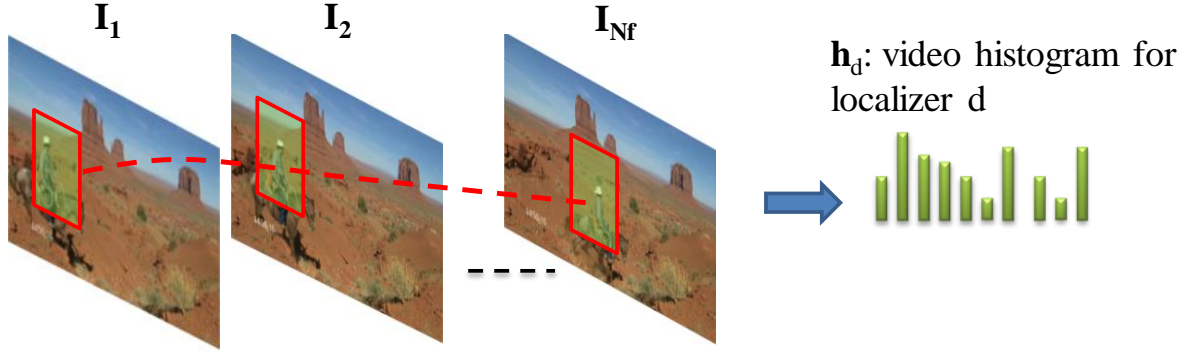


Figure 4.6: Our objective is to compute a video representation by selecting the sub-region in each video frame that maximizes the probability in Equation (4.6) and accumulating the histograms of sub-regions across all video frames.

4.3.1.2 Estimating Histograms to Represent Localized Sub-Regions

Once the sub-region probabilities of each frame are computed, our objective is to compute a video level histogram representation for each localizer. The straightforward way is to select the sub-region in each video frame that maximizes the probability in Equation (4.6) and accumulate the histograms of sub-regions across frames as shown in Figure 4.6, i.e.

$$\mathbf{h}_d = \sum_{f \in F} \mathbf{h}_{f, r^{max}} \quad (4.7)$$

where $r^{max} = \arg \max_r p^f(r; \phi_d)$. However, since the *max* operator is not differentiable, we use the sub-region probabilities to compute the estimated histograms using softmax approxima-

tion. The final feature representation for localizer $d \in D$, denoted \mathbf{x}_d , is obtained by using this expectation calculation to aggregate over all frames:

$$\mathbf{x}_d = \sum_{f \in F} \sum_{r \in R_f} \mathbf{h}_{f,r} \cdot p^f(r; \phi_d), \quad (4.8)$$

where F is the number of frames for the given video sequence, R_f is again the set of all possible sub-regions within a frame, $\mathbf{h}_{f,r}$ represents the histogram of features for sub-region r in f and $p^f(r, \phi_d)$ is computed as in Equation (4.6).

4.3.1.3 Kernel Map

The histogram intersection kernel has proven to be a popular component of a bag of visual words model for object or action recognition. Unfortunately, applying this kernel is difficult because the training data is weakly supervised. In a traditional kernel-based classifier, new examples are classified by using the kernel function to compute the dot product between transformed versions of the example and various training examples. However, the actual sub-region for each frame in the training videos is not known. While recent work has proposed a methodology for extending kernel methods to latent models, like the latent SVM model [15], the range of possible values for the latent sub-region location creates significant computational issues.

Instead, we employ the technique introduced by Vedaldi et al. [16] to approximate non-linear kernels via explicit feature maps. This enables applying the efficient learning methods for linear kernels to non-linear kernels including histogram intersection kernel. An additive kernel is defined

by

$$K(\mathbf{x}, \mathbf{y}) = \sum_{b=1}^B k(\mathbf{x}_b, \mathbf{y}_b), \quad (4.9)$$

where b is the bin index. If $k(x, y)$ is homogeneous then the additive kernel is called homogeneous i.e.

$$\forall c \geq 0 : k(cx, cy) = ck(x, y). \quad (4.10)$$

Most popular non-linear kernels used in computer vision applications, including histogram intersection and chi-square kernels, are homogeneous kernels.

A feature map $\Psi(x)$ for a kernel is defined as a function which maps x into a high-dimensional feature space as follows:

$$\forall x, y : k(x, y) = \langle \Psi(x), \Psi(y) \rangle. \quad (4.11)$$

Vedaldi et al. [16] proposed a technique using kernel signatures to analytically construct closed-form feature maps for homogeneous kernels which allows transforming the data into a format suitable for linear classifiers. Also, they propose a method using Fourier transform of the kernel signature and sampling theorem to approximate the infinite dimensional feature map $\Psi(x)$ with the finite dimensional feature map $\hat{\Psi}(x)$. The feature map approximation of the estimated histogram

\mathbf{x}_d , where d is the index of one of the D detectors, is given as follows:

$$\frac{[\hat{\Psi}(\mathbf{x}_d)]_j}{\sqrt{\mathbf{x}_d L}} = \begin{cases} \sqrt{\kappa(0)}, & j = 0, \\ \sqrt{2\kappa(\frac{j+1}{2}L)} \cos(\frac{j+1}{2}L \log \mathbf{x}_d), & j > 0 \text{ odd}, \\ \sqrt{2\kappa(\frac{j}{2}L)} \sin(\frac{j}{2}L \log \mathbf{x}_d), & j > 0 \text{ even}, \end{cases} \quad (4.12)$$

where $j = 0, 1, \dots, 2n$, n is the number of samples used in the discrete approximation, L is the sampling period, and κ is the density function which is given by

$$\kappa(\lambda) = \frac{2}{\pi} \frac{1}{1 + 4\lambda^2} \quad (4.13)$$

for histogram intersection kernel. In our experiments we set the sampling number $n = 3$ which gives a good approximation for the histogram intersection kernel.

4.3.1.4 Action Classification

After computing the kernel map transformation of the estimated histograms, we use them to estimate the probability of action labels for each video. The key problem is that the system must aggregate the information in D different localizers to produce this final probability. This aggregation is implemented using a probabilistic model. The distribution $p(d, c)$ denotes the joint probability that the video is best described by class c and contained in sub-region located by localizer d .

Using the kernel map features representations for each localizer, denoted by $\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_D$, we can

compute the joint probability distribution $p(d, c)$ as:

$$p(d, c) = \frac{\exp(\hat{\mathbf{x}}_d^\top \boldsymbol{\theta}_{dc})}{\sum_{c'=1}^C \sum_{d'=1}^D \exp(\hat{\mathbf{x}}_{d'}^\top \boldsymbol{\theta}_{d'c'})}, \quad (4.14)$$

where $\boldsymbol{\theta}_{dc}$ is the set of weights that define $p(d, c)$ and $\hat{\mathbf{x}}_d = \hat{\Psi}(\mathbf{x}_d)$ as defined in Section 4.3.1.3.

If the primary goal is labeling the action in the video, the probability in Equation (4.14) can be marginalized over d to estimate the marginal distribution over the class labels. The final label is selected by choosing the label with the highest probability in this marginal distribution.

4.3.2 Learning

For a set of training videos $\{V_n\}$ and corresponding set of ground truth labels $\{l^n\}$, where $n = 1, \dots, N$, our goal is to maximize the probability of ground truth label for each video by simultaneously optimizing both the localization parameters Φ and the classification parameters Θ . Converting the criterion to a loss function by taking the negative logarithm of the likelihood and adding regularization terms, we get:

$$L(\Phi, \Theta) = - \sum_{n=1}^N \sum_{c=1}^C \mathbb{1}_c(l^n) \log y_c^n(\Phi, \Theta) + \frac{1}{2} \epsilon_1 \sum_{d=1}^D \|\phi_d\|^2 + \frac{1}{2} \epsilon_2 \sum_{d=1}^D \sum_{c=1}^C \|\boldsymbol{\theta}_{dc}\|^2 \quad (4.15)$$

where the probability of video V_n is classified as class c is given by

$$y_c^n(\Phi, \Theta) = \frac{\sum_{d=1}^D \exp(\hat{\mathbf{x}}_d^n(\phi_d)^\top \boldsymbol{\theta}_{dc})}{\sum_{c'=1}^C \sum_{d=1}^D \exp(\hat{\mathbf{x}}_d^n(\phi_d)^\top \boldsymbol{\theta}_{dc'})} \quad (4.16)$$

and $\mathbb{1}_c(l^n)$ is a class indicator function

$$\mathbb{1}_c(l^n) = \begin{cases} 0 & , c \neq l^n \\ 1 & , c = l^n \end{cases}. \quad (4.17)$$

We train the localization parameters, $\Phi = \{\phi_d\}$, and classification parameters, $\Theta = \{\boldsymbol{\theta}_{dc}\}$ using standard conjugate gradient descent optimization package. The gradients are computed efficiently using the backpropagation algorithm.

4.3.3 Experimental Evaluation

In this section, we first show the effectiveness of our proposed model at localizing discriminative sub-regions on UCF Sports dataset which is the most consistently used dataset in recent work on localization [8–10] because ground-truth localizations are available at each frame. Then, we present the action recognition results of our method on two major action recognition datasets: HMDB and UCF101.

We compare our results with the baseline global BoW model and state-of-the-art methods using STIP and MBH features. For the baseline BoW model, we compute the global histograms of visual words and transform the histograms using Kernel Map to approximate the Histogram Intersection

Kernel (HIK). Then, we train multi-class linear SVM using the transformed features. This essentially provides very similar results as training a non-linear SVM using HIK. However, we use Kernel Map in our baseline BoW setting in order to make a direct comparison with our proposed weakly-supervised method.

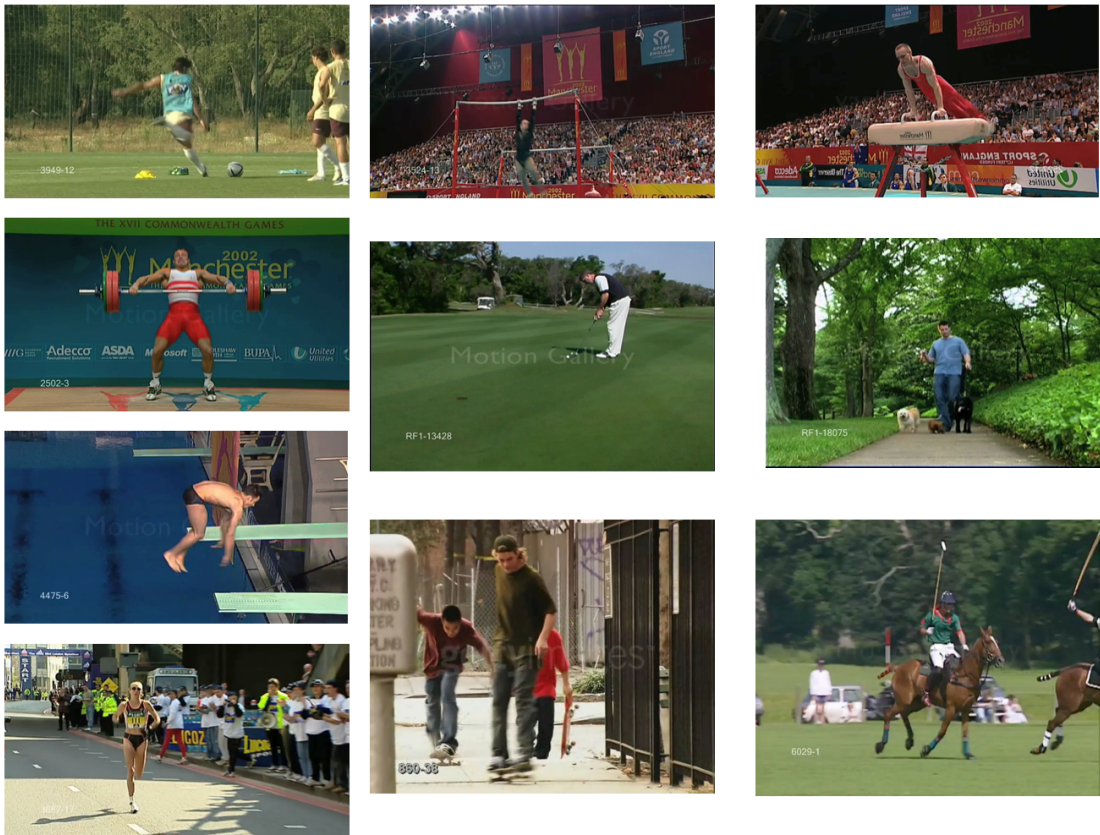


Figure 4.7: Sample video frames from UCF Sports dataset for each category.

4.3.3.1 UCF Sports Dataset

The UCF Sports dataset contains 150 video sequences and includes 10 human actions: diving, golf swinging, kicking (a ball), weight lifting, horse riding, running, skateboarding, swinging (on the pommel horse and on the floor), swinging (at the high bar) and walking. It is a challenging dataset

due to large variations in camera motion, object appearance and pose, object scale, viewpoint, cluttered background and illumination conditions. The sample video frames from UCF Sports dataset are shown in Figure 4.7.

4.3.3.1.1 Accuracy Results

While it is common to use a Leave-One-Out-Cross-Validation (LOOCV) testing methodology when conducting experiments with the UCF Sports dataset, Lan et al. [8] have recently pointed out that many of the videos in this dataset are clips taken from a longer video. This is problematic when conducting LOOCV tests because several training clips will often be drawn from the same video as the testing clip. This causes the classifier to perform best when it effectively memorizes the appearance of the training clips to exploit the strong inherent context correlation among clips drawn from the same video segment. In order to overcome this issue, [8] suggest using approximately a third of the videos from each action class for testing while the remaining videos are reserved for training.

Table 4.1: Mean per-class action recognition accuracies (split) on the UCF Sports dataset. We show that our method outperforms both global and results reported in [8–10]. * Both [8] and [9] use ground truth annotations during training where as our model is weakly supervised and does not require ground truth annotations.

Method	Accuracy(%)
Global BoW+SVM [Linear]	65.95
Our Method [Linear]	72.86
Lan et al. [8]	73.1*
Shapovalova et al. [10]	75.3
Raptis et al. [9]	79.4*
Global BoW+SVM[Kernel Map]	70.21
Our Method [Kernel Map]	80.95

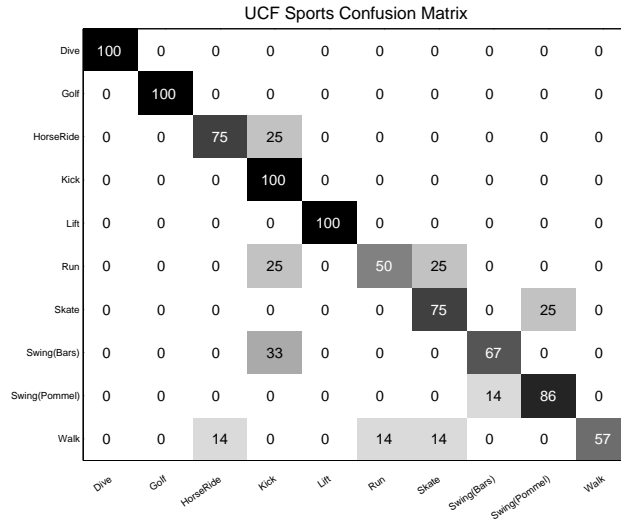


Figure 4.8: Confusion matrix for the UCF Sports dataset using train/test split.

Table 4.1 shows mean per-class accuracy results and Figure 4.8 shows the corresponding confusion matrix using the train-test split suggested in [8]. We observe that using a global bag-of-words approach only leads to an approximate 66% accuracy. This is similar to results reported in [8–10] and demonstrates the inability of the system to differentiate between discriminative vs non-discriminative features.

While a slight improvement is achieved by switching from a linear to non-linear kernel, our localization based-system is able to improve the classification accuracy of the global bag-of-words system by more than 10%. Compared with recent action recognition systems, Table 4.1 also shows two important aspects of the performance of this system:

1. The 80.95% recognition accuracy of our system is more than 5% better than the accuracy of the recently proposed weakly-supervised system in [10], with the added advantage of not requiring any object saliency detector for limiting candidate sub-regions. This is, in part, due to the ability of our system to learn localizers that are trained to find discriminative

sub-regions.

2. Our system is competitive with recently proposed methods that require hand-annotated regions in the training data [8, 9], significantly improving on [8].

Because these results were computed by treating each frame independently, we also experimented with a Conditional Random Field (CRF) model that includes a smoothness terms between frames. Instead of finding the discriminative locations independently, the CRF system optimizes the locations jointly over the frames. However, we saw no improvement in the recognition performance. This could be because the localizers were able to find strong enough visual information even when the frames were treated independently. Also, histograms are aggregated across frames which helps smooth out the effect of jitters in the localization.

We also performed experiments using the LOOCV method on the UCF Sports dataset. The accuracies for the LOOCV method are provided in Table 4.2. We can see that our method is comparable to results reported by others.

Table 4.2: Mean per-class action recognition accuracies (LOOCV) on the UCF Sports dataset. Our LOOCV results are comparable to the state-of-the-art.

Method	Accuracy(%)
Kovashka et al. [31]	87.3
Klaser [45]	87.3
Wang et al. [42]	85.6
Yeffet et al. [59]	79.3
Rodriguez et al. [2]	69.2
Lan et al. [8]	83.7
Our Method	83.7

Lastly, we conducted experiments using different numbers of discriminative sub-region localizers. The corresponding classification accuracies are provided in Table 4.3. When the number of

localizers is zero, the whole frame is considered, and the results are similar to the global bag of words model. Even with only 2 localizers, the classification performance significantly increases, peaking with 10 localizers. Increasing the number of localizers further, results in a slight drop in performance which may be due to the model overfitting on the training data.

Table 4.3: Classification accuracy w.r.t. the number of discriminative sub-region detectors. We observe that increasing localizers improves recognition performance up until $D = 10$. Beyond that, there is a drop in performance, which can be attributed to overfitting.

# of detectors (D)	Accuracy(%)
0	71.78
1	73.45
2	77.85
5	79.52
10	80.95
15	78.45

4.3.3.1.2 Localization Results

Because our method is trained using weakly-supervised data, this method is limited to learning to isolate discriminative sub-regions in the video. However, the results in this section will show that this approach is able to actually locate the action with accuracy that is comparable to previous work that was trained with hand-annotated bounding boxes.

The action location is found using the action/location probabilities. Just as the distribution in Equation (4.14) was marginalized to compute a distribution over action class, it can be marginalized to compute a marginal distribution over which localizer has identified the action. The sub-region can

then be found by picking the localizer with highest probability:

$$d^* = \operatorname{argmax}_{d \in D} \sum_{c=1}^C p(d, c). \quad (4.18)$$

Once we decide on the localizer, we pick the sub-region with the highest probability in each video frame using Equation (4.6).

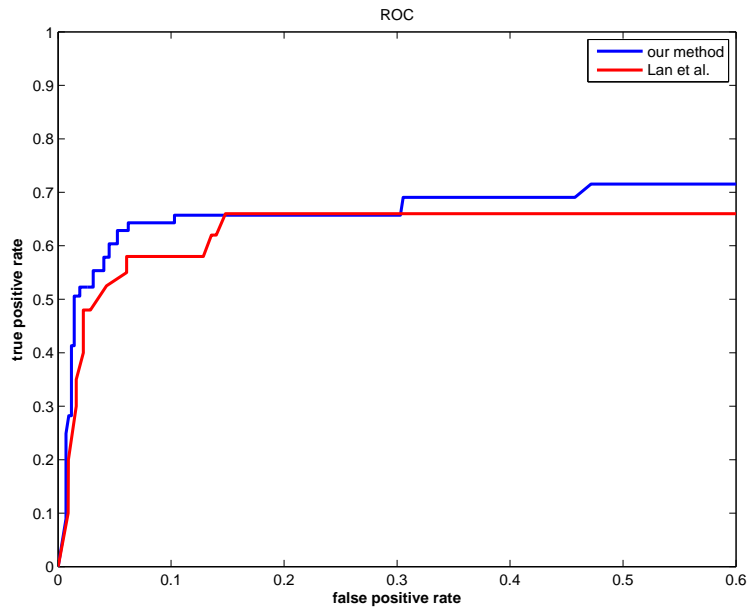
Figure 4.9 shows localization results obtained using our proposed technique that provide empirical evidence that this localizes the actual action well, despite only being trained to locate discriminative sub-regions. This indicates that the sub-regions containing the actual action tend to be the most discriminative sub-regions.

In order to evaluate how well our discriminative sub-regions are localizing actions, we use the same evaluation criterion given in [8]: we first compute the intersection-over-union score per frame using $\text{Area}(R_f \cap R_f^g) / \text{Area}(R_f \cup R_f^g)$ where R_f is the detected sub-region in frame f and R_f^g is the ground-truth action bounding box. Then, we compute the average intersection-over-union score for a video using all frames in the video. If this score is greater than a threshold σ , we consider the video to be correctly localized. We compute ROC curves for each action class similar to [8]. A video is considered as correctly predicted if both the prediction label and the localization match the ground truth.

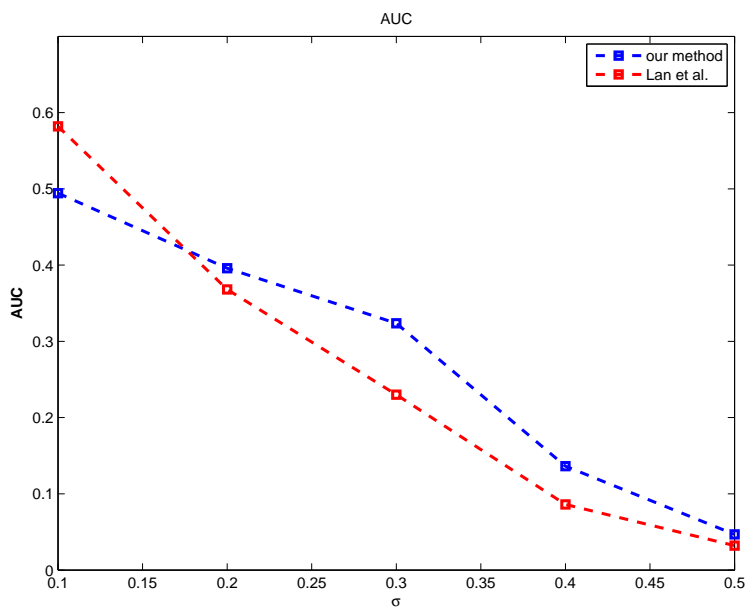
Figure 4.10 shows our average ROC curve for action classes and the ROC curve from [8] for $\sigma = 0.2$. We also compute the area under the ROC curve (AUC) for different σ values. Although our system has no access to ground-truth bounding boxes during training, while the system in [8] does, our system performs comparably with [8] and in many cases outperforms it.



Figure 4.9: We show localization results obtained using our method on the UCF Sports action dataset. We can see that our model is able to correctly localize action specific sub-regions as the best possible representation of the action being conducted in the video.



(a)



(b)

Figure 4.10: Comparison of action localization performance against Lan et al. [8]. (a) ROC curves for $\sigma = 0.2$. (b) Area Under ROC for different σ . σ is the threshold that determines if a video is correctly localized. Compared with [8], which requires the action be manually located in the training data, our system produces comparable or improved results.

4.3.3.2 HMDB Dataset

The UCF Sports dataset is often used for action localization tasks since it incorporates the ground truth annotations of the actions. However, it is a relatively small dataset for action recognition tasks with only 10 action classes and 150 videos. We ran experiments on HMDB dataset [3] to demonstrate the action classification performance of our method on larger action recognition datasets. The HMDB dataset consists of 51 action categories and 6849 video clips (see Figure 4.11 for sample video frames). Each action category contains a minimum of 101 clips. The action categories in HMDB dataset are grouped in five categories as follows:

1. General facial actions: smile, laugh, chew, talk.
2. Facial actions with object interaction: smoke, eat, drink.
3. General body movements: cartwheel, clap hands, climb, climb stairs, dive, fall on the floor, backhand flip, handstand, jump, pull up, push up, run, sit down, sit up, somersault, stand up, turn, walk, wave.
4. Body movements with object interaction: brush hair, catch, draw sword, dribble, golf, hit something, kick ball, pick, pour, push something, ride bike, ride horse, shoot ball, shoot bow, shoot gun, swing baseball bat, sword exercise, throw.
5. Body movements with human interaction: fencing, hug, kick someone, kiss, punch, shake hands, sword fight.

In our experiments, we follow the original approach using the three train-test splits [3] and report the average accuracy. For each class and split, there are 70 videos for training and 30 videos for testing. Note that the dataset includes both the original videos and their stabilized version. In our experiments we used the original videos.

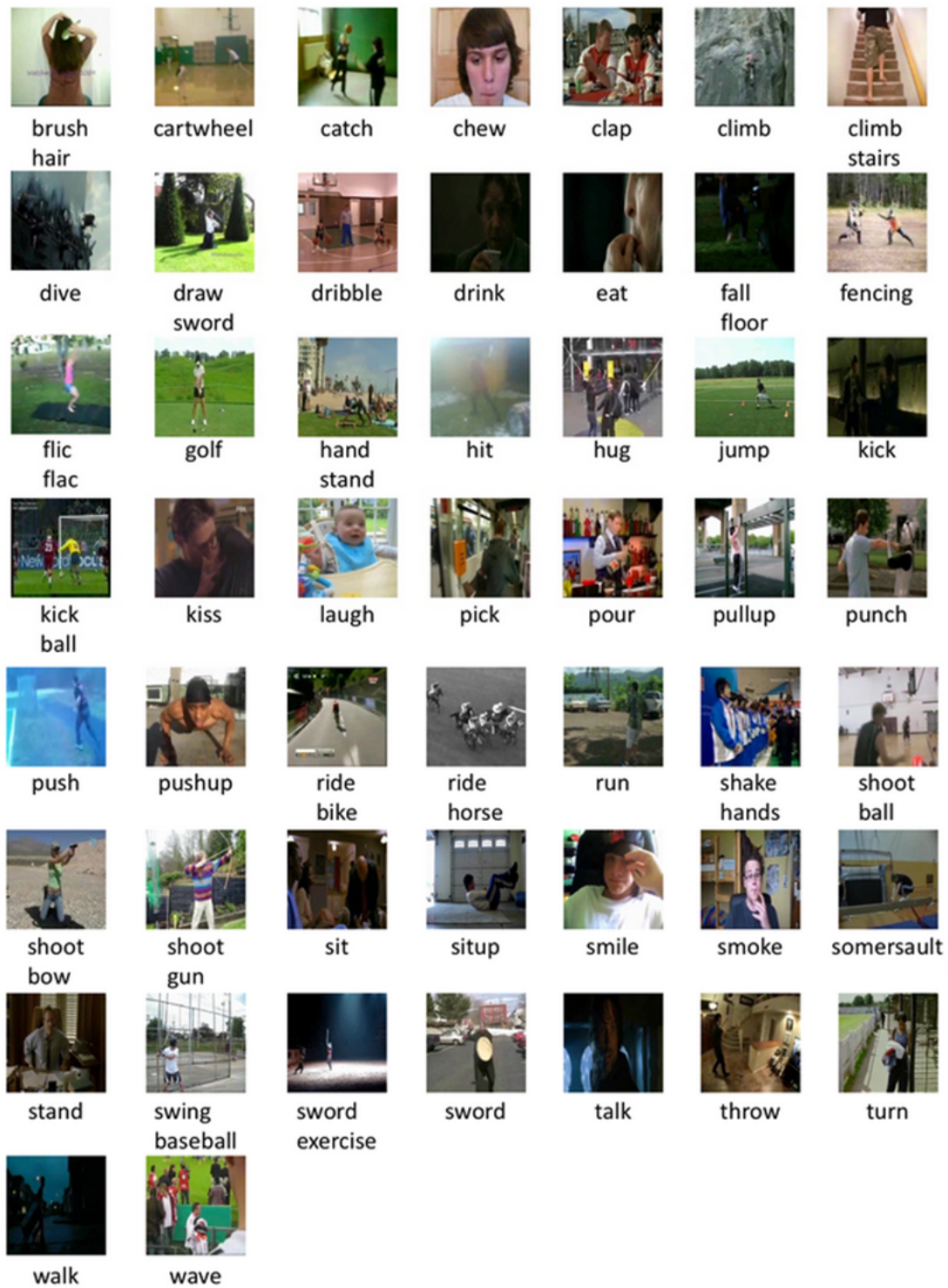


Figure 4.11: Sample video frames from HMDB dataset.

Table 4.4: Mean per-class action classification accuracies for the first 10 action classes from HMDB dataset using STIP features. The number of localizers is set to 10 in a similar setting to the UCF Sports Dataset.

Method	Accuracy(%)
Global BoW	47.1
Our method	57.0

First, we ran an experiment using the first 10 classes from the HMDB dataset. In this experiment, we have used 10 localizers in order to demonstrate performance of our framework on HMDB dataset in a similar setting to the UCF Sports Dataset. The Table 4.4 shows that there is a 10% improvement (21% relative increase) over the baseline global BoW model with histogram intersection kernel, which is consistent with the results on UCF Sports dataset.

Next, we ran experiments on the complete dataset using different number of localizers. Table 4.5 shows that the best performance was obtained using 1 localizer and the performance drops as the number of localizers increases. Overfitting is likely becoming the problem as the number of localizers increases. The HMDB dataset contains 51 action categories and the second layer of our model has over 14 million parameters for HMDB dataset with 10 localizers and gets larger as the number of localizers are increasing.

Table 4.5: Mean per-class action classification accuracies on HMDB dataset using STIP features for different number of localizers.

# of Localizers	Accuracy(%)
1	27.12
5	26.21
10	25.50
15	25.43
20	24.51

In order to demonstrate that our video representation explained in Section 4.1 can be used with any local feature descriptor, we ran experiments using densely sampled Motion Boundary Histogram (MBH) descriptor [44, 60] which outperforms the STIP descriptor in most of the realistic datasets [61]. MBH was proposed by Dalal et al. [60] for detecting humans in films and videos where there is camera motion and background clutter and used by Wang et al. [44] for action recognition problem. MBH is computed using the derivatives of the optical flow in vertical and horizontal directions. Since it is the gradient of the optical flow, the constant camera motion is removed and therefore it is more robust to camera motion than the optical flow. We follow [44] and compute the MBHx and MBHy descriptors along the densely sampled SIFT trajectories. In [44], authors construct different codebooks for MBHx and MBHy. For classification, they use a non-linear SVM with RBF- χ^2 in a one-vs-rest setting. Different descriptors are combined using a multi-channel approach. In our experiments, we combined the MBHx and MBHy into a single feature descriptor and constructed a codebook with 4000 visual words by clustering a subset of 100,000 randomly selected training features using k-means.

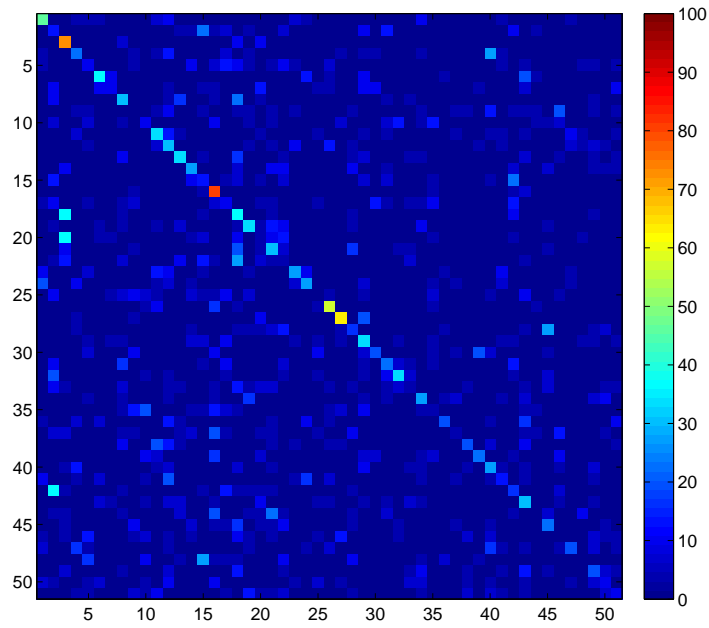
Table 4.6: Comparison of our method with global BoW and other methods that use STIP (HOG/HOF) and MBH features.

Method	Accuracy(%)
HOG/HOF [3]	20.0
C2 [3]	23.0
Action Bank [12]	26.9
Dense Trajectory [MBH] [44]	43.2
Global BoW [STIP]	21.0
Global BoW [MBH]	36.6
Our method [STIP]	27.12
Our method [MBH]	42.42

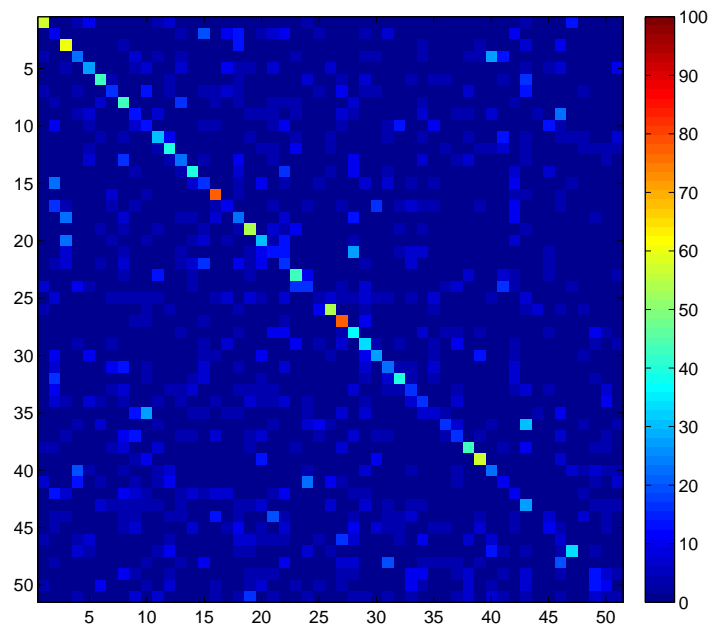
Table 4.6 provides a comparison of our method with the global BoW method and other methods that are using STIP (HOG/HOF) and MBH features. As shown in the table, the classification

accuracy of our method is 6.12% better than the global BoW model for STIP features and 5.82% better for MBH features. Our results are better than or comparable to the state-of-the-art methods that use STIP features. However, we were not able to generate the results reported in [44] for MBH features using global BoW model.

Figures 4.12 and 4.14 show the confusion matrices for global BoW model and our method using both MBH and STIP features. Finally, the per-class action classification accuracies are provided in Figures 4.13 and 4.15, where each bar shows the percentage of the videos that were correctly classified in each action category. As shown in figures, our proposed method provides better classification accuracies for most of the action classes when compared to the global BoW model.



(a)



(b)

Figure 4.12: Confusion matrix for HMDB dataset using STIP features. (a) global BoW (b) our method

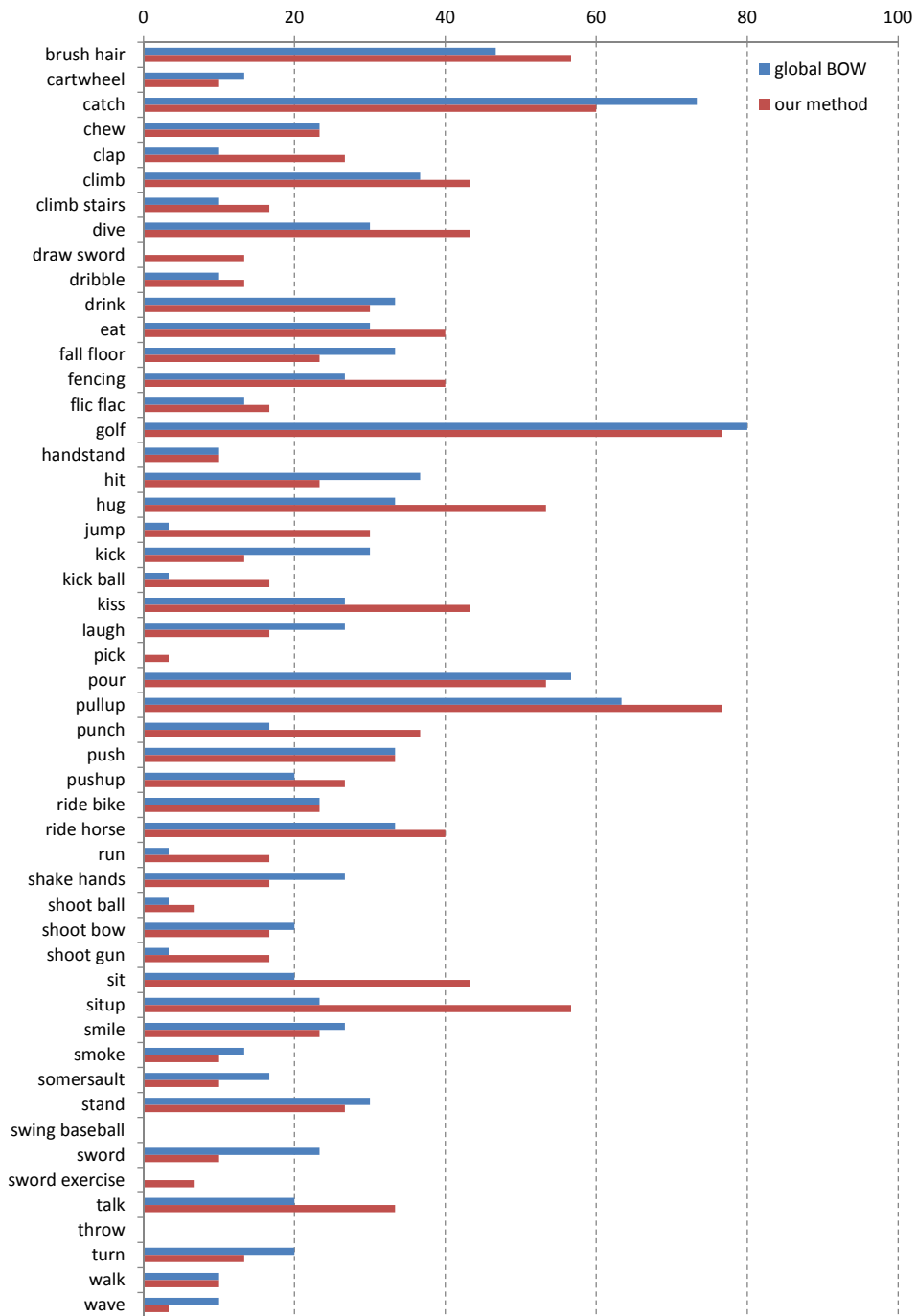
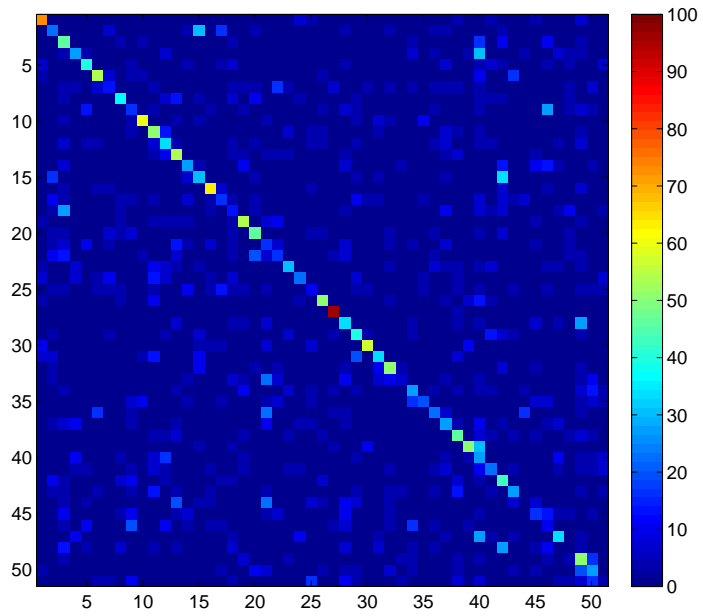
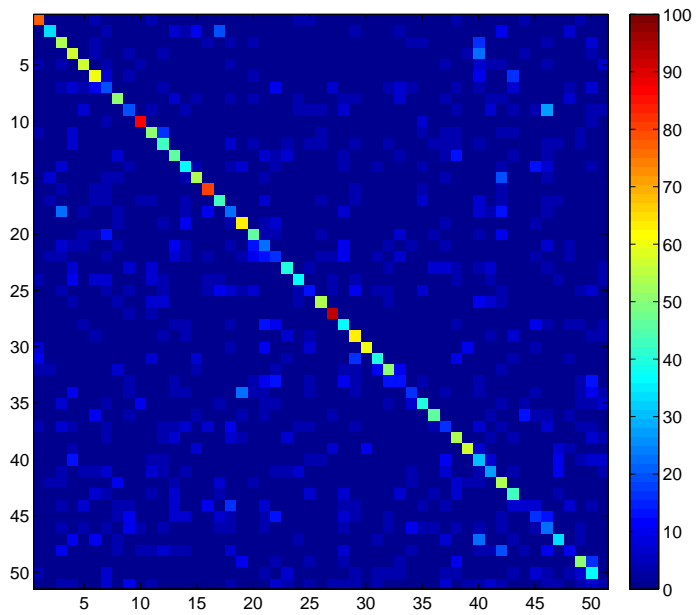


Figure 4.13: Comparison of action class accuracies for HMDB dataset using STIP features. The bars show the percentage of the videos that are correctly classified in each action class.



(a)



(b)

Figure 4.14: Confusion matrix for HMDB dataset using MBH features. (a) global BoW (b) our method

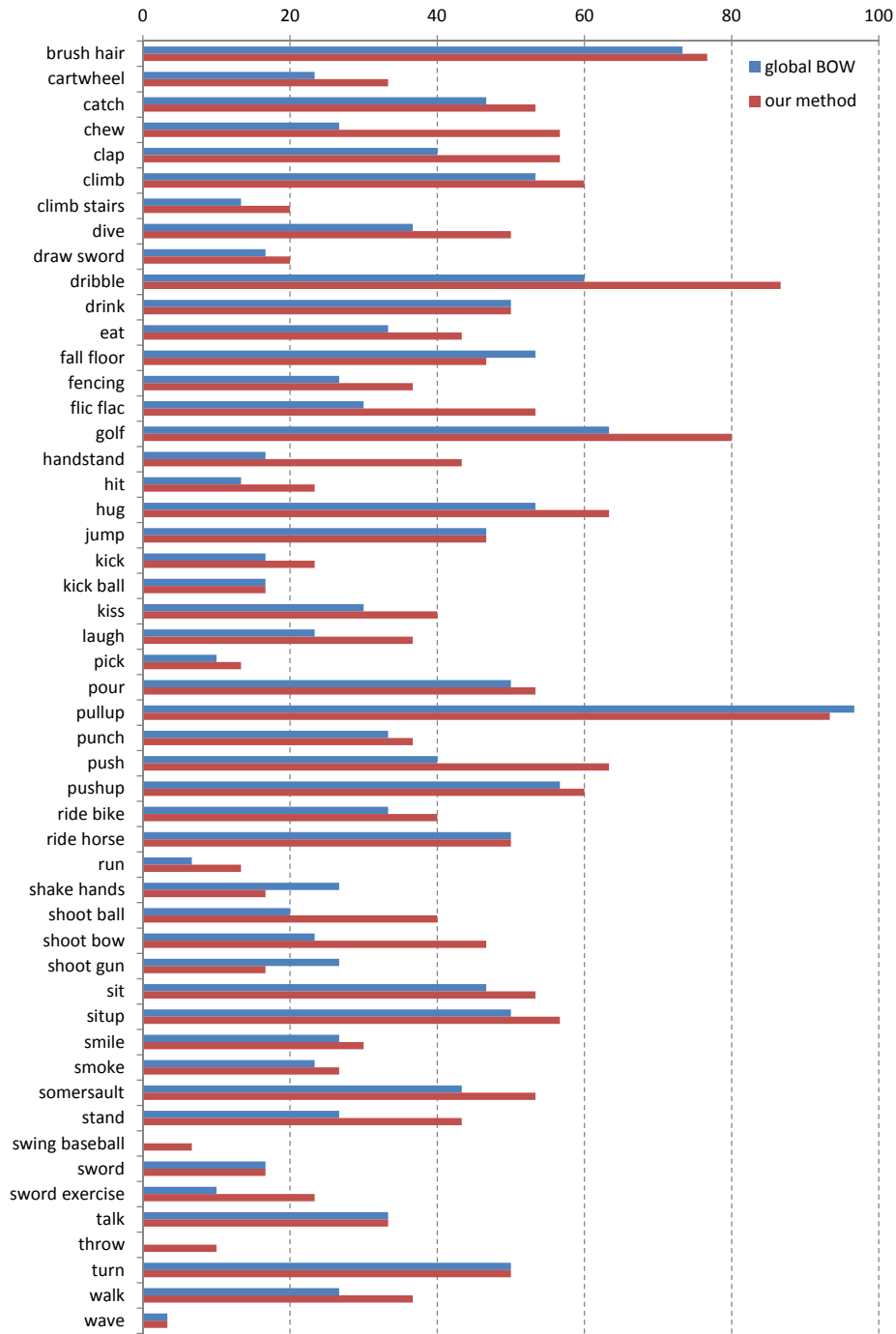


Figure 4.15: Comparison of action class accuracies for HMDB dataset using MBH features. The bars show the percentage of the videos that are correctly classified in each action class.

4.3.3.3 UCF101 Dataset

UCF101 [4] is an action recognition dataset collected from YouTube. It is an extension of UCF50 dataset and consists of 13320 videos from 101 action categories. The videos in each action category are grouped into 25 groups, where each group consists of 4-7 videos of an action. The videos from the same group may share some common features, such as similar background, similar viewpoint, etc.

The action categories in UCF101 dataset are grouped into five types as follows: 1) Human-Object Interaction 2) Body Motion Only 3) Human-Human interaction 4) Playing Musical Instruments 5) Sports. The action categories for UCF101 data set are: Apply Eye Makeup, Apply Lipstick, Archery, Baby Crawling, Balance Beam, Band Marching, Baseball Pitch, Basketball Shooting, Basketball Dunk, Bench Press, Biking, Billiards Shot, Blow Dry Hair, Blowing Candles, Body Weight Squats, Bowling, Boxing Punching Bag, Boxing Speed Bag, Breaststroke, Brushing Teeth, Clean and Jerk, Cliff Diving, Cricket Bowling, Cricket Shot, Cutting In Kitchen, Diving, Drumming, Fencing, Field Hockey Penalty, Floor Gymnastics, Frisbee Catch, Front Crawl, Golf Swing, Haircut, Hammer Throw, Hammering, Handstand Pushups, Handstand Walking, Head Massage, High Jump, Horse Race, Horse Riding, Hula Hoop, Ice Dancing, Javelin Throw, Juggling Balls, Jump Rope, Jumping Jack, Kayaking, Knitting, Long Jump, Lunges, Military Parade, Mixing Batter, Mopping Floor, Nun chucks, Parallel Bars, Pizza Tossing, Playing Guitar, Playing Piano, Playing Tabla, Playing Violin, Playing Cello, Playing Daf, Playing Dhol, Playing Flute, Playing Sitar, Pole Vault, Pommel Horse, Pull Ups, Punch, Push Ups, Rafting, Rock Climbing Indoor, Rope Climbing, Rowing, Salsa Spins, Shaving Beard, Shotput, Skate Boarding, Skiing, Skijet, Sky Diving, Soccer Juggling, Soccer Penalty, Still Rings, Sumo Wrestling, Surfing, Swing, Table Tennis Shot, Tai Chi, Tennis Swing, Throw Discus, Trampoline Jumping, Typing, Uneven Bars, Volleyball Spiking, Walking with a dog, Wall Pushups, Writing On Board, Yo Yo.

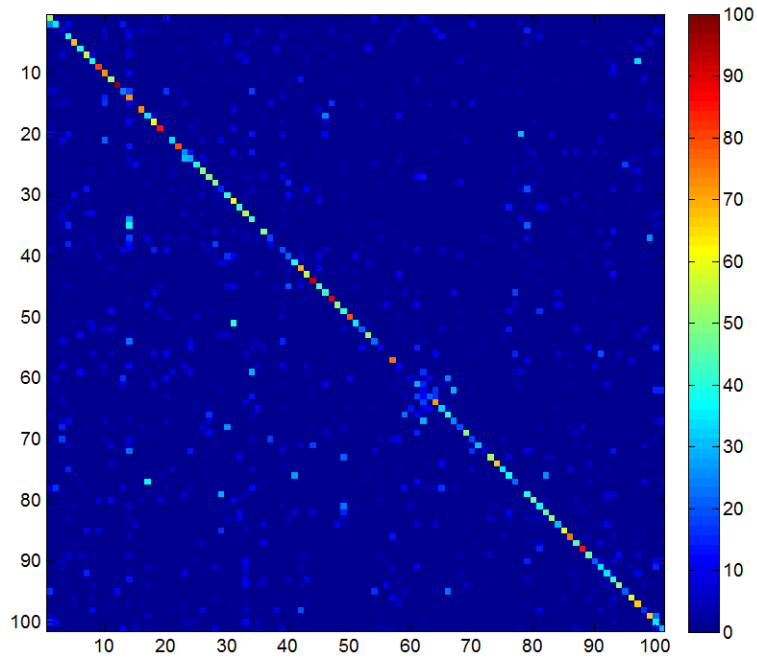
In our experiments, we follow the original approach using the three splits for training-testing [4] and report the average accuracy. Table 4.7 compares the accuracy results of our method using STIP features with the baseline BoW and [4] which is the only reported result on UCF101. Table 4.8 compares the accuracy results of our method using MBH features with the baseline BoW. Our method is 7.69% better than the state-of-the-art [4] for STIP features and 6.56% better than the global BoW model for MBH features.

Table 4.7: Comparison of our method with global BoW using STIP features.

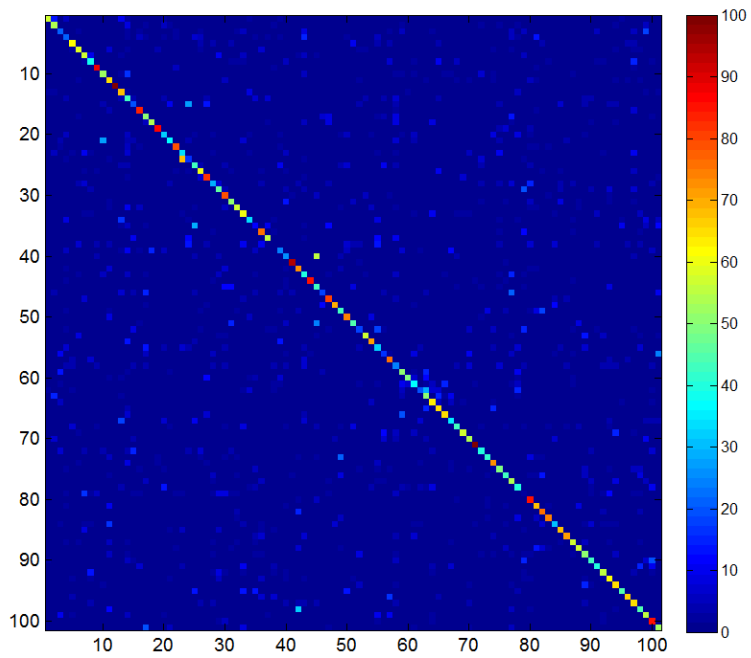
Method	Accuracy(%)
Global BoW + Linear SVM	34.99
Global BoW + SVM [Kernel Map]	43.94
Global BoW [Harris3D + STIP] [4]	43.90
Our method with 1 localizer	51.59
Our method with 5 localizers	51.21

Table 4.8: Comparison of our method with global BoW using MBH features.

Method	Accuracy(%)
Global BoW + SVM [Kernel Map]	65.28
Our method with 1 localizer	71.84
Our method with 5 localizers	70.35



(a)



(b)

Figure 4.16: Confusion matrix for UCF101 dataset using STIP features. (a) global BoW (b) our method

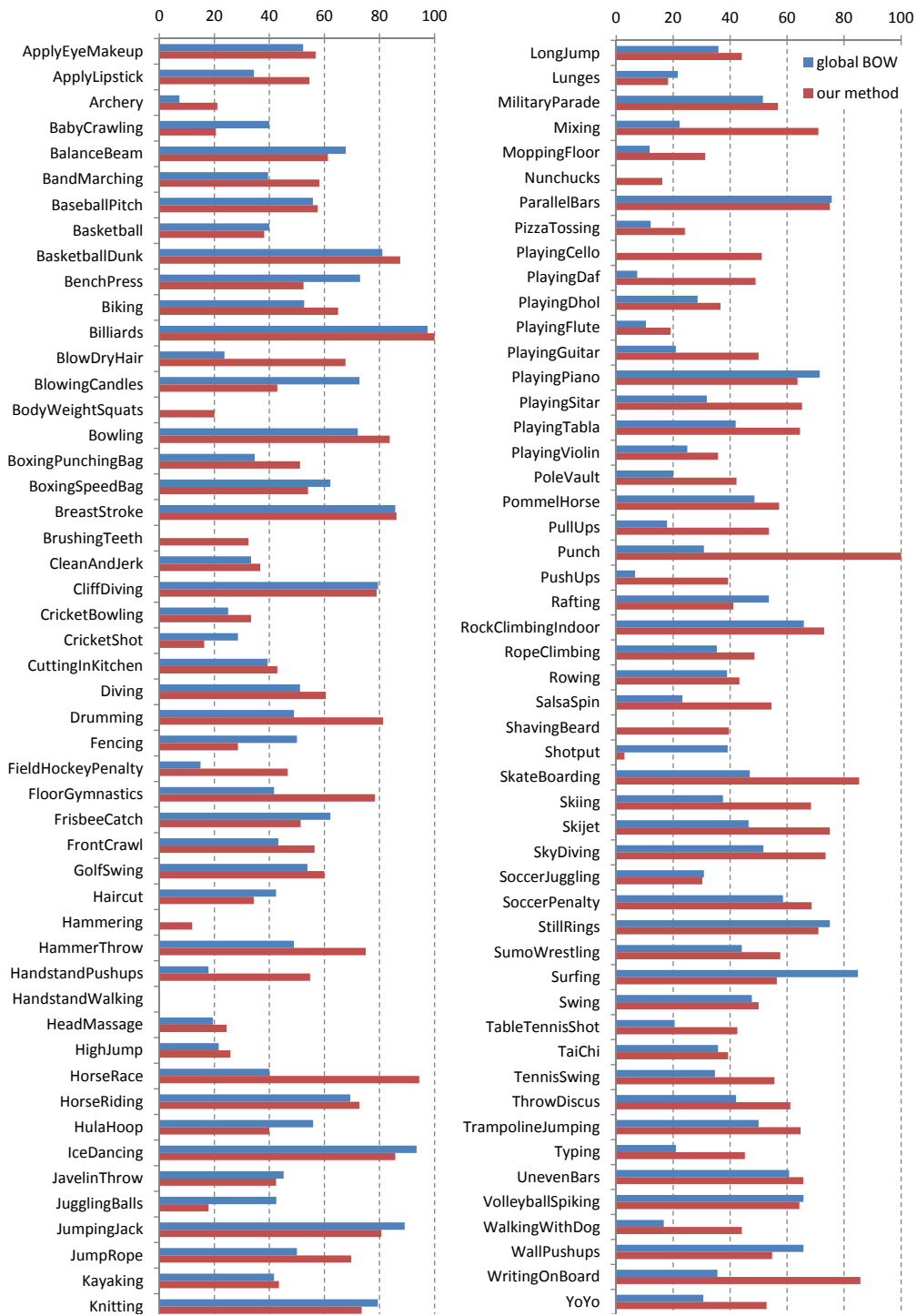
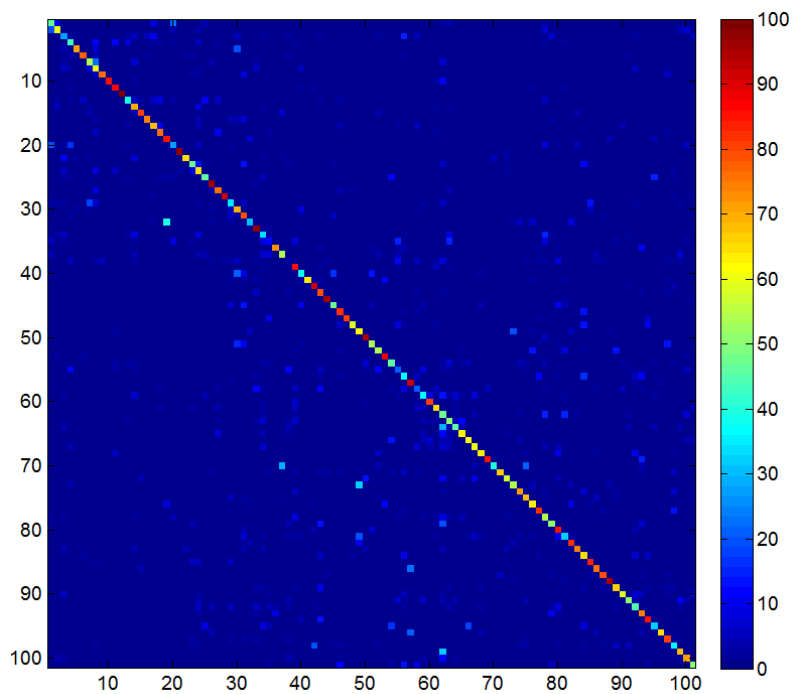
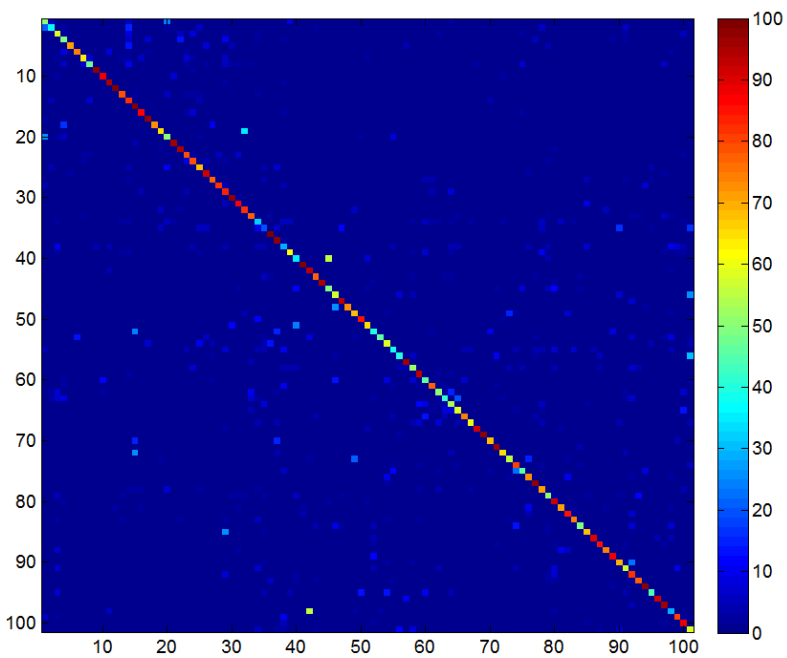


Figure 4.17: Comparison of action class accuracies for UCF101 dataset using STIP features. The bars show the percentage of the videos that are correctly classified in each action class.



(a)



(b)

Figure 4.18: Confusion matrix for UCF101 dataset using MBH features. (a) global BoW (b) our method

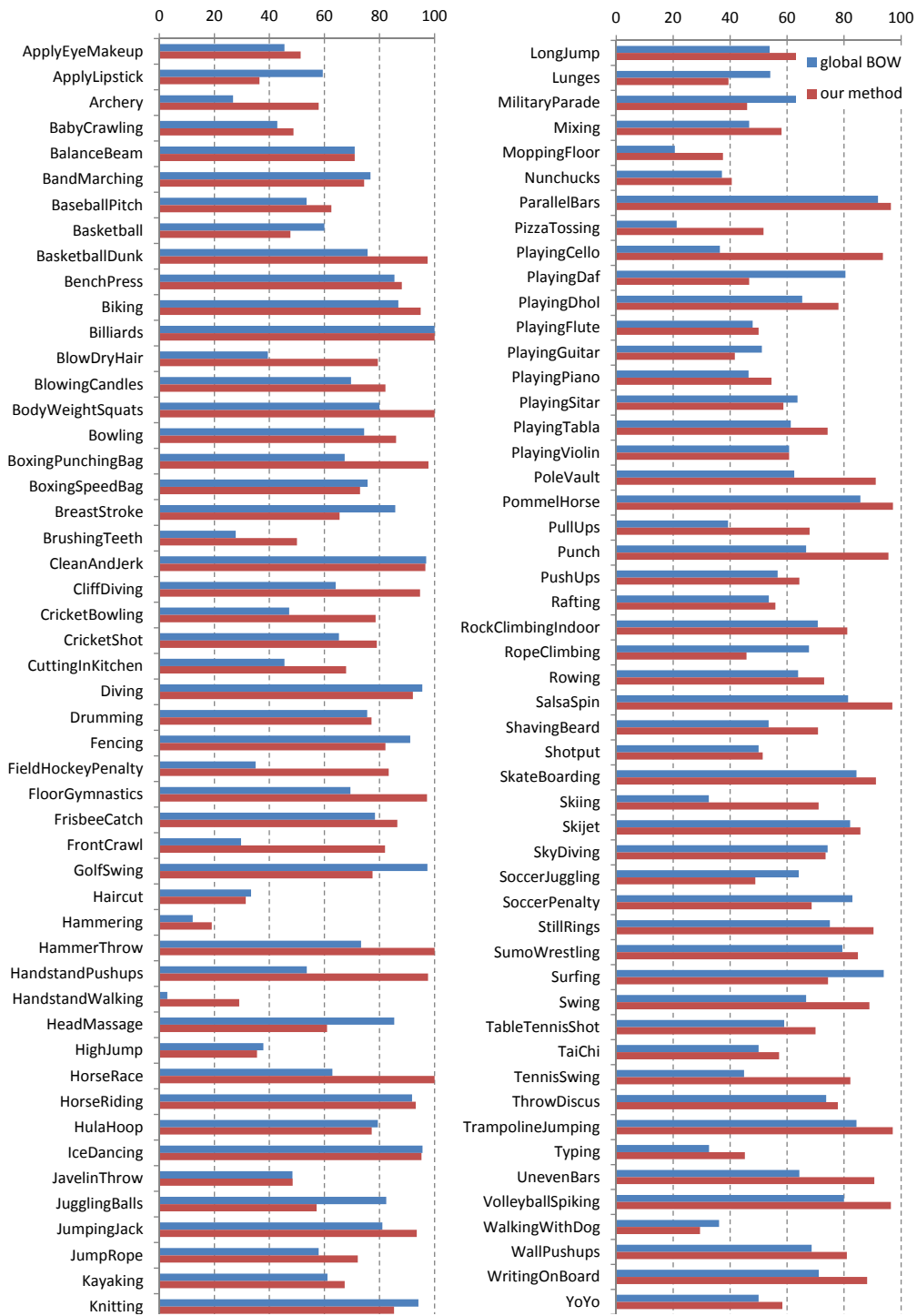


Figure 4.19: Comparison of action class accuracies for UCF101 dataset using MBH features. The bars show the percentage of the videos that are correctly classified in each action class.

CHAPTER 5: ACTION RECOGNITION USING NEURAL NETWORKS WITH DROPOUT

One common problem with datasets containing large number of classes is the overfitting of the parameters on the training data. As an example, the second layer of our framework has over 14 million parameters for HMDB dataset with 10 localizers and gets larger as the number of localizers increases. Most commonly used approach to avoid overfitting is introducing a regularization term to the cost function and controlling the overfitting by preventing the weight values getting very large.

A recently introduced technique for feed-forward deep neural networks, called “dropout” [17], has been applied successfully to object recognition problem by preventing the overfitting substantially [18]. The dropout procedure can be viewed as a very efficient way of performing model averaging with neural networks. The overfitting on the data is reduced by setting the output of hidden units to zero with a probability of 0.5 and those units do not contribute to the forward pass and back-propagation. Every time a training input is presented, the architecture of the network changes, but still sharing the same weights and prevents complex co-adaptations on the training data.

Building upon success of dropout in [17, 18], we integrate the dropout technique into our action recognition framework introduced in Chapter 4 by replacing the multinomial logistic regression classifier with a multilayer neural network as shown Figure 5.1. The output of the kernel map transformation are used as input to the neural network. The last layer of the network is a C-way softmax classifier for final action classification. Using a deep neural network instead of a single layer logistic regression model enables the system to learn more complex representations from features and therefore results in a more discriminative classifier. Also, using the dropout

technique for training the neural network reduces overfitting of the parameters on the training data and improves the accuracy results. Even though the success of dropout was demonstrated on large scale object recognition datasets, to our knowledge it has not been used for action recognition problem.

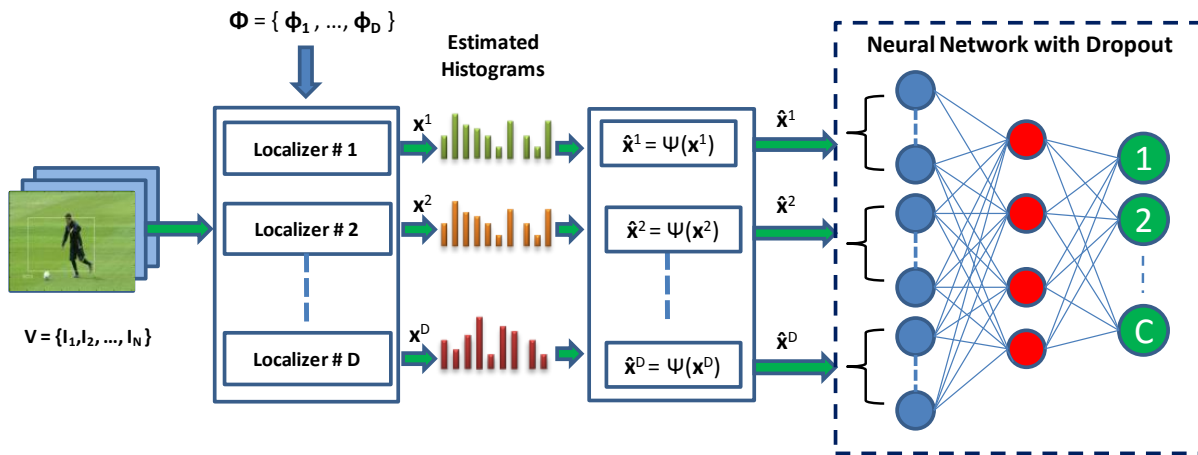


Figure 5.1: Action recognition framework using neural network with dropout. The first stage of the framework is kept same as the one in Figure 4.5 and the second stage is replaced with a two-layer neural network.

The rest of this chapter is organized as follows. Section 5.1 provides an overview of the video representation. Section 5.2 discusses the proposed action recognition framework with neural network and dropout. Section 5.3 explains how the neural network parameters are trained. Finally, Section 5.4 provides the experimental results on HMDB and UCF101 datasets.

5.1 Video Representation

A video sequence is represented as a collection of descriptor frames as explained in Section 4.1. We use two different feature descriptors in our experiments: STIP and MBH.

The STIP features are computed using Laptev’s STIP detector [37] with default settings. The interest points are sampled at eight spatial and two temporal scales. 72 dimensional HOG and 90 dimensional HOF feature descriptors are extracted at each interest point and concatenated into a 162 dimensional STIP feature vector.

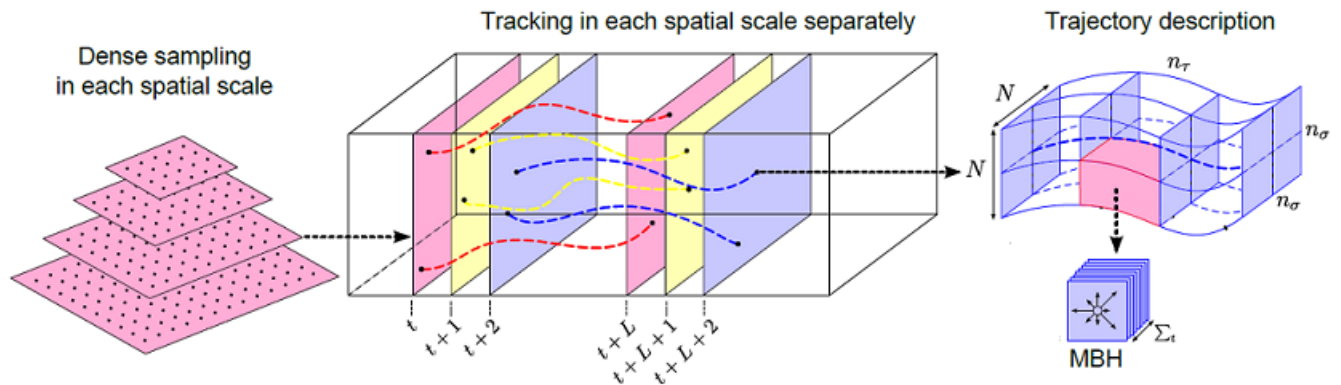


Figure 5.2: Feature points are sampled densely at each spatial scale and tracked for $L = 15$ frames. MBH features are computed within a space-time volume aligned with the trajectories. (Figure is taken from [44])

The MBHx and MBHy descriptors, which represent the gradient of the horizontal and vertical components of optical flow respectively, are computed along the densely sampled SIFT trajectories as explained in [44]. Dense trajectories are extracted by first sampling feature points on a grid spaced by 5 pixels at different spatial scales and then tracking feature points on each spatial scale separately through the video. There are 8 spatial scales at most depending on size of the video and the spatial scale increases by a factor of $1/\sqrt{2}$. MBHx and MBHy descriptors are computed within the space-time volumes aligned with the trajectories as shown in Figure 5.2. The size of the volume is $N \times N$ pixels and L frames where $N = 32$ and $L = 15$. We concatenate these two components into a single MBH feature.

We construct separate codebooks for STIP and MBH descriptors by clustering a subset of 100,000 randomly selected training features using k-means. We set the number of visual words in each codebook to 4000. Finally, each feature point p_j is represented as the tuple (x_j, y_j, t_j, c_j) , denoting that it was observed at (x_j, y_j) in frame t_j of the video; the label c_j corresponds to the index of the visual word in the codebook that is closest in feature space to p_j 's descriptor.

5.2 Framework Overview

Figure 5.1 shows the proposed action recognition framework with neural network. The first stage of the framework is same as the localization stage of the framework introduced in Section 4.3.1. The second stage of the original framework is replaced with a two-layer feed-forward neural network.

In the first stage of the framework, the video histograms are computed using the localizer weights as explained in Section 4.3.1.2. We use the same localizer weights learned in Section 4.3.3 for computing the histograms. Once the histograms are computed, they are transformed to a higher dimension using kernel map to approximate the histogram intersection kernel (HIK). The output of the kernel map are used as the input layer of the neural network.

For simplicity, we will first consider the case when there is a single localizer and then, extend it to multiple localizers. Figure 5.3 shows the network diagram for a single localizer. The input, hidden and output variables are represented by nodes and the weight parameters are represented by the links between nodes. The \hat{x}_i represents the i^{th} bin of the feature vector \hat{x} and the \hat{x}_0 and z_0 represent the bias terms which are set to 1. The input for the j^{th} hidden unit is computed as the linear combination of the input variables as follows:

$$a_j = \sum_{i=1}^B w_{ji}^{(1)} x_i + w_{j0}^{(1)} \quad (5.1)$$

where $j = 1, \dots, M$, B is total number of inputs and the superscript (1) indicates the first layer of the network. Output of the hidden units are computed using a nonlinear activation function h :

$$z_j = h(a_j) \tag{5.2}$$

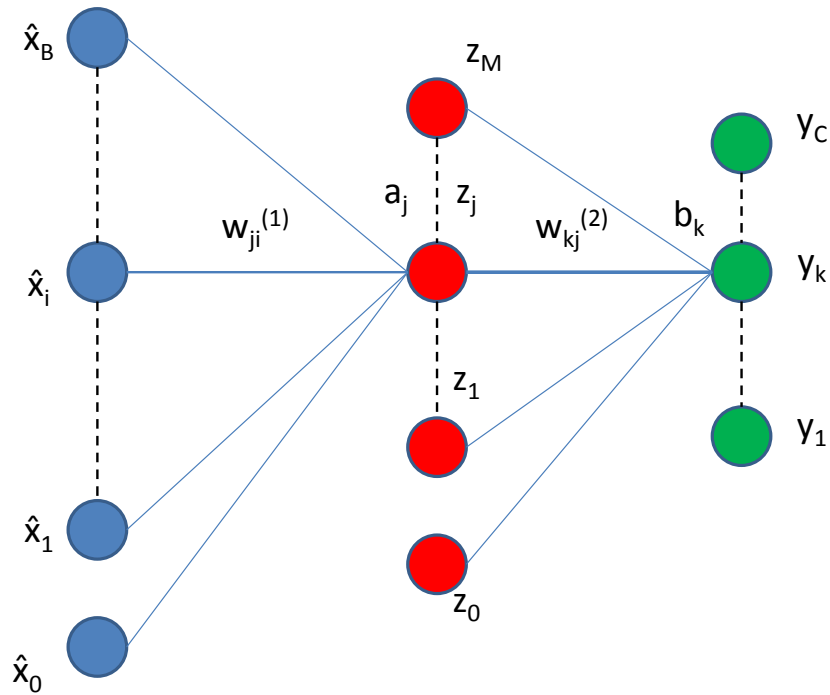


Figure 5.3: Network diagram for a two-layer neural network with B inputs, M hidden units and C outputs. The output of the kernel map are used as input to the neural network.

We have used two different activation functions in our experiments , i.e.

$$h(a) = \begin{cases} 1/(1 + e^{-a}) & , \text{logistic sigmoid} \\ (1 - e^{-2a})/(1 + e^{-2a}) & , \text{hyperbolic tangent (tanh)} \end{cases} \quad (5.3)$$

and the accuracy results using the logistic sigmoid was slightly better than the hyperbolic tangent.

The outputs of the hidden units are linearly combined as input to the output units

$$b_k = \sum_{j=1}^M w_{kj}^{(2)} z_j + w_{k0}^{(2)} \quad (5.4)$$

where $k = 1, \dots, C$ and C is the total number of action classes. Finally, the softmax activation function is used to provide the class probability outputs as follows:

$$y_k = f(b_k) = \frac{e^{b_k}}{\sum_{k'} e^{b_{k'}}} \quad (5.5)$$

The output with the highest probability is selected as the predicted class label. The following equation summarizes the output as a function of network inputs and weight parameters

$$y_k(\hat{\mathbf{x}}, \mathbf{w}) = f \left(\sum_{j=1}^M w_{kj}^{(2)} h \left(\sum_{i=1}^B w_{ji}^{(1)} \hat{x}_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right). \quad (5.6)$$

The network diagram shown in Figure 5.3 can be extended to the case where there are multiple localizers. We have experimented using two different network structures as shown in Figure 5.4. Even though the accuracy results of both networks were similar, the network shown in Figure 5.4(a) was easier to implement using parallel processing.

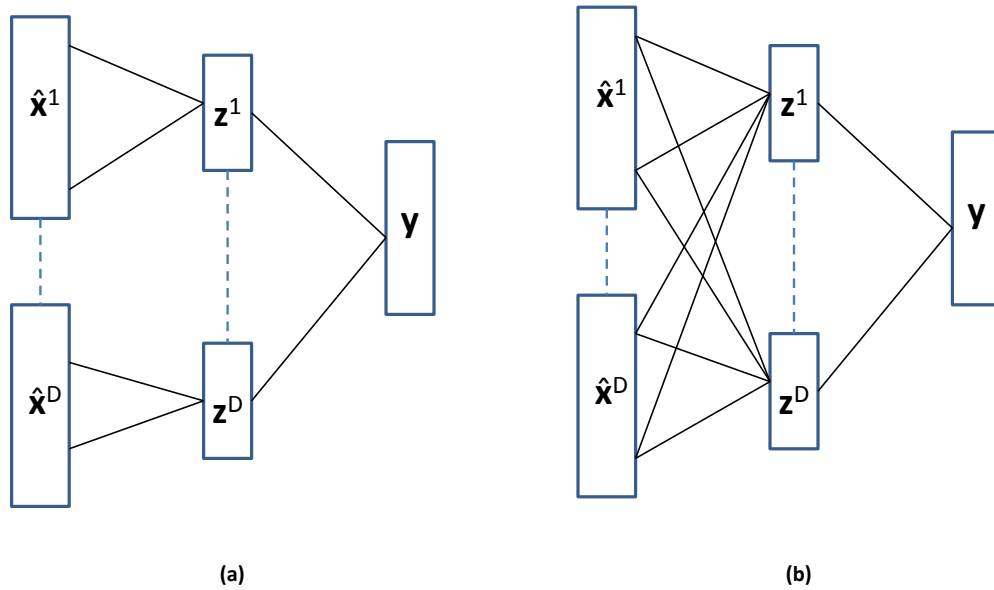


Figure 5.4: Two different network structures are used for multiple localizers: (a) the localizer parameters are only connected to a subset of the hidden units (b) fully connected neural network where all localizer parameters are connected to all hidden units.

5.3 Network Training

We have already trained the parameters of the first stage of our framework in Section 4.3.3. We use the output of the first stage as input of the neural network and only train the neural network parameters of the proposed framework. For a given set of training vectors $\{\hat{x}_n\}$ and corresponding set of ground truth labels $\{l^n\}$, where \hat{x} is the output of the kernel map and $n = 1, \dots, N$, we

define the error function as negative logarithm of the likelihood function on the training set, i.e.

$$E(\mathbf{w}) = - \sum_{n=1}^N \sum_{k=1}^C t_{kn} \log y_k(\hat{\mathbf{x}}_n, \mathbf{w}) + \frac{1}{2} \epsilon \|\mathbf{w}\|^2 \quad (5.7)$$

where

$$y_k(\hat{\mathbf{x}}_n, \mathbf{w}) = \frac{\exp(b_k(\hat{\mathbf{x}}_n, \mathbf{w}))}{\sum_{j=1}^C \exp(b_j(\hat{\mathbf{x}}_n, \mathbf{w}))} \quad (5.8)$$

and

$$t_{kn} = \begin{cases} 0 & , k \neq l^n \\ 1 & , k = l^n \end{cases} \quad (5.9)$$

We trained the neural network weights, \mathbf{w} , using stochastic gradient descent with a batch size of 100 examples and momentum (μ) of 0.5. We set the number of hidden units, M , to 500 for each localizer. The following is the pseudo-code for our stochastic gradient descent algorithm:

1. Initialize the weight parameters, \mathbf{w} , from a zero mean Gaussian distribution with standard deviation of 0.01.
2. Initialize the learning rate $\eta = 0.01$.
3. Repeat until error function is zero or maximum number of iterations is reached:
 - 3.1. Randomly shuffle the examples in the training set.
 - 3.2. For each batch $i = 1, 2, \dots, n$, do:
 - i. If enabled, apply **dropout**.
 - ii. Compute the average error gradient $\nabla E_i(\mathbf{w}_i)$ over the i th batch.
 - iii. Update the weights:

$$\Delta \mathbf{w}_{i+1} = \mu \Delta \mathbf{w}_i - \eta \nabla E_i(\mathbf{w}_i)$$

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \Delta \mathbf{w}_{i+1}$$

The gradients were computed using error backpropagation. The learning rate was initialized to 0.01 and reduced by a factor of 10 when the error rate stopped improving with the current learning rate.

5.3.1 Dropout

The dropout [17, 18] is a recently introduced technique to reduce the overfitting in neural networks and it consists of setting the output of each hidden unit to zero with probability 0.5. The hidden units that are dropped-out do not contribute to the forward pass and not included in backpropagation as shown in Figure 5.5. Every time a training input is presented, the architecture of the network changes and this prevents complex co-adaptations on the training data. During testing, all the neurons are used but their outputs are multiplied by 0.5.

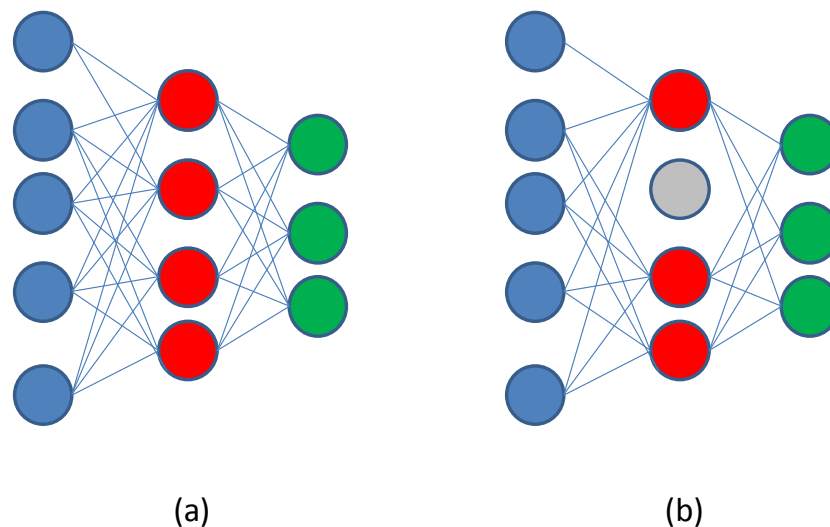


Figure 5.5: (a) In a fully connected neural network, all the hidden units contribute to the forward pass and backpropagation. (b) When there is dropout, the output of each hidden unit is set to zero with probability 0.5 and the hidden units that are dropped out do not contribute to the forward pass and backpropagation.

5.3.2 Computing the Error Gradients

Error gradients can be computed efficiently using the error backpropagation. We can re-write the error function given in Equation 5.7 as

$$E(\mathbf{w}) = \sum_{n=1}^N E_n(\mathbf{w}) + \frac{1}{2}\epsilon\|\mathbf{w}\|^2 \quad (5.10)$$

where

$$E_n(\mathbf{w}) = - \sum_{k=1}^C t_{kn} \log y_k(\hat{\mathbf{x}}_n, \mathbf{w}). \quad (5.11)$$

Hence, the gradient of the error function can be written as a linear combination of the gradient contributions from each training sample as follow

$$\nabla E(\mathbf{w}) = \sum_{n=1}^N \nabla E_n(\mathbf{w}) + \mathbf{w} \quad (5.12)$$

Using 5.5 in 5.11, we obtain

$$E_n(\mathbf{w}) = \sum_{k=1}^C \left(-t_{kn} b_k + t_{kn} \log \left(\sum_{k'=1}^C \exp(b_{k'}) \right) \right) \quad (5.13)$$

First, we compute the derivative of E_n with respect to the second-layer weights as follows:

$$\frac{\partial E_n}{\partial w_{kj}^{(2)}} = \frac{\partial E_n}{\partial b_k} \frac{\partial b_k}{\partial w_{kj}^{(2)}}. \quad (5.14)$$

We define the following notation for simplicity:

$$\delta_k \equiv \frac{\partial E_n}{\partial b_k} \quad (5.15)$$

which is computed using Equation 5.13 as

$$\delta_k = y_k - t_k. \quad (5.16)$$

Using Equation 5.4, we can write

$$\frac{\partial b_k}{\partial w_{kj}^{(2)}} = z_j. \quad (5.17)$$

Substituting Equations 5.16 and 5.17 into Equation 5.14, we obtain

$$\frac{\partial E_n}{\partial w_{kj}^{(2)}} = \delta_k z_j. \quad (5.18)$$

Next, we compute the derivative of E_n with respect to the first-layer weights as follows:

$$\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}^{(1)}} = \delta_j z_i. \quad (5.19)$$

where

$$\frac{\partial a_j}{\partial w_{ji}^{(1)}} = z_i \quad (5.20)$$

and

$$\delta_j \equiv \frac{\partial E_n}{\partial a_j} = \sum_{k=1}^C \frac{\partial E_n}{\partial b_k} \frac{\partial b_k}{\partial a_j}. \quad (5.21)$$

Using Equations 5.4, 5.2 and 5.15 in Equation 5.21, we obtain

$$\delta_j = h'(a_j) \sum_{k=1}^C w_{kj}^{(2)} \delta_k. \quad (5.22)$$

The pseudo-code for the backpropagation procedure is given as follows.

1. For a given input vector $\hat{\mathbf{x}}$, compute the network outputs using Equations 5.1 through 5.5.
2. Compute δ_k for all output units using Equation 5.16.
3. Backpropagate the δ_k s using Equation 5.22 to obtain δ_j for each hidden unit.
4. Use Equations 5.18 and 5.19 for computing the derivatives with respect to the first-layer and second-layer weights.

5.4 Experimental Evaluation

We evaluated the performance of the proposed action recognition framework on HMDB and UCF101 datasets.

The Table 5.1 provides the mean per-class action accuracies on HMDB dataset for different number of localizers using STIP features. We compare the accuracy results of our proposed framework introduced in Section 5.2 with the accuracies reported in Table 4.5 which were obtained using the multinomial logistic regression (MLR). As shown in the table, using neural network (NN) without dropout slightly improves the results over the MLR. However, using NN with dropout improves the results over MLR more than 2% which corresponds to approximately 9% relative increase.

We also ran experiments using MBH features on HMDB dataset. The Table 5.2 compares our results with other methods that use STIP and MBH features. As shown in the table, our method with dropout outperforms the state-of-the-art methods that use STIP and MBH features.

Table 5.1: Comparison of the action classification accuracies for STIP features on HMDB dataset with and without using the dropout.

# of Localizers	Multinomial Logistic Regression	Neural Network without Dropout	Neural Network with Dropout
1	27.12%	27.38%	29.56%
5	26.21%	26.92%	28.84%
10	25.50%	26.21%	27.65%

Table 5.2: Comparison of our method with other methods that use STIP (HOG/HOF) and MBH features on HMDB dataset.

Method	Accuracy(%)
HOG/HOF [3]	20.0
C2 [3]	23.0
Action Bank [12]	26.9
Dense Trajectory (MBH) [44]	43.2
BoW with STIP + SVM [Kernel Map]	21.0
Our method + MLR (STIP)	27.12
Our method + NN without dropout (STIP)	27.38
Our method + NN with dropout (STIP)	29.56
BoW with MBH + SVM [Kernel Map]	36.6
Our method + MLR (MBH)	42.42
Our method + NN without dropout (MBH)	43.48
Our method + NN with dropout (MBH)	45.29

Finally, the Table 5.3 shows the mean per-class action classification accuracies on UCF101 dataset. Using dropout improves the accuracy results over weakly supervised method proposed in Section 4.3. Our best accuracy on UCF101, which is 78.77%, was obtained by combining the MBH and STIP features together and using dropout.

Table 5.3: Mean per-class action classification accuracies for UCF101 dataset using STIP and MBH features.

Method	Accuracy(%)
Global BoW [Harris3D + STIP] [4]	43.90
BoW + SVM [Kernel Map] (STIP)	43.94
Our method + MLR (STIP)	51.59
Our method + NN with dropout (STIP)	53.35
BoW + SVM [Kernel Map] (MBH)	65.28
Our method + MLR (MBH)	71.84
Our method + NN with dropout (MBH)	74.24
Our method + NN with dropout (MBH + STIP)	78.77

CHAPTER 6: CONCLUSION AND FUTURE WORK

6.1 Summary of Contributions

In this dissertation, we have addressed the problem of human action recognition and localization in unconstrained videos under a number of challenging conditions which include background clutter, occlusions, camera motion, different camera viewpoints and changes in illumination.

First, we introduced a technique for detecting actions in both space and time by representing the videos as a collection of sequential 2D video projections. This video representation enables us to apply the best techniques from object detection to the action detection problem. We also introduced a novel method for searching the 2D projections to localize actions, termed Two-Point Sub-window Search (TPSS). We evaluated the action detection performance of our proposed method on Microsoft Research (MSR) Action II dataset which is a particularly difficult dataset for action detection since the video clips are longer than the duration of the actions and each video clip contains multiple instances of the same or different actions. Our experiments show that video projection outperforms the latest results on action detection in a direct comparison.

Second, we proposed a weakly-supervised probabilistic model for localizing actions in videos. Different from the above method and other supervised methods, our proposed method does not require the ground truth action locations of the actions during training. We ran experiments on two different datasets, MSR Action II and UCF Sports, demonstrating that our weakly-supervised approach is able to localize actions as well as or better than the methods trained on hand-annotated data. We extended this method to action recognition datasets where the training and test data are made up of short video clips that have been tightly trimmed temporally to contain only the action. We demonstrated that localizing the discriminative regions in videos that are likely to contain

actions improves the action recognition accuracies over the standard bag-of-words models which ignore the spatial location of actions. Our results on UCF Sports, HMDB and UCF101 datasets were better than or comparable to the state-of-the-art recognition systems on those datasets.

Third, we addressed the overfitting problem in action recognition datasets with large number of classes. We integrated a recently introduced technique, called dropout, for preventing the overfitting in neural networks into our action recognition framework. The dropout procedure can be viewed as an efficient way of performing model averaging with neural networks. The overfitting on the training data is reduced by setting the output of hidden units in the neural network to zero with a probability of 0.5. The units that are dropped out do not contribute to the forward passing and backpropagation. We evaluated the performance our proposed framework with dropout on HMDB and UCF101 dataset. Our results show that using dropout improves the accuracy results on those dataset over the methods that do not use dropout.

6.2 Future Work

In this section, we present some possibilities for future directions to further the research carried out in this dissertation.

The weakly-supervised action localization and recognition model, introduced in Chapter 4, recognizes actions by localizing the discriminative sub-regions in video frames that correspond well to action locations. One limitation of this model is that the sub-region size is fixed during training and all the localizers use the same sub-region size. This can be modified so that each localizer uses a different sub-region size. In this way, different localizers would search for discriminative regions at different spatial scales.

In Chapter 5, we showed that using dropout reduces the overfitting and improves the accuracy

results. However, we applied dropout only for training the classification parameters and used the same localization parameters learned in Section 4.3.2. Using dropout also for training the localization parameters may further improve the performance of our model. This can be implemented by modifying the model introduced in Section 4.3 so that the localizers are dropped-out with a probability of 0.5 during training. This could prevent the complex co-adaptations on localizer data.

Recently, deep convolutional neural networks have been successfully used for object recognition and achieved better accuracy results than previous state-of-the-art on large image classification datasets [18]. Our weakly-supervised model introduced in this dissertation has similarities to the convolutional neural networks. In our model, the estimated histogram of a video frame is computed by aggregating the sub-region histograms into a single histogram as explained in Section 4.3.1.2. This is similar to the convolutional layer in a deep neural network where the filter size is the same as the frame size. However, our system uses sub-region histograms instead of actual image pixels. Building on success of deep convolutional neural networks we can modify our model so that smaller filter sizes are used for aggregating the sub-region histograms. In this way, only the histograms from neighboring sub-regions would be aggregated instead of combining the histograms from all sub-regions within a frame. The output of this convolutional layer can be applied as an input to the next convolutional layer until we have a single histogram to represent the frame. Introduction of those additional layers may learn more complex representations than the currently proposed model.

LIST OF REFERENCES

- [1] C. Schuldt, I. Laptev, and B. Caputo, “Recognizing human actions: a local svm approach,” in *ICPR*, 2004.
- [2] M. Rodriguez, J. Ahmed, and M. Shah, “Action mach: A spatial-temporal maximum average correlation height filter for action recognition,” in *CVPR*, 2008.
- [3] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, “Hmdb: a large video database for human motion recognition,” in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2556–2563, IEEE, 2011.
- [4] A. R. Z. Khurram Soomro and M. Shah, “Ucf101: A dataset of 101 human action classes from videos in the wild,” in *CRCV-TR-12-01*, 2012.
- [5] L. Cao, Z. Liu, and T. Huang, “Cross-dataset action detection,” in *Proceedings of Computer Vision and Pattern Recognition*, 2010.
- [6] C. Lampert, M. Blaschco, and T. Hofmann, “Beyond sliding windows: Object localization by efficient subwindow search,” in *Proceedings of Computer Vision and Pattern Recognition*, 2008.
- [7] J. Yuan, Z. Liu, and Y. Wu, “Discriminative subvolume search for efficient action detection,” in *CVPR*, 2009.
- [8] T. Lan, Y. Wang, and G. Mori, “Discriminative figure-centric models for joint action localization and recognition,” in *ICCV*, 2011.
- [9] M. Raptis, I. Kokkinos, and S. Soatto, “Discovering discriminative action parts from mid-level video representations,” in *CVPR*, pp. 1242–1249, 2012.

- [10] N. Shapovalova, A. Vahdat, K. Cannons, T. Lan, and G. Mori, “Similarity constrained latent support vector machine: An application to weakly supervised action classification,” in *ECCV*, 2012.
- [11] H. Wang, A. Klaser, C. Schmid, and C.-L. Liu, “Action recognition by dense trajectories,” in *CVPR*, 2011.
- [12] S. Sadanand and J. J. Corso, “Action bank: A high-level representation of activity in video,” in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 1234–1241, IEEE, 2012.
- [13] S. Singh, A. Gupta, and A. A. Efros, “Unsupervised discovery of mid-level discriminative patches,” in *ECCV*, pp. 73–86, 2012.
- [14] W. Brendel and S. Todorovic, “Learning spatiotemporal graphs of human activities,” in *ICCV*, pp. 778–785, 2011.
- [15] W. Yang, Y. Wang, A. Vahdat, and G. Mori, “Kernel latent svm for visual recognition,” in *Advances in Neural Information Processing Systems (NIPS)*, 2012.
- [16] A. Vedaldi and A. Zisserman, “Sparse kernel approximations for efficient classification and detection,” in *CVPR*, 2012.
- [17] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *CoRR*, vol. abs/1207.0580, 2012.
- [18] A. Krizhevsky, I. Sutskever, and G. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25*, pp. 1106–1114, 2012.

- [19] P. Turaga, R. Chellappa, V. S. Subrahmanian, and O. Udrea, "Machine recognition of human activities: A survey," *IEEE Trans. Cir. and Sys. for Video Technol.*, vol. 18, pp. 1473–1488, 2008.
- [20] D. Weinland, R. Ronfard, and E. Boyer, "A survey of vision based methods for action representation, segmentation and recognition," in *CVIU*, 2011.
- [21] R. Poppe, "A survey on vision-based human action recognition," in *IVC*, 2010.
- [22] A. Bobick and J. Davis, "The recognition of human movement using temporal templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, 2001.
- [23] A. Yilmaz and M. Shah, "Actions as objects: a novel action representation," in *Proceedings of Computer Vision and Pattern Recognition*, 2005.
- [24] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," in *Proceedings of International Conference on Computer Vision*, 2005.
- [25] Y. Ke, R. Sukthankar, and M. Hebert, "Event detection in crowded videos," in *Proceedings of International Conference on Computer Vision*, 2007.
- [26] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *VS-PETS*, 2005.
- [27] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *CVPR*, 2008.
- [28] M. Marszalek, I. Laptev, and C. Schmid, "Actions in context," in *CVPR*, 2009.
- [29] L. Ballan, M. Bertini, A. D. Bimbo, L. Seidenari, and G. Serra, "Effective codebooks for human action categorization," in *ICCV workshop on Video-oriented Object and Event Classification (VOEC)*, 2009.

- [30] J. Liu, J. Luo, and M. Shah, “Recognizing realistic actions from videos ”in the wild”,” in *CVPR*, 2009.
- [31] A. Kovashka and K. Grauman, “Learning a hierarchy of discriminative space-time neighborhood features for human action recognition,” in *CVPR*, 2010.
- [32] J. Liu and M. Shah, “Learning human actions via information maximization,” in *CVPR*, 2008.
- [33] J. Liu, Y. Yang, and M. Shah, “Learning semantic visual vocabularies using diffusion distance,” in *CVPR*, 2009.
- [34] J. C. Niebles, H. Wang, and L. Fei-fei, “Unsupervised learning of human action categories using spatial-temporal words,” in *In Proc. BMVC*, 2006.
- [35] I. Laptev and T. Lindeberg, “Space-time interest points,” 2003.
- [36] I. Laptev, B. Caputo, and T. Lindeberg, “Local velocityadapted motion events for spatio-temporal recognition,” *CVIU*, pp. 207–229, 2007.
- [37] I. Laptev, “On space-time interest points,” in *IJCV*, 2005.
- [38] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *In CVPR*, pp. 886–893, 2005.
- [39] A. Klser, M. Marszaek, and C. Schmid, “A spatio-temporal descriptor based on 3d-gradients,” in *In BMVC08*.
- [40] P. Scovanner, “A 3-dimensional sift descriptor and its application to action recognition abstract,” in *Proceedings of the International Conference on Multimedia*, 2007.
- [41] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

- [42] H. Wang, M. M. Ullah, A. Klaser, I. Laptev, and C. Schmid, “Evaluation of local spatio-temporal features for action recognition,” in *BMVC*, 2009.
- [43] G. Willems, T. Tuytelaars, and L. Gool, “An efficient dense and scale-invariant spatio-temporal interest point detector,” in *Proceedings of the 10th European Conference on Computer Vision: Part II*, pp. 650–663, 2008.
- [44] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, “Action Recognition by Dense Trajectories,” in *IEEE Conference on Computer Vision & Pattern Recognition*, (Colorado Springs, United States), pp. 3169–3176, June 2011.
- [45] A. Klaser, “Learning human actions in videos,” in *PhD thesis, Universit de Grenoble*, 2010.
- [46] S. Z. Masood, A. Nagaraja, N. Khan, J. Zhu, and M. F. Tappen, “Correcting cuboid corruption for action recognition in complex environment,” in *ICCV workshop on Video Event Categorization, Tagging and Retrieval for real-world applications (VECTaR)*, 2011.
- [47] A. Yao, J. Gall, and L. V. Gool, “A hough transform-based voting framework for action recognition,” in *CVPR*, 2010.
- [48] M. R. Amer and S. Todorovic, “A chains model for localizing participants of group activities in videos,” in *ICCV*, 2011.
- [49] H. Boyraz, M. F. Tappen, and R. Sukthankar, “Localizing actions through sequential 2d video projections,” in *Fourth IEEE Workshop on CVPR for Human Communicative Behavior Analysis (CVPR4HB)*, 2011.
- [50] W.-L. Lu, K. Okuma, and J. J. Little, “Tracking and recognizing actions of multiple hockey players using the boosted particle filter,” in *IVC*, 2009.
- [51] N. Ikizler-Cinbis and S. Sclaroff, “Object, scene and actions: combining multiple features for human action recognition,” in *ECCV*, 2010.

- [52] P. Felzenszwalb, D. McAllester, and D. Ramanan, “A discriminatively trained, multiscale, deformable parts model,” in *CVPR*, 2008.
- [53] K. Schindler and L. Van Gool, “Action snippets: How many frames does human action recognition require?,” in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, 2008.
- [54] B. Alexe, T. Deselaers, and V. Ferrari, “What is an object?,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 73–80, IEEE, 2010.
- [55] M. Pandey and S. Lazebnik, “Scene recognition and weakly supervised object localization with deformable part-based models,” in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 1307–1314, IEEE, 2011.
- [56] A. Johnson and M. Hebert, “Using spin images for efficient object recognition in cluttered 3D scenes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, 1999.
- [57] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *CVPR*, 2001.
- [58] K.-K. Sung and T. Poggio, “Example-based learning for view-based human face detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, 1998.
- [59] L. Yeffet and L. Wolf, “Local trinary patterns for human action recognition,” in *ICCV*, 2009.
- [60] N. Dalal, B. Triggs, and C. Schmid, “Human detection using oriented histograms of flow and appearance,” in *In ECCV*, pp. 428–441, 2006.
- [61] H. Wang, A. Klser, C. Schmid, and C.-L. Liu, “Dense trajectories and motion boundary descriptors for action recognition,” *International Journal of Computer Vision*, vol. 103, no. 1, pp. 60–79, 2013.