Towards Usable Multimedia Event Detection

Zhenzhong Lan

CMU-LTI-17-002
May, 2017

Language Technologies Institute
Schoold of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:
Dr. Alexander G. Hauptmann (Chair)
Dr. Bhiksha Raj Ramakrishnan
Dr. Louis-Philippe Morency
Dr. Leonid Sigal (Disney Research)

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
in Language and Information Technologies.

Keywords: Multimedia Event Detection, Action Recognition, Video Analysis, Machine Learning, Neural Networks

To my beloved wife, Betty; and my dear friend, Ken.

Abstract

We often come across events on our daily commute such as a traffic jam, a person running a red light, or an ambulance approaching. These are complex events that human can effortlessly recognize and to which we react appropriately. For computers to be capable of recognizing complex events in a reliable way, like humans can, will facilitate many important applications such as self-driving cars, smart security systems, and elderly care systems. However, existing computer vision and multimedia research focuses mainly on detecting elementary visual concepts (for example, actions, objects, and scenes). Such detection alone are generally insufficient for decision making. Hence we have a pressing need for complex event detection systems. Much research emphasis should be placed on developing such systems.

Compared to elementary visual concept detection, complex event detection is much more difficult in terms of representing both the task and the data that describe the task. Unlike elementary visual concepts, complex events are higher level abstractions of longer temporal spans, and they have richer content with more dramatic variations. The web videos which describe those events are generally much larger in size, noisier in content, and sparser in labels than the images used for concept detection research. Thus, complex event detection introduces several novel research challenges that have not been sufficiently studied in the literature. In this dissertation, we propose a set of algorithms to address such challenges. These algorithms enable us to build a multimedia event detection (MED) system which is practically useful for complex event detection.

The proposed algorithms significantly improve the accuracy and speed of our MED system by addressing the aforementioned challenges. For example, our new data augmentation step and a new way of integrating multi-modal information significantly reduce the impact of the large event variation problem; our two-stage Convolutional Neural Network (CNN) training method allows us to get in-domain CNN features using noisy labels; our new feature smoothing technique is a thorough solution to the problem that noisy and uninformative background contents dominate the video representations; and so forth.

We have implemented most of the proposed methods into the CMU-Elamp system. They are one of the major reasons for its leading performances in the TRECVID MED competition 2011 $\sim$ 2015, the most representative task for MED. Our governing aim, however, has been to uncover enduring insights that can be widely used. Given the complexity of our task and the significance of those improvements, we believe that our algorithms and lessons derived could be generalized to other tasks. Indeed, our methods have already been used by other researchers on tasks such as medical video analysis and image segmentation.

Acknowledgments

I would like to express my deepest gratitude to my PhD advisor, Alex, for supporting me over the years. What I learned from Alex is much more than being a good researcher. He also taught me how to be a better person. His integrity, kindliness, humility, and detail oriented personality profoundly shaped how I do things and treat people.

I am grateful to my committee: Bhiksha, LP, and Leon, for providing a lot of useful suggestions and insights in preparing my dissertation.

A very special gratitude goes out to my dear friend, Ken, for spending countless days and nights with me on revising papers and practicing talks.

I am also grateful to my current and past lab mates, Arnold, Shoou-I, Yi, Lei, Wei, Lu, Xuanchong, Xiaojun, Bernie, Zexi, Ye, and many others. It was fantastic to have the opportunity to work with these guys.

I would also like to thank LTI Academic coordinator, Stacey, who took care of me and other PhD students like an elder sister.

And of course, to my family, especially my wife Betty, for being by my side all the time.

And finally, last but by no means least, also to everyone that helped me on this journey. For example, Guoqing, for buying food for me and explaining complex algorithms to me; Doug, for being my English tutor for more than one year.

Thanks for all your encouragement!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In our daily life, we often come across dozens of different events such as a meeting, an accident, or a party. In order to make proper decisions and react appropriately to these events, we need to understand what these events are. In most cases, given the visual and/or audio information, humans detect these events with ease. I speak, of course, of normal adults who have no problem with seeing or hearing.

The ease at which human can accomplish these tasks lead us to wonder how we can build machines that perform a similar function – detecting complex events. Digging into this research question will not only lead to a deeper understanding of human visual perception, but also provide practical solutions for many applications. For example, a self-driving car would be able to negotiate traffic more safely if it can detect a person running a red light; an airport smart security system would be more effective if it can identify a person leaving luggage unattended; an elder-care system would be more helpful if it can recognize events like a person falling on the floor; and a video retrieval system can cover much broader range of videos if it can tell what happens inside the videos without human descriptions. Figure 1.1 shows some more specific examples. As illustrated, we would not need to search for our pets if surveillance cameras can tell us that somebody had dumped our cat into a trash bin (Figure 1.1a); it would be easy to find out how safe an intersection is if we can search for the number of accidents that happened within a period of time (Figure 1.1b); the girl in the picture can easily relive her birthday party years later if she were able to search inside her large personal photo and video collections (Figure 1.1c).

Despite its profound research and application values, automatic event detection is immensely

(a) Where is my cat? (b) How often does this happen? (c) When did this happen?

Figure 1.1: Exemplar scenarios where a MED system can play a role in.

complex and difficult. As will be explained later, it not only involves sub-tasks that are far from solved, but also requires computational resources that are too expensive for large-scale application. It is for this reason that most current computer vision and multimedia research focuses on detecting simpler and more elementary concepts such as an object, a scene, or an action. More often than not, recognizing these simple concepts alone is insufficient for decision making. Detecting a bag itself, for example, is not enough for an airport security system to signal an alert.

However, due to the recent advances in machine learning, computer vision, multimedia and computer architecture, it has become much more feasible to address this complex event detection problem. For that reason, since 2010, CMU Informedia lab has started to work with National Institute of Standard and Technology (NIST) to develop an accurate, efficient and customizable Multimedia Event Detection (MED) solution. This dissertation summarizes my contributions on this development. These contributions have been one of the major reasons for the leading performance of our CMU MED system [65] in TRECVID MED competition 2011 $\sim$ 2015, which is the most representative forum for MED.

1.1 Research Problem, Challenges, and Solutions

Broadly speaking, my long-term goal is to enable computers to understand complex events in unconstrained environments using limited resources. In this dissertation, I focus on a more concrete problem of efficiently detecting complex events from web videos. To a large extend, web videos represent the complexity of the real world due to their high diversity in content, style, production qualities, encoding, language, etc. Also, web videos often depict those events that humans are interested in and are gaining popularity as a way of communication. According to Cisco, video content will take up approximately 82% of Internet traffic by 2020 [115].

In this dissertation, an event is defined by its event name, event definition, description and some exemplar videos, as shown in Figure 1.2. Given the exemplar videos, our task is to find others videos in the database that belong to the same event class. We assume that no video in the database associates with Meta-data. That is to say, the search is purely based on understanding the content of the videos. Compared to elementary concept detection, MED is more challenging for the following reasons.

First, events are much more complex than elementary concepts. An event is a higher level semantic abstraction of video sequences than a concept and consists of multiple concepts. For instance, a *batting a run in* event can be described by multiple objects (e.g., bag, glove, and fence), scenes (e.g., indoor and outdoor), and actions (e.g., swinging a bat, running, and cheering). Because of their complexities, the variances within each event are often large while the differences among events can be subtle. For example, to recognize an event called "attempting a board trick", our algorithms need to discover the similarities among "snow boarding", "skate boarding", and "surf boarding", while distinguishes them from 'skiing', "roller skating", and "swimming".

Second, an elementary concept can be depicted by one or several frames but an event often lasts much longer. A *baseball*, for example, can be recognized from a frame, whereas a full event where a player bats a run in may last from less than ten seconds to several minutes. Normally,

Event Name: Batting a run in
Definition: Within a single play during a baseball-type game, a batter hits a ball and one or more runners (possibly including the batter) scores a run.
Evidential Description:
scene: outdoor or indoor ball fields (official or ad hoc), during the day or night
objects/people: baseball, bat, glove, crowd in background, fence, pitchers mound, bases, other players, officials
activities: pitching, swinging a bat, running, throwing a ball, cheering or clapping, making a call, crossing home plate
Exemplars:

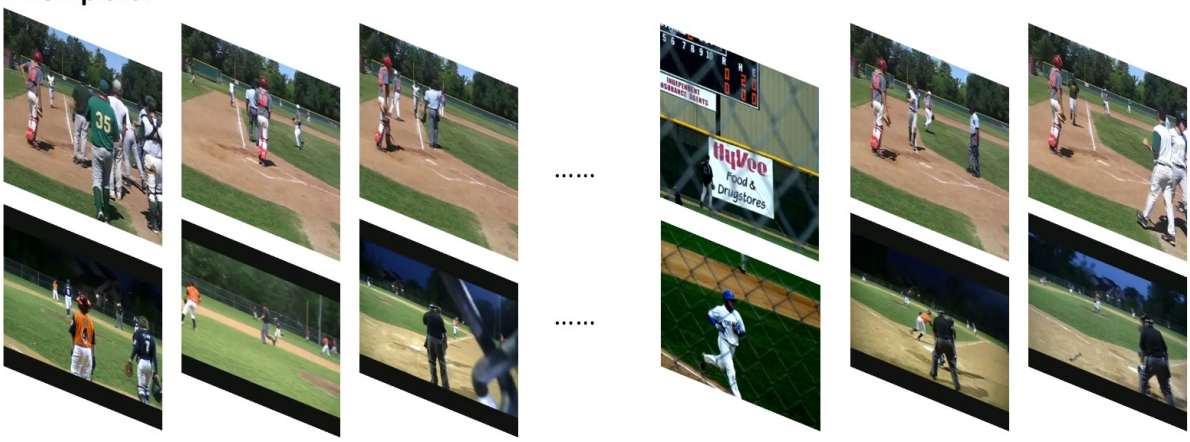


Figure 1.2: An exemplar event of batting a run in.

from within one or a few frames, it is hard to tell whether a runner scores a run or not. Therefore, we need to handle a much larger size of data for MED than for elementary concept detection. It is for this reason that efficiency is a critical factor in developing a MED system.

Third, high-quality event labels are more difficult to get than concept labels. This difficulty is mainly because labeling long videos is much more time-consuming than labeling short clips or images. It is also because the subjective definition of an event. For example, without further specification, it is hard to tell what counts a trick in the event of "attempting a board trick". Consequently, the number of the manually labelled videos we have is much smaller than the number of manually labelled images. For example, ImageNet [23] is one of the largest manually labelled image datasets and has 14 millions labelled images, while TRECVID MED2014 [62], which is one of the largest manually labelled image datasets, has less than 250,000 labelled videos.

Finally, event detection often requires various sources or multi-modal information while elementary concept detection often deals with single-source information. For example, in MED, we need to detect visual information like object, motion, scene, text, and audio information like speech and music that occur in the videos. Whereas, object or scene recognition only needs images.

As a result, MED introduces several novel research challenges that have not been sufficiently studied in the literature. This dissertation deals with the following challenges:

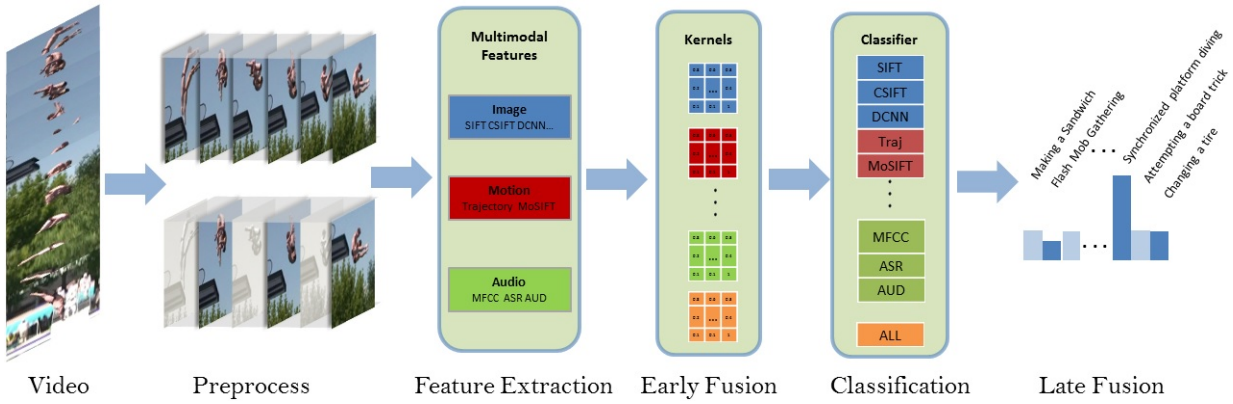


Figure 1.3: CMU Informedia MED System

- The first major challenge of building an event detector is the high variation within an event and subtle differences among events. To address this problem, we propose a video preprocessing step that reduces the speed variance of motions and a feature encoding method to mitigate the impact of dominated factors such as repeated patterns in the background.
- The second challenge we faced is how to maintain the efficiency given the size of the data. We deal with this problem by improving the system in the following three aspects. These three improvements include a new Convolutional Neural Network (CNN) feature, an efficient space-time encoding method, and a systematical study of the efficacy and efficiency of different features given the existence of others.
- The third challenge comes from that fact that video labeling is expensive. To better utilize the sparse labels we have, we propose a two-stage training procedure to improve our CNN features and a reranking algorithm to utilize information from other event classes. Additionally, we propose an unsupervised local video feature learning method that combines the advantages of handcrafted and learning based methods.
- Finally, we also face the challenge of how to smartly fuse these multiple sources of information in MED. As a solution, we propose a new feature fusion methods that combines the advantages of traditional early fusion and late fusion methods [109].

1.2 Dissertation Overview

This section provides an overview of the dissertation and a comprehensive picture about how we address the challenges described in the last section. As shown in Figure 1.3, CMU MED system uses a standard video classification pipeline, in which we first preprocess the videos to gain some desired characteristics. We then extract and encode multimodal features followed by mapping different features into kernel spaces and performing early fusion [109]. After that, we train a set of classifiers and apply them to the testing data to get their initial predictions. Late fusion and some postprocessing steps are carried out at the end to get the final predictions. I will present the proposed methods in the same order.

A briefly review of MED related works will be shown in Chapter 2 followed by a description

of evaluation datasets and metrics in Chapter 3.

Chapter 4 introduces a new algorithm called Multi-skIp Feature Stacking (MIFS) as a new preprocessing step for our MED system that helps to achieve speed invariance for video features [CVPR'15]. Chapter 5 shows experiments on the CNN video features that significantly improve the accuracy of our system [TRECVID'13, MMM'14]. This chapter also introduces a new CNN-based video feature called Deep local Video Feature (DoVF). Chapter 6 introduces an unsupervised descriptors learning methods to learn descriptors for a state-of-the-art handcrafted motion feature. Despite its superior performance compared to state-of-the-art methods, we did not use it for MED because of the high cost in processing local features. I show it here in the hope that it can provide lessons for future unsupervised global feature learning development. Also, it has the theoretical value of showing the connection between traditional handcrafted methods and CNNs. Chapters 7 and 8 detail two simple improvements for the feature encoding step of our MED system. The first one, called Rank Normalization (RNorm), fundamentally addresses the problem that uninformative background patterns dominate the video representations. The second one, called Space-Time Extended descriptors (STED), replaces the classic spatial pyramid matching (SPM) method as a more efficient way to encode space-time information [CVPR'15, MM'15]. Chapter 9 introduces a new fusion method called double fusion, which incorporates the advantages of early fusion and late fusion by fusing before and after classification [MMM'12, MTA'14]. Chapter 10 elaborates the training-free reranking technique called Multi-class Iterative Re-ranking (MIR) that captures relationships among multiple event classes and significantly improves the ranking performance of our MED system. MIR is a postprocessing step for our MED system [MM'15]. Chapter 11 presents a study comparing the cost and efficiency tradeoffs of multiple features for MED.

Finally, Chapter 12 concludes this dissertation by summarizing the main lessons I have learned in undertaking this project as well as the promising directions that can be explored in the future.

Chapter 2

Related Work

The field of multimedia event detection has changed drastically over the past few years. In this chapter, I will briefly review some important works that has contributed to the improvement of MED. More detailed reviews about each problem we addressed can be found in its corresponding chapter.

2.1 Features

Previous works [84] [116] [26] [77] on developing and evaluating features for MED can be divided in two main categories depending whether they used low-level features or high-level semantic concepts. Yang *et al*. [134] and Tamrakar *et al*. [116] proposed to evaluate the individual performance of different low-level visual features (for ecample, SIFT [79], STIP [69], and Trajectories [126]) as well as their combinations. Meler *et al*. [84], Ebadollahi *et al*. [26] and Liu *et al*. [77] focus on testing high-level features' performance on event recognition.

In terms of motion features, the trajectory based approaches [53, 82, 113, 125, 126], especially the Improved Dense Trajectory (IDT) method proposed by Wang *et al*. [125, 126], together with the FV encoding [96] is the basis of the current state-of-the-art performances on several benchmark action recognition datasets. Peng *et al*. [93] further improved the performance of IDT by increasing the codebook sizes and fusing multiple coding methods. However, with the rise of deep neural network methods, these traditional methods are gradually becoming less relevant.

Motivated by this success of CNNs, researchers are working intensely towards developing CNN equivalents for learning video features. Several accomplishments have been reported from using CNNs for action recognition in videos [121, 130, 140]. Karpathy *et al*. [54] trained deep CNNs through one million weakly labelled YouTube videos and reported moderate success while using it as a feature extractor. Simonyan & Zisserman [106] demonstrated a result competitive to IDT [126] through training CNNs using both sampled frames and stacked optical flows. Wang *et al*. [127–129] show multiple insightful analysis about how to improve two-stream frameworks and find several useful observations including pre-training two-stream ConvNets, using a smaller learning rate, and using deeper networks, etc. With these observations, their approach finally outperforms IDT [126] by a large margin on UCF101 dataset [110]. However, all these approaches

rely on shot-clip predictions to get the final video scores without using global features.

In image classification, we often take a whole image as the input to CNNs. However, in video classification, because of the much larger size of videos, we often use sampled frames/clips as inputs. One major problem of this common practice is that video-level label information can be incomplete or even missing at frame/clip-level. This information mismatch leads to the problem of false label assignment, which motivates another line of research that tries to do CNN-based video classification beyond short snippets. Ng *et al*. [87] reduced the dimension of each frame/clip using a CNN and aggregated frame-level information using LSTM. Varol *et al*. [122] stated that Ng *et al*. 's approach is suboptimal as it breaks the temporal structure of videos in the CNN step. Instead, they proposed to reduce the size of each frame and use longer clips (e.g., 60 frames vs 16 frames) as inputs. They managed to gain significant accuracy improvements compared to shorter clips. However, the way they reduced the spatial resolution comes at a cost of a large accuracy drop. In the end, the overall accuracy improvement is less impressive. Wang *et al*. [129] experimented with sparse sampling and jointly train on the sparsely sampled frames/clips. In this way, they incorporate more temporal information while preserving the spatial resolution. Diba *et al*. [24], Qiu *et al*. [97], and We [68] took a step forward along this line by using the networks of Wang *et al*. [129] to scan through the whole video, aggregate the features (output of a layer of the networks) using some pooling methods, and fine-tune the last layer of the network using the aggregated features. We believe that these approaches are still sub-optimal as they again break the end-to-end learning into a two-stage approach. However, because these approaches are currently the best at incorporating global temporal information, they represent the current state-of-the-art in this field.

Another direction of CNN-based video feature learning is to address the speed bottleneck problem from using traditional way to pre-compute optical flow for temporal stream CNN. Compared to the CNN step, the optical flow calculation step is computationally expensive. It is the major speed bottleneck of the current two-stream approaches. As an alternative, Zhang *et al*. [141] proposed to use motion vectors, which can be obtained directly from compressed videos without extra calculation, to replace the more precise optical flow. This simple improvement brought more than 20x speedup compared to the traditional two-stream approaches. However, this speed improvement came with an equally significant accuracy drop. The encoded motion vectors lack fine structures, and contain noisy and inaccurate motion patterns, leading to much worse accuracy compared to the more precise optical flow [139]. These weaknesses are fundamental and can hardly be improved. Another more promising approach is to learn to predict optical flow using supervised CNNs, which is closer to our approach. There are two representative works in this direction. Ng. *et al*. [88] used optical flow calculated by traditional methods as supervision to train a network to predict optical flow. This method avoids the pre-computation of optical flow at inference time and greatly speeds up the process. However, as we will demonstrate later, the quality of the optical flow calculated by this approach is limited by the quality of the traditional flow estimation, which again limits its potential on action recognition. The other representative work is by Ilg *et al*. [40] and uses the network trained on synthetic data where ground truth flow exists. The performance of this approach is again limited by the quality of the data used for supervision. The ability of synthetic data to represent the complexity of real data is very limited. Actually, in Ilg *et al*. [40]'s work, they show that there is a domain gap between real data and synthetic data. To address this gap, they simply grew the synthetic data to narrow the

gap. The problem with this solution is that it may not work for other datasets and it is infeasible to do this for all datasets.

In terms of multi-scale feature representations [3, 76], a multi-scale image key-point detector was initially proposed by Lindeberg [75]. Lowe [79] used it to detect scale invariant key points using Laplacian pyramid methods, where Gaussian smoothing is used iteratively for each pyramid level. Simonyan & Zisserman [107] reported a significant performance improvement on Imagenet Challenge 2014 by using a multi-scale CNNs. In video processing, Space Time Interest Points (STIP) [69] extends SIFT [79] to the temporal domain by finding the scale invariant feature points in 3D space. Shao *et al.* [105] also try to achieve scale invariance for action recognition using 3-D Laplacian pyramids and 3D Gabor filters. However, without awareness of the fundamental difference between image and video processing, [105] was not very successful when compared to the state-of-the-art methods.

2.2 Encoding

For local feature encoding, Fiher Vector (FV) [95] and Vector of Local Aggregated Descriptors (VLAD) [46] are the most significant encoding methods and have been widely used to encode various features including IDT [126] and CNN features [7, 25, 132]. There are very similar encoding methods [6] and both have been popular for image and video classification [6, 85, 96, 126, 132]. In the original scheme [44, 95], they either do not require post processing [95] or use only ℓ_2 normalization [44]. Although ℓ_2 normalization can reduce the influence of background information and transform the linear kernel into an ℓ_2 similarity measurement [96], it does not disperse the data. As a result, the original FV encoding method showed inconclusive results compared to other state-of-the-art encoding methods [95]. It is the introduction of Power Normalization (PNorm) [6, 96] that significantly improved the performance of those encoding methods and thus made them useful in practice. PNorm alleviates the problem of sparse and bursty distribution of FV and VLAD. Later on, as a special design for VLAD, Arandjelovic and Zisserman [6] proposed Intra-normalization (INorm) to further reduce the bursty distribution problem of VLADs. Jégou and Chum [44] used PCA to decorrelate a low dimensional representation and adopted multiple clustering to reduce the quantization errors for VLADs. Nonetheless, those methods only partially alleviated the burstiness problem [45]. The sparsity of the encoded descriptors still depends on that of the sparsity of the original representations.

There has been a large amount of work in building representations that keep spatial information of image patterns. Among them, spatial pyramid matching [71] is the most popular one. However, building spatial pyramids requires dimensions that are orders of magnitude higher than the original spatial invariant representations and hence make it less suitable for high dimensional encoding methods such as FV [96] and VLAD [6]. Spatial FV [57] and spatial augmentation [83, 100] provide more compact representations to encode spatial information and show similar performance as spatial pyramid methods. Few approaches consider encoding global temporal information into video representations. Oneata *et al.* [91] show that better action recognition performance can be achieved by dividing videos into two parts and encoding each one separately. Codella *et al.* [20] try to use a temporal pyramid for event detection. They use n temporal segments, where n incrementally increases from 1 to 10.

2.3 Classifiers

Given the encoded features, state-of-the-art methods often use 'one versus the rest' SVM, which does not consider the relationships among action classes. To model those relationships, Bergamo & Torresani [13] suggested a meta-class method for identifying related image classes based on mis-classification errors from a validation set. Hou *et al*. [37] identified similar class pairs and grouped them together to train 'two versus the rest' classifiers. By combining 'two versus the rest' with 'one versus the rest' classifiers, they observed significant improvements from baselines. However, when calculating the similarities, most of these methods only take a subset of classes into consideration, which restricts the improvements they can make. Neural network based methods such as mixtures of experts also shown superior performance on video classification [2]. However, due to the size of the parameters, they tend to requires more data for training. Another interesting direction is training classifiers that takes noisy labels into consideration [29, 112]. This noisy label learning is of particular useful in Internet video learning. However, because the main focus on this dissertation is video feature learning, we will not go into too much details of this noisy label classifier learning topic.

2.4 Fusion

Historically, researchers in image and video retrieval [41, 52, 73, 109, 136] found that a combination of multi-modality information almost always helps in obtaining high retrieval accuracy. In general, researchers use two types of combination strategies, namely early fusion and late fusion [109]. Early fusion combines features before performing classification, such as multi-kernel learning [21, 33]. Late fusion combines output of classifiers from different features, such as average fusion, committee voting [120] and co-regularized least squared regression [15]. There is no universal conclusion as to which strategy is preferred for multimedia content analysis and retrieval. Snoek *et al*. [109] found that early fusion is better than late fusion in TRECVID 2004 semantic indexing task. By studying data on the TRECVID 2006 semantic indexing (SIN) task, Ayache *et al*. [8] found that early fusion gets better results on most of tasks while late fusion is more robust and can handle some more complex tasks.

Chapter 3

Datasets and Evaluation Criteria

In this section, we describe the benchmark datasets and evaluation criteria we used to evaluate our algorithms in this thesis. Besides evaluating on MED tasks, we also evaluate our algorithms on several action recognition tasks. There are two reasons for us to choose action recognition tasks over others such as scene and object recognition. First of all, it is the concept recognition task that is most similar to MED tasks in terms of difficulty level and the data it works on. Both of them use video data and need to gather information from multiple frames. Therefore, the improvements on action recognition tasks can often be used for MED tasks. Second, most of the action recognition tasks focus on evaluating the motion part of an algorithm, thus they are useful for us to determine where an improvement comes from. In the rest of this chapter, we provide the detailed description of the two tasks and their corresponding datasets and evaluation criteria. Together, we have evaluated on eleven datasets including six MED datasets and five action recognition datasets. However, most of these datasets are quite similar in terms of sources and contents. Therefore, our results can be bias towards these datasets and possibly that they will not generalize to some new datasets such as surveillance datasets. However, in the process of designing these algorithms, we try to avoid exploring the data bias so that we can reduce the possibility of overfitting.

3.1 Multimedia Event Detection

3.1.1 Definition

Recalling what we discussed in in section 1.1, the goal of an event detection task is to detect events of interest such as *Birthday Party* and *Parade*, solely based on the video content.

3.1.2 Datasets

Started at 2010, NIST has gradually built up a video event database that contains 8000 hours of videos and over 40 events, which is by far the largest event detection collection. Over the years, we have used the following five benchmark datasets including MED10, MED11, MEDTEST13, KINDREDTEST, and MEDTEST14, of which MEDTEST14 is the latest and major dataset that

P01: Assembling shelter	E22: Cleaning an appliance
P02: Batting a run	E23: Dog show
P03: Making a cake	E24: Giving directions to a location
E01: Attempting a board trick	E25: Marriage proposal
E02: Feeding an animal	E26: Renovating a home
E03: Landing a fish	E27: Rock climbing
E04: Wedding ceremony	E28: Town hall meeting
E05: Working on a woodworking project	E29: Winning a race without a vehicle
E06: Birthday Party	E30: Working on a metal crafts project
E07: Changing a vehicle tire	E31: Beekeeping
E08: Flash mob gathering	E32: Wedding shower
E09: Getting a vehicle unstuck	E33: Non-motorized vehicle repair
E10: Grooming an animal	E34: Fixing musical instrument
E11: Making a sandwich	E35: Horse riding competition
E12: Parade	E36: Felling a tree
E13: Parkour	E37: Parking a vehicle
E14: Repairing an appliance	E38: Playing fetch
E15: Working on a sewing project	E39: Tailgating
E21: Attempting a bike trick	E40: Tuning musical instrument

Table 3.1: MED event ID and name. Event ID staring with 'P' and 'E' indicate events in 2010 and after 2010.

we will use throughout this thesis.

For TRECVID MED 2010, we used both the annotated training and testing data, which consists of 114 hours of video clips and three events with event ID P01 to P03 in Table 3.1.

The MED2011 dataset has about 1570 hours of video clips which includes MED 2011 Training dataset (about 370 hours) and MED 2011 Testing dataset (about 1200 hours). There are 15 events including E01 to E015. For this dataset, we test event detection on the training data where the labels have been released by NIST. We randomly split the 15 events into equal sizes of training and testing data and put all the video clips labeled as NULL in MED 2011 into testing set. After the splitting, we have 3135 video clips for training and a set of 6687 videos for testing. We also report results of the whole MED2011 dataset evaluated by NIST.

The MEDTEST2013 and KINDDREDTEST are two standard system evaluation datasets released by NIST in 2013. Each of them contains around 10 percent of the whole MED collection and has 20 events ranging from E05 to E15 and E21 to E30 (Table 3.1). They consist of two tasks, i.e. EK100 and EK10. As their names suggested, EK100 task has 100 positive training samples while EK10 has 10. For both tasks, they have around 5000 background samples. Together, each dataset has 8000 training samples and 24000 testing samples.

The MEDTEST14 dataset was released by NIST in 2014. It has similar amount of data as MEDTEST13 and KINDDRESTEST datasts. The events in this dataset range from E21 to E40 (Table 3.1). It also has the EK100 and EK10 scenarios. Because MEDTEST14 is the latest and the most difficult evaluaton datasets in the history of MED, I will use it as the major datasets to

(a) HighJump

(b) Kick

(c) HandShake

(d) HighJump

Figure 3.1: Examples frames from (a) UCF101 / UCF50, (b) HMDB51, (c) Hollywood2, (d) Olympic Sports.

Datasets	Source	Mean Duration	Clips	Classes	Size	Difficulty level
HMDB51	YouTube/Movie	3.14s	6766	50	2.1 G	Middle
UCF101	YouTube	7.21s	13320	101	6.8 G	Easy
UCF50	YouTube	7.44s	6681	50	3.2 G	Easy
Hollywood2	Movie	11.55s	1707	12	8.6 G	Hard
Olympic	YouTube	7.74s	783	16	11 G	Easy

Table 3.2: Meta data for action datasets.

evaluate all our algorithms. Other datasets are only used for historical reasons or when additional evaluation is need.

Unless specified, we use mean average precision (MAP) as our evaluation criteria. It is a very popular measure in information retrieval. In our case, given a ranked list returned for an event class, the precision indicates the fraction of videos belonging to that class at each ranking position. Average precision (AP) is the mean of the precision for each position where there is a correctly labeled video. MAP further takes the mean of APs across all the event classes.

3.2 Action Recognition

3.2.1 Definition

Given a collection of short clips of videos that usually last a few seconds, the goal of an action recognition task is to determine if they contain actions of interest such as *running* and *kissing*, solely based on the video content. Unlike MED tasks where the videos last much longer, action recognition often uses short video clips.

3.2.2 Datasets

We use five widely used action recognition benchmark datasets including the HMDB51 [59], UCF101 [110], UCF50 [98], Hollywood2 [81] and Olympic Sports datasets [89]. Figure 3.1 show some example frames of these datasets.

As summarized in Table 3.2, the HMDB51 dataset [59] has 51 action classes and 6766 video clips extracted from digitized movies and YouTube. Kuehne *et al.* . [59] provides both the original videos and the stabilized ones. We only use the original videos in this paper. The UCF101 dataset [110] has 101 action classes spanning over 13320 YouTube videos clips. We use the standard splits with training and testing videos provided by Soomro *et al*. [110]. The UCF50 dataset [98] has 50 action classes spanning over 6618 YouTube videos clips that can be split into 25 groups. The video clips in the same group are generally very similar in background.The Hollywood2 dataset [81] contains 12 action classes and 1707 video clips that are collected from 69 different Hollywood movies. There are 12 action classes such as answering a phone, driving a car and standing up. Each video of this dataset may contain multiple actions. We use the clean training dataset and standard splits with training (823 samples) and test videos (884 samples) provided by Marszalek *et al*. [81]. The Olympic Sports dataset [89] consists of 16 athletes practicing sports such as high-jump, pole-vault and basketball lay-up. It has a total of 783 video clips. We use standard splits with 649 training clips and 134 test clips. Note that in this standard split each class only has about 8 testing samples, so the results of this dataset may not be able to reliably evaluate the quality of the model.

We report mean accuracy (MAcc) for HMDB51, UCF101 and UCF50 and mean average precision (MAP) for Hollyowood2 and Olympic Sports datasets as in the original papers. And again, we will only use them we we need to evaluate the motion part of our algorithms. Based on the performance (Macc or MAP) of the state-of-the-art algorithms on these datasets, we also assign a difficulty level for each datasets. As can be seen, the meta data such as length and size has very little impact on determining the performance of the algorithms on these videos.

Chapter 4

Video Preprocessing: Multi-skip Feature Stacking (MIFS)

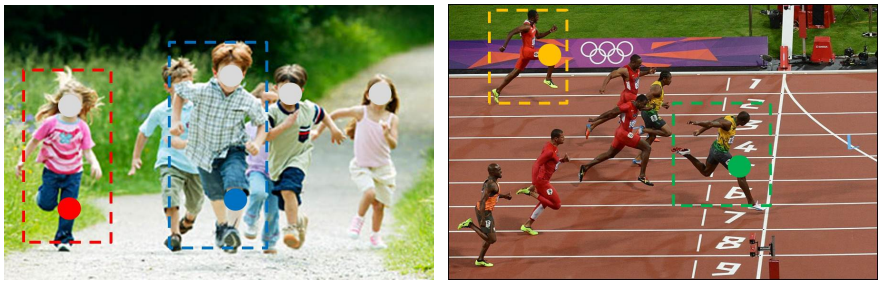
4.1 Introduction

In this chapter, we introduce a novel preprocessing step that helps to generate speed invariant motion representations. Part of this chapter has been published in [62].

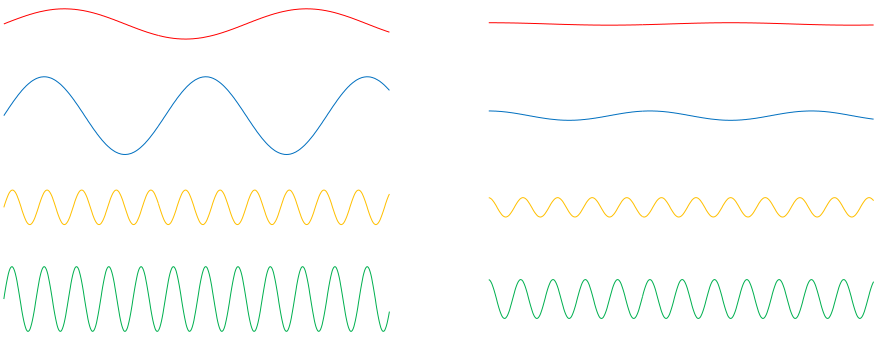
As pointed out by Marr [80] and Lindeberg [76], robust visual representations, or visual features, are of utmost importance for a vision system. The advances of visual representations are not only the chief reasons for tremendous progress in the field, but also lead to a deeper understanding of the information processing in the human vision system. In fact, most of the qualitative improvements to visual analysis can be attributed to the introduction of improved representations, from SIFT [79] to CNNs [103], STIP [69] to IDT [125] . A common characteristic of these several generations of visual features is that they all, in some way, benefit from the idea of multi-scale representation, which is generally viewed as an indiscriminately applicable tool that reliably yields improvements in performance when applied to almost all feature extractors.

At the core of image multi-scale representation is the requirement that no new detail information should be artificially found at the coarse scale of resolution [56]. Gaussian Pyramid, a unique solution based on this constraint, generates a family of images where fine-scale information is successively suppressed by Gaussian smoothing. However, in motion representations, we often desire the opposite requirements. For example, in generating action features using differential filters, we need coarse-scale features to: 1) recover the information that has been filtered out by highpass filters at fine scales, e.g., the red and cyan signals in Figure 4.1c are likely to be filtered out; 2) generate features at higher frequency for matching similar actions at different speeds and ranges of motion, e.g., the orange and green signals in Figure 4.1c. Neither of these requirements can be satisfied with a Gaussian Pyramid representation.

In this chapter, we introduce a new video representation called Multi-skIp Feature Stacking (MIFS). MIFS stacks features extracted by a family of differential filters parameterized with multiple time skips (scales). Through skipping frames, MIFS extracts and represents motions with different magnitudes. MIFS has several attractive properties:



(a) Kids Running (b) Olympics Running



(c) Action signal S (d) ΔS

Figure 4.1: Simplified action signals (4.1c) from "running" actions (4.1a,4.1b) show dramatic differences among subjects and scenes. With such dramatic differences among action signals, a differential operator with single scale is incapable of covering a full range of action frequency and tends to lose low frequency information (the red and cyan signals) .

- It is an broadly applicable tool that can be reliably and easily adopted by any feature extractor with differential filters, like Gaussian Pyramid.
- It generates features that are shift-invariant in frequency space, hence making it easier to match similar actions at different speeds and ranges of motion.
- It stacks features at multiple frequencies, which allows covering a longer range of motion signals compared to conventional action representations.
- It generates feature matrices that have lower conditional numbers and variances, hence higher learnability [104] compared to the conventional original-scale representations, based on our theoretical analysis.
- It significantly improves the performance of state-of-the-art methods, based on experimental results on both action and event benchmarks.
- It exponentially enhances the learnability of the resulting feature matrices. Therefore the required additional number of scales is logarithmic to the bandwidth of the action signals. Empirical studies show that one or two additional scales are enough to recover the information lost by differential operators. Hence the additional computational cost of MIFS is

small.

In the remainder of this chapter, we provide more background information about motion and multi-scale presentations. We then describe MIFS in detail, followed by theoretically proving that MIFS improves the learnability of video representations exponentially. After that, an evaluation of our method on both action recognition and MED tasks is performed. Further discussions including potential improvements are given at the end of this chapter.

4.2 Related Work

There is an extensive body of literature about action recognition and MED in videos; here we just mention a few relevant ones involved with state-of-the-art feature extractors and feature encoding methods. See [4] for an in-depth survey. In conventional video representations, features and encoding methods are the two chief reasons for considerable progress in the field. Among them, the trajectory based approaches [53, 82, 113, 125, 126], especially the IDT method proposed by Wang *et al*. [125, 126], together with the FV [96] yields the current state-of-the-art performances on several benchmark action recognition datasets. Peng *et al*. [93] further improved the performance of IDT by increasing the codebook sizes and fusing multiple coding methods. Some success has been reported recently using CNNs for action recognition in videos. Karpathy *et al*. [54] trained a CNN using 1 million weakly labeled YouTube videos and reported a moderate success on using it as a feature extractor. Simonyan & Zisserman [106] reported a result that is competitive to IDT [126] by training CNNs using both sampled frames and optical flows. MIFS is an broadly applicable tool that can be used to improve all of above mentioned feature extractors.

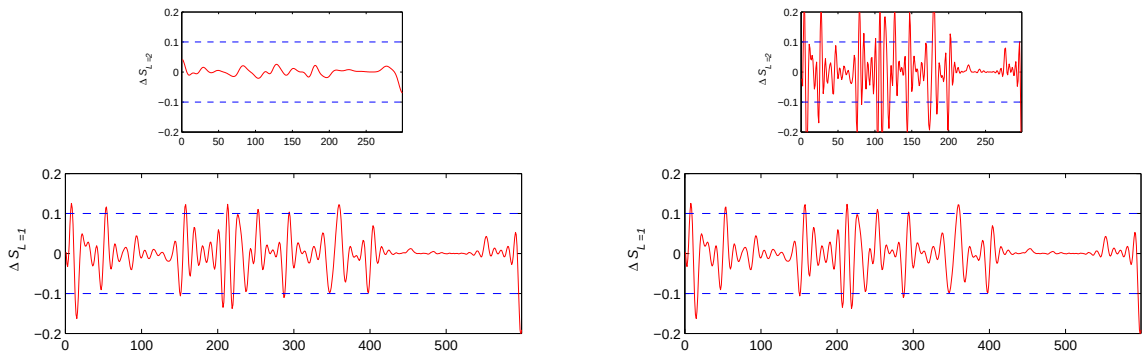
Multi-scale representations [3, 76] have been very popular for most image processing tasks such as image compression, image enhancement and object recognition. Simonyan & Zisserman [107] reported a significant performance improvement on the Imagenet Challenge 2014 by using a multi-scale CNN. In video processing, STIP [69] extends SIFT to the temporal domain by finding the scale invariant feature points in 3D space. Shao *et al*. [105] also try to achieve scale invariance for action recognition using 3-D Laplacian pyramids and 3D Gabor filters. However, without awareness of the fundamental difference between image and video processing, [105] was not very successful when compared to the state-of-the-art methods.

For lab datasets where human poses or action templates can be reliably estimated, Dynamic Time Warping (DTW) [22], Hidden Markov Models (HMMs) [133] and Dynamic Bayesian Networks (DBNs) [92] are well studied methods for aligning actions that have speed variation. However, for noisy real-world actions, these methods have not shown themselves to be very robust.

4.3 Multi-skIp Feature Stacking (MIFS)

We now formalize our notation. For the present discussion a video X is just a real function of three variables:

$$X = X(x, y, t). \tag{4.1}$$



(a) Gaussian Pyramid (b) MIFS

Figure 4.2: Comparison of Gaussian Pyramid and MIFS for a real action signal. The left figure (a) shows that as the level (L) goes higher (from 1 to 2), the resulting features (ΔS) from a differential operator become less prominent. So once a feature has been filtered out (assume the threshold for a feature to be represented is 0.1), it cannot be recovered by higher level features under the Gaussian Pyramid framework. The right figure (b) shows that under MIFS, the features (ΔS) become more prominent as the levels go higher and can represent those signals that have been filtered out at low levels.

The normalized coordinates $(x, y, t) \in R^3$ are the Cartesian coordinates of the video space. Since we focus on temporal domain, we omit (x, y) in further discussion and denote a video as $X(t)$. The length of the video is assumed to be normalized, that is $t \in [0, 1]$. In our model, the content of a video is generated by a linear mixture of k latent action signals:

$$\bar{X} = [\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \cdots, \bar{\mathbf{x}}_k] \,. \tag{4.2}$$

The mixing weight of each latent action signal $\bar{\mathbf{x}}_i$ at time t is denoted as $\alpha_i(t)$. Therefore, a given video is generated as

$$X(t) = \bar{X}\boldsymbol{\alpha}(t) + \boldsymbol{\epsilon}(t) \tag{4.3}$$

$$\boldsymbol{\alpha}(t) = [\alpha_1(t), \alpha_2(t), \cdots, \alpha_k(t)]^{\mathrm{T}} \,. \tag{4.4}$$

where $\boldsymbol{\epsilon}(t)$ is additive subgaussian noise with noise level σ. We assume $\forall i$,

$$|\alpha_i(t)| \leq 1 \quad \mathrm{E}_t\{\alpha_i(t)\} = 0 \tag{4.5}$$

$$\mathrm{E}_t\{\alpha_i(t)^2\} \leq 1 \quad \mathrm{E}_t\{\alpha_i(t) \times \alpha_j(t)|_{i \neq j}\} = 0 \,. \tag{4.6}$$

The feature extractor is assumed to be modeled as a differential operator $\mathcal{F}[\cdot, \tau]$ parameterized with time skip τ. Given a fixed τ, the feature extractor $\mathcal{F}[X(t), \tau]$ generates $T = \lfloor 1/\tau \rfloor$ features.

$$\mathcal{F}[X(t), \tau] = [\mathbf{f}(t_1, \tau), \mathbf{f}(t_2, \tau), \cdots, \mathbf{f}(t_T, \tau)] \,.$$

where $t_1, t_2, \cdots, t_T$ are uniformly sampled on $[0, 1]$. The i-th feature vector $\mathbf{f}(t_i, \tau)$ is generated by

$$\begin{aligned} f(t_i, \tau) &= X(t_i + \tau) - X(t_i) \\ &= \bar{X} \times (\boldsymbol{\alpha}(t_i + \tau) - \boldsymbol{\alpha}(t_i)) + \boldsymbol{\epsilon}(t_i + \tau) - \boldsymbol{\epsilon}(t_i) \,. \end{aligned} \tag{4.7}$$

We can rewrite the feature matrix as

$$\mathcal{F}[X(t), \tau] = \bar{X}P + \sum_{i=1}^{T} \boldsymbol{\epsilon}(t_i + \tau) - \boldsymbol{\epsilon}(t_i)$$

where P is a $k \times T$ matrix, representing the gradients of latent signal weights. That is to say, $P_{i,j} = \boldsymbol{\alpha}_i(t_j + \tau) - \boldsymbol{\alpha}_i(t_j)$.

Most action feature extractors are different versions of $\mathcal{F}$. For example, STIP [69] and DT [125] can be derived from $\mathcal{F}[\cdot, \frac{1}{L}]$, where L is the number of frames in the video.

MIFS stacks multiple $\mathcal{F}[X(t), \tau]$ with different τ. By stacking multiple features with different frequencies, MIFS seeks invariance in the frequency domain via resampling in the time domain. Figure 4.2 shows the difference of Gaussian Pyramid and MIFS for a real signal from an unconstrained video. It is clear that, because of smoothing, Gaussian Pyramid fails to recover signals once they have been filtered out. As the levels go higher, the feature generated by Gaussian Pyramids can only become weaker. While in MIFS, the generated features become more prominent and can be recovered as the levels go higher.

4.4 The Learnability of MIFS

In this section, we first show that under our model in Eq. (4.2), the standard feature extraction method cannot produce a feature matrix conditioned well enough. Then we show that MIFS improves the condition number of the extracted feature matrix exponentially. One of the key novelties of the MIFS is that it also reduces the uncertainty of the feature matrix simultaneously. This reduction is not possible in a naive approach.

4.4.1 Condition number of P under a fixed τ

In this subsection, we will prove, based on the Matrix Bernstein's Inequality [118], that the condition number of P is not necessarily a small number.

In static feature extractors such as SIFT, the weight coefficient matrix $\boldsymbol{\alpha}$ is independent of t. While in a video stream, the action signal is dynamic in t. To measure the dynamic of an action signal, we introduce γ_i as an index.

Definition 1 *A latent action signal is γ dynamic, if given a non-negative constant $c \in [0, 1]$, $\forall \tau \in [0, 1]$,*

$$1 - (1 + c)\exp(-\gamma/\tau) \leq \mathrm{E}_t|\alpha(t)\alpha(t + \tau)| \leq 1 - \exp(-\gamma/\tau) \ ,$$

provided $1 - (1 + c)\exp(-\gamma/\tau) \geq 0$.

The value γ measures how fast the coefficient $\alpha(t)$ varies along time t. Here we take the exponential function by assuming the correlation between $\alpha(t+\tau)$ and $\alpha(t)$ to be at least subgaussian. If in a given video, the i-th action signal is a high frequency component, then its coefficient $\boldsymbol{\alpha}_i(t)$ will behave like a random number for time skip τ. Therefore, we would expect that the correlation between $\boldsymbol{\alpha}_i(t)$ and $\boldsymbol{\alpha}_i(t + \tau)$ is close to 0. Or if the action signal is a low frequency component, the correlation of $\boldsymbol{\alpha}_i(t)$ and $\boldsymbol{\alpha}_i(t + \tau)$ hence the correlation indicator γ should be

close to 1. For the sake of simplicity, we rearrange latent action signal $\bar{X}$ by their frequency to have $\gamma_1 \leq \gamma_2 \leq \cdots \leq \gamma_k$.

In a learning problem, we hope the feature matrix $\mathcal{F}[X(t), \tau]$ is well-conditioned. Given feature matrix $\mathcal{F}[X(t), \tau]$, we can recover $\bar{X}$ by various methods, such as subspace clustering. The sampling complexity of any recovery algorithm depends on the condition number of P. Clearly when P is ill–conditioned, we require a large number of training examples to estimate $\bar{X}$. The learnability of $\mathcal{F}[X(t), \tau]$ depends on its condition number [18] which in return depends on P again. In the following, we will prove that for a fixed time skip τ, P is not necessarily well conditioned. Therefore the learnability of $\mathcal{F}[X(t), \tau]$ is suboptimal. The intuition behind our proof is that when an action signal has a large γ, then a small time skip τ will make the coefficient of that signal close to zero. Therefore, P is ill-conditioned. Formally, we have the following theorem to bound the condition number $\beta(PP^{\mathrm{T}})$ of PP^{T}(see the proof in Appendix A).

Theorem 1 *Given a fixed time skip τ, with probability at least $1 - \delta$, the condition number $\beta(PP^{\mathrm{T}})$ is bounded by*

$$\beta(PP^{\mathrm{T}}) \leq \frac{(1+c)\exp(-\gamma_1/\tau) + \Delta_\tau}{\exp(-\gamma_k/\tau) - \Delta_\tau} \tag{4.8}$$

$$\beta(PP^{\mathrm{T}}) \geq \frac{(1+c)\exp(-\gamma_1/\tau) - \Delta_\tau}{\exp(-\gamma_k/\tau) + \Delta_\tau} \ . \tag{4.9}$$

where

$$\Delta_\tau = 2\sqrt{k\frac{1}{T}(1+c)\log(2k/\delta)} \tag{4.10}$$

provided the number of feature points

$$T \geq \frac{1}{9(1+c)} k \log(2k/\delta) \ . \tag{4.11}$$

Theorem 1 shows that when the number of features T is large enough, the condition number $\beta(PP^{\mathrm{T}})$ is a random number concentrated around its expectation $(1+c)\frac{\exp(-\gamma_1/\tau)}{\exp(-\gamma_k/\tau)}$. Since $\gamma_1 \ll \gamma_k$, the numerator is much greater than the denominator when τ is fixed. Since our proof is based on Bernstein's Inequality, the upper bound is tight. This forces $\beta(PP^{\mathrm{T}})$ to be a relatively large value. More specifically, the following corollary shows that when γ_k is linear to γ_1, $\beta(PP^{\mathrm{T}})$ is exponentially large in expectation.

Corollary 1 *When $\gamma_k \geq (M+1)\gamma_1$,*

$$\mathrm{E}\{\beta(PP^{\mathrm{T}})\} \geq (1+c)[\exp(\frac{\gamma_1}{\tau})]^M \geq (1+c)(1+\frac{\gamma_1}{\tau})^M \ . \tag{4.12}$$

Corollary 1 shows that when the actions in the video span across a vast range of dynamic (large M), the feature extractor with single τ tends to have ill-conditioned feature matrices. A naive solution to this problem is to increase τ to reduce the condition number in expection. However, this will increase the variance Δ_τ of $\beta(PP^T)$ because of a smaller number of features.

In practice, a large τ also increases the difficulty in optical flow calculation and tracking. Hence, as will also be observed in our experiments, choosing a good τ can be fairly difficult. Intuitively speaking, selecting τ is a trade-off between feature bias and variance. A feature extractor with a large τ covers a long range of action signals but with less feature points hence generates features with small bias but large variance. Similarly, a feature extractor with a small τ will generate feature with large bias but small variance.

4.4.2 Condition number of P under multiple τ

From Theorem 1, to make PP^{T} well-conditioned, we need τ as large as possible. However, when τ is too large, we cannot sample enough high quality feature points, the variance in PP^{T} will increase. To address this problem, we propose to use MIFS, which incrementally enlarges the time skip τ , then stacks all features under various τ_i to form a feature matrix. Hopefully, by increasing τ, we improve the condition number $\beta(PP^{\mathrm{T}})$ and by stacking, we sample enough features to reduce the variance.

Assuming we have features extracted from $\{\tau, 2\tau, \cdots m\tau\}$. For the $i\tau$ skip, the number of extracted features is $T_i = \lfloor 1/(i\tau) \rfloor$. The following theorem bounds the condition number of MIFS (see the proof in supplementary materials).

Theorem 2 *With probability at least $1-\delta$, the condition number of PP^{T} in the MIFS is bounded by*

$$\beta(PP^{\mathrm{T}}) \leq \frac{\sum_i \frac{T_i}{T} 2(1+c)\exp(-\gamma_1/\tau_i) + \Delta_\tau}{\sum_i \frac{T_i}{T} 2\exp(-\gamma_k/\tau_i) - \Delta_\tau} \,. \tag{4.13}$$

where

$$\Delta_\tau \leq 2\sqrt{k\frac{1}{\sum_i T_i}(1+c)\log(2k/\delta)} \,. \tag{4.14}$$

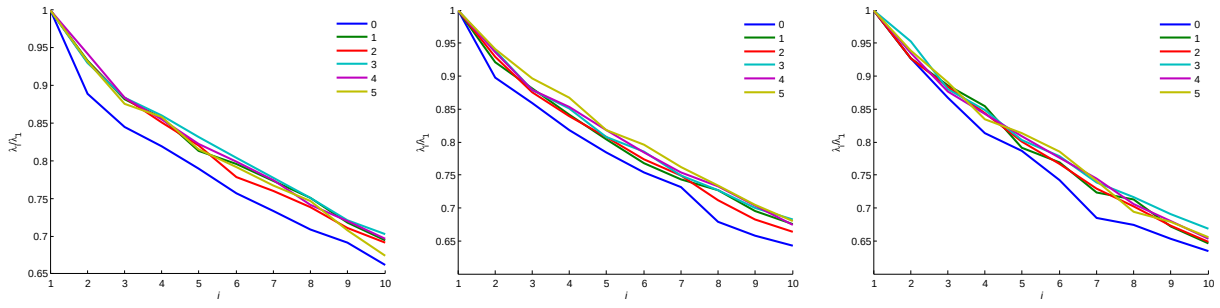
Theorem 2 shows that, in the MIFS, the expected condition number $\beta(PP^{\mathrm{T}})$ is roughly the weighted average of condition numbers under various τ_i. Since $\tau_{i+1} > \tau_i$, the condition number under τ_{i+1} is smaller than the one under τ_i. Therefore, the condition number is reduced as we expected. What's nicer is that the variance component Δ_τ is actually on order of $1/\sqrt{\sum_i T_i}$, which is also much smaller than a single τ scenario. In summary, we prove:

> The MIFS representation improves the learnability of differential feature extractors because it reduces the expectation and variance of condition number $\beta(PP^{\mathrm{T}})$ simultaneously.

4.5 Experiments

We examine our hypothesis and the proposed MIFS representation on two tasks: action recognition and MED. The experimental results show that MIFS representations outperform conventional original-scale representations on seven challenging real-world datasets.

IDT with Fisher Vector encoding [126] represents the current state-of-the-arts for most real-world action recognition datasets. Therefore, we use it as a foundation to evaluate our method.



(a) HMDB51 (b) Hollywood2 (c) UCF101

Figure 4.3: The decaying trend of singular values of feature matrices for HMDB51, Hollywood and UCF101 Datasets. 0 to 5 indicate the MIFS level and i indicates the ith singular value. From all three datasets, we can see that MIFS representations do have a slower singular value decaying trend compared to conventional representations (blue lines).

4.5.1 Action Recognition

Experimental Settings

The goal of this task is to recognize human actions in short clips of videos. Five action recognition datasets including HMDB51, UCF101, UCF50, Hollywood2 and Olympic Sports are used in this evaluation.

IDT features are extracted using 15 frame tracking, camera motion stabilization and Root-SIFT normalization and described by Trajectory, HOG, HOF, MBHx and MBHy descriptors [126]. We use PCA to reduce the dimensionality of these descriptors by a factor of two. After reduction, we augmented the descriptors with three dimensional normalized location information. The only difference between MIFS and other conventional methods is that instead of using feature points extracted from one time scale, we extract and stack all the raw feature points from different scales together before encoding. For Fisher Vector encoding, we map the raw descriptors into a Gaussian Mixture Model with 256 Gaussians trained from a set of randomly sampled 256000 data points. Power and L2 normalization are also used after concatenating different types of descriptors into a video based representation. For classification, we use a linear SVM classifier with a fixed $C = 100$ as recommended by Wang *et al*. [126] and the one-versus-all approach is used for multi-class classification scenario.

Results

We demonstrate that the conditional number $\beta(PP^{\mathrm{T}})$ is improved by MIFS. However, it would not be meaningful to compute $\beta(PP^{\mathrm{T}})$ directly because we have noise $\boldsymbol{\epsilon}$ in $\mathcal{F}[X(t), \tau_i]$ and the smallest singular value λ_{min} is in noise space . A workaround is to examine the decaying speed of singular values of the feature matrix. The singular values are normalized by dividing the maximum singlular value $\lambda_{\max}$. We only plot the top 10 singular values, since the subspace spanned by the small singular values is noise space. Clearly, when MIFS improves the learnability, we should get a slower decaying curve of the top k singular values. Shown in Figure 4.3 are the

	HMDB51 (MAcc%)		Hollywood2 (MAP%)		UCF101 (MAcc%)		UCF50 (MAcc%)		Olympics Sports (MAP%)	
L	single-scale	MIFS	single-scale	MIFS	single-scale	MIFS	single-scale	MIFS	single-scale	MIFS
0	62.1		67.0		87.3		93.0		89.8	
1	63.1	63.8	66.4	67.5	87.3	88.1	93.3	94.0	89.4	**92.9**
2	54.3	64.4	62.5	67.9	85.5	88.8	92.2	94.1	88.1	91.7
3	43.8	65.1	60.5	**68.0**	81.3	**89.1**	89.7	**94.4**	85.3	91.4
4	24.1	**65.4**	58.1	67.4	74.6	**89.1**	84.3	**94.4**	85.0	90.3
5	15.9	**65.4**	54.4	67.1	66.7	89.0	76.7	94.3	82.3	91.3

Table 4.1: Comparison of different scale levels for MIFS.

trends of $\frac{\lambda_i}{\lambda_{max}}$ on the first three datasets: HMDB51, Hollywood2 and UCF101. On all three datasets, the singular values of MIFS decrease slower than the conventional one (0). It is also interesting to see that by having one or two additional levels, we have already exploited most of the potential improvement.

We further examine how performance changes with respect to the MIFS level, as shown in Table 4.1 (detailed results can be found in Appendix B). First, let us compare the performance of L=0 to the standard location-insentative feature representation. Our performance on HMDB51, Hollywood2, UCF101 and UCF50 datasets are 62.1% MAcc, 67.0% MAP, 87.3% MAcc and 93.0% respectively. These numbers are higher than Wang & Schmid [126]'s results, which are 57.2%, 64.3%, 85.9% and 91.2%, respectively. This improvement is largely because of our location sensitive feature representation. Next, let us check the behavior of MIFS. For completeness, we list both single-scale and stacking performance. For single-scale performance, we observe that for HMDB51, its performance increases from 62.1% to 63.1% and then decreases rapidly. Similar patterns can be seen in other datasets except some of them do not increase at $L = 1$. These results are consistent with our observation that different action types need different scale ranges. They also substantiate our proof that selecting time interval τ is a trade-off between the feature bias and its variance. Now let us compare MIFS with a single-scale representation. We observe that for MIFS representations, although there is still a bias and variance trade-off as in single-scale representations for different levels, they all perform better than single-scale representation and the levels at which performance starts decreasing occur later than those in the single-scale representations. We also observe that for MIFS representations, most of the performance improvement comes from $L = 1$ and $L = 2$, which supports what we observed in Figure 4.3 that, in practice, having one or two more scales is enough to recover most of the lost information due to the differential operations. Higher scale features become less reliable due to the increasing difficulty in optical flow estimation and tracking. It is also interesting to observe that HMDB51 enjoys a higher performance improvement from MIFS than the other four datasets have. We believe that the main reason is that HMDB dataset is a mixture of videos from two sources: amateur and movies, which results in larger action velocity range than pure movie videos or pure amateur videos in the other two datasets.

In Table 4.2, we compare MIFS at $L = 3$, which performs well across all three action datasets, with the algorithm from which we build on our approaches. We can see that MIFS consistently improves IDT by a large margin except for Olympic Sports dataset.

HMDB51 (MAcc. %)		Hollywood2 (MAP %)		UCF101(MAcc. %)		UCF50 (MAcc. %)		Olympics Sports (MAP %)	
Wang *et al.*	57.2	Wang *et al.*	64.3	Wang *et al.*	85.9	Wang & Schmid	91.2	Wang & Schmid	91.1
MIFS (L=3)	**65.1**	MIFS (L = 3)	**68.0**	MIFS (L = 3)	**89.1**	MIFS (L=3)	**94.4**	MIFS (L = 3)	**91.4**

Table 4.2: Comparison between our results to the baseline method.

	MEDTEST13		MEDTEST14	
	EK100	EK10	EK100	EK10
Baseline	34.2	17.7	27.3	12.7
MIFS (L=3)	**36.3**	**19.3**	**29.0**	**14.9**

Table 4.3: Performance Comparison on the MED task.

4.5.2 Multimedia Event Detection

Experimental Settings

Both MEDTEST2013 and MEDTEST2014 are used in this evaluation. A similar setting as discussed in section 4.5.1 is applied except we use five-fold cross-validation to choose the penalty parameter C for linear SVM. For each classifier, C is chosen among $10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}, 10^{3}$. We only test MIFS with $L = 3$ as recommended in section 4.5.1 because extracting IDT features from such a large datasets itself is very time consuming. It took us 4 days to generate representations for both MEDTEST2013, MEDTEST2014 using a cluster with more than 500 Intel E565+ series processors. We use MAP as our evaluation criteria.

Results

Table 4.3 lists the overall MAP (detailed results can be found in Appendix B). The baseline method is a conventional single-scale representation with $L = 0$. From Table 4.3, we can see that for both MEDTEST2013 and MEDTEST2014, MIFS representations consistently improve over the original-scale representation by about 2% in both EK100 and EK10. It is worth emphasizing that MED is such a challenging task that 2% of absolute performance improvement is quite significant.

4.5.3 Computational Complexity

Level 0 of a MIFS representation has the same cost as other single pass methods, e.g., Wang & Schmid. [126]. For each level l, the cost becomes $1/l$ of the level 0. So with a MIFS up to level 2, the computational cost will be less than twice the cost of a single pass through the original video, yet it can significantly improve the single-pass methods.

4.6 Conclusion

In this chapter, we describe a new video preprocessing method called MIFS. MIFS enhances the learnability of motion representations for MED. It stacks features extracted using a family

of differential filters parameterized with multiple time skips and achieves shift-invariance in the frequency space. In contrast to Gaussian Pyramid, MIFS generates features at all scales and tends to cover a longer range of motion signals. Theoretical results show that MIFS improves the learnability of motion representation exponentially. Extensive experiments on seven real-world datasets show that MIFS exceeds state-of-the-art methods. A potential improvement is to determine the appropriate level for different motion types. Additionally, we would like to improve the quality of optical flow calculation and tracking at coarse scales.

Chapter 5

Feature: Learning based Video Features

5.1 Introduction

In this chapter, we will discuss our works on using CNNs to learn video representations. Despite much effort and progress, deep convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have yet to achieve the same success on video classification as they have on image classification. This can in large part be attributed to the following two differences between image and videos, differences which are key to deep-learning based approaches. First, videos are much larger in size and thus it becomes memory prohibitive to train and apply CNNs/RNNs at the video level. Second, it is very difficult to construct the large labeled video datasets required for training deep networks. Recent approaches [106, 128, 129] circumvent these problems by learning on sampled frames or very short video clips (temporally local inputs[1]) with video-level (global) labels.

However, video-level label information can be incomplete or even missing at frame/clip-level. This information mismatch leads to the problem of false label assignment. In other words, the imprecise frame/clip-level labels populated from video labels are too noisy to guide precise mapping from local video snippets to labels. To deal with this problem, a common practice is to sample multiple frames/clips from a video at testing time and aggregate the prediction scores of these sampled frames/clips to get the final results for that video. However, simply averaging the prediction scores, without another level of mapping, is not enough to recover the damages brought by false label assignment.

We instead compensate for the noisy labels by treating the deep networks trained on local inputs as feature extractors as shown in Figure 5.1. Local features extracted using the pre-trained networks are aggregated into global video-level features and another mapping function (e.g., a shallow network) is learned using the same dataset to assign video-level labels.

Our method is therefore related to the fine-tuning practices that are popular in image classification. The major difference is that we train our feature extraction networks with local data and with very noisy labels due to the false label assignment. We thus rely heavily on the shallow network to compensate for the suboptimal local feature learning.

Our method is also similar to the common practice of using networks pre-trained on the

[1] local from here on will mean temporally local

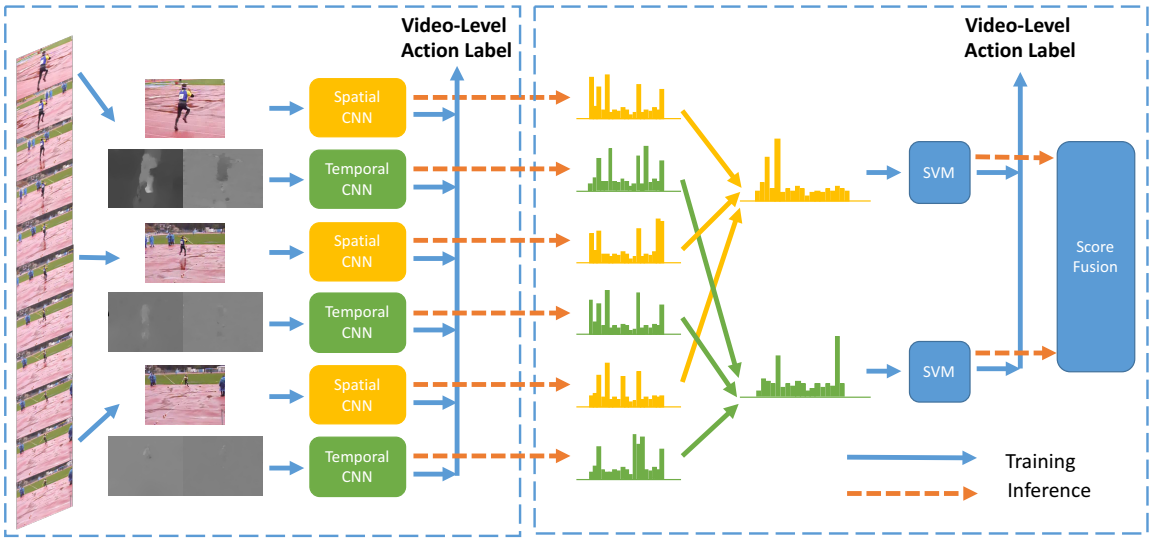


Figure 5.1: Overview of our proposed framework which consists of two stages. First (Left): A temporal segment network [129] is trained using local video snippets with the video-level labels. These networks are used as local feature extractors. Second (Right): The local features are aggregated to form a global feature which is then mapped to the video-level labels. Our results show that this two stage process compensates for the noisy snippet labels that result from propagating the video-level labels.

ImageNet image classification task to extract frame-level (local) features for video classification [65, 132]. The main difference is that our local feature extractors (deep networks) are trained on the target dataset. Therefore, the features extracted from deep networks are in-domain. We don't have the domain gap problem as we have in using the ImageNet trained deep networks.

We name our new class of local video features Deep lOcal Video Feature (DOVF).

In summary, DOVF is a class of local video features that are extracted from deep neural networks trained on local video clips using global video labels. In this paper, we investigate the following design choices related to DOVF:

- From which layer(s) of the CNNs should the local features be extracted? Without further investigation, the only guidance we have is that we should avoid the probability layer as it is likely to severely overfit the noisy training data and thus result in a distribution that varies greatly between the training and test sets.
- What is the best way to aggregate the local features into video-level global features? We consider a number of feature aggregation methods such as mean pooling, max pooling, Fisher Vectors (FV) encoding, etc..
- How densely should the local features be extracted? Sparse temporal sampling would be preferred from a efficiency standpoint.
- How complementary is DOVF to traditional local features such as IDT [126]? The more

complementary they are, the more opportunity there is for improvement by applying techniques that have been developed for traditional local features.

The remainder of this chapter is organized as follows. We first provide some background on video features with an emphasis on recent work on learning with deep neural networks. We then describe the experimental settings we use to evaluate our framework on the HMDB51 and UCF101 datasets. We conclude with a discussion including potential improvements.

5.2 Related Works

New video representations have typically been the main source of breakthroughs for video classification.

In traditional video representations, trajectory based approaches [53, 126], especially the Dense Trajectory (DT) and its improved form, IDT [125, 126], are the basis of current state-of-the-art hand-crafted algorithms. These trajectory-based methods are designed to address the shortcomings of image-extended video features. Their superior performance validates the need for motion feature representations. Many studies have tried to improve upon IDT due to its success and popularity. Peng *et al*. [93] enhanced the performance of IDT by increasing the codebook sizes and fusing multiple coding methods. Sapienza *et al*. [101] explored ways to subsample and generate vocabularies for DT features. Hoai and Zisserman [36] achieved superior performance on several action recognition datasets by applying data augmentation, modeling the score distributions over video subsequences, and capturing the relationships among action classes. Fernando *et al*. [30] modeled the evolution of appearance in video and achieved state-of-the-art results on the Hollywood2 dataset. [67] proposed to extract features from videos at multiple playback speeds to achieve speed invariance. However, these traditional, hand-crafted methods have recently started to become overshadowed by the rise of deep learning using neural networks.

Motivated by the success of CNNs, researchers have invested significant effort towards developing CNN equivalents for learning video features. Several accomplishments have been reported from using CNNs for action recognition in videos [121, 130, 140, 142]. Karpathy *et al*. [54] trained deep CNNs using one million weakly labeled YouTube videos and reported moderate success using the network as a feature extractor. Simonyan and Zisserman [106] demonstrated a result competitive with IDT [126] by training deep CNNs using both sampled frames and stacked optical flow. Tran *et al*. [117] explored 3D CNNs to simultaneously learn spatiotemporal features without pre-computing optical flow. This allows them to achieve competitive performance at much faster rates. Wang *et al*. [127–129] provide insightful analyses on improving two-stream frameworks such as pre-training two-stream CNNs, using smaller learning rates, using deeper networks, etc. These improvements result in a CNN-based approach that finally outperforms IDT [126] by a large margin on the UCF101 dataset. These approaches, however, all rely on shot-clip predictions to determine the final video labels and do not use global features.

Two concurrent works [24, 97] on global features for action recognition have recently been posted on arXiv. Both propose new feature aggregation methods to pool the local neural network features to form global video features. Diba *et al*. [24] propose a bilinear model to pool the outputs of the last convolutional layers of pre-trained networks and achieve state-of-the-art results

ID	VGG16			Inception-BN		
	Name	Dimension	Type	Name	Dimension	Type
L-1	fc8	101	FC	fc-action	101	FC
L-2	fc7	4096	FC	global_pool	1024	Conv
L-3	fc6	4096	FC	inception_5b	50176	Conv
L-4	pool5	25088	Conv	inception_5a	50176	Conv
L-5	conv5_3	100352	Conv	inception_4e	51744	Conv

Table 5.1: Names, dimensions and types of the layers that we consider in the VGG16 and Inception-BN networks for local feature extraction.

Layers	Spatial CNNs (%)		Temporal CNNs (%)		Two-stream (%)	
	VGG-16	Inception-BN	VGG16	Inception-BN	VGG-16	Inception-BN
L-1	77.8	83.9	82.6	83.7	89.6	91.7
L-2	79.5	88.3	85.1	88.8	91.4	94.2
L-3	80.1	88.3	86.6	88.7	91.8	93.9
L-4	83.7	85.6	86.5	85.3	92.4	91.4
L-5	83.5	83.6	87.0	83.6	92.3	89.8
TSN	79.8	85.7	85.7	87.9	90.9	93.5

Table 5.2: Layer-wise comparison of VGG-16 and Inception-BN networks on the split 1 of UCF101. The values are the overall video-level classification accuracy of our complete framework.

on both the HMDB51 and UCF101 datasets. Qiu *et al*. [97] propose a new quantization method similar to FV and achieve comparable performance to [24]. However, neither work provides detailed analysis of the local neural network features that are used. In this paper, we perform an extensive analysis and show that a simple max pooling can achieve similar or better results compared to much more complex feature aggregation methods such as those in [24, 97].

5.3 Experimental Settings

5.4 Methodology

In this section, we first review temporal segment networks [129], the architecture upon which our approach is built. We next describe our Deep lOcal Video Features (DOVF), methods for aggregating them to form global features, and the mapping of the global features to video-level labels. Finally, we provide our experimental settings.

5.4.1 Temporal Segment Networks

With the goal of capturing long-range temporal structure for improved action recognition, Wang *et al*. propose temporal segment networks (TSN) [129] with a sparse sampling strategy. This allows an entire video to be analyzed with reasonable computational costs. TSN first divides a video evenly into three segments and one short snippet is randomly selected from each segment. Two-stream networks are then applied to the short snippets to obtain the initial action class prediction scores. TSN finally uses a segmental consensus function to combine the outputs from multiple short snippets to predict the action class probabilities for the video as a whole.

Wang *et al*. [129] show TSN achieves state-of-the-art results on the popular action recognition benchmarks UCF101 and HMDB51. These results demonstrate the importance of capturing long-range temporal information for video analysis. However, the training of the local snippet classifiers is performed using the video-level labels. As noted earlier, these are likely to be noisy labels and will thus limit the accuracy of the snippet-level classification.

We there propose instead to use the snippet-level analysis for local feature extraction and add a second stage which maps the aggregated features to the video-level labels. The combination of DOVF and a second classification stage compensates for the suboptimal snippet-level classification that results from the noisy training dataset.

5.4.2 Deep local video feature (DOVF)

Instead of performing action recognition in a single step like [24, 129], our framework consists of two stages. In the first stage, deep networks (*e.g.* , TSN) that have been trained with video-level labels to perform snipped-level classification are used as local feature extractors. In the second stage, the local features are aggregated to form global features and another classifier which has also been trained using the video-level labels performs the video-level classification.

The training of our classification framework proceeds as follows where each video V in the training set has ground truth action label p. In the first stage, V is evenly divided into N segments, $v_1, v_2, \cdots, v_N$ and one short snippet is randomly selected from each segment, $s_1, s_2, \cdots, s_N$. These snippets are assigned the video-level labels and the snippets from all the training videos are used to train a two-stream CNNs (single RGB video frame and stack of consecutive optical flow images). The details on training the two-stream CNNs can be found in [128, 129]. Once trained, the network is used to extract local features, $f_1, f_2, \cdots, f_N$ from a video.

In the second stage, the local features are aggregated into a global feature f_G,

$$f_G = G(f_1, f_2, \cdots, f_N) \tag{5.1}$$

where G denotes the aggregation function. We explore different aggregation functions in Section 5.5.2. We then learn a classifier that maps the global feature f_G to the video label p:

$$p = M(f_G). \tag{5.2}$$

Once trained, the framework can be used to predict the label of a video. Figure 5.1 contains an overview of the framework.

Layers	Spatial CNNs (%)		Temporal CNNs (%)		Two-stream (%)	
	HMDB51	UCF101	HMDB51	UCF101	HMDB51	UCF101
$Mean$	56.0	87.5	63.7	88.3	71.1	93.8
$Mean\_Std$	58.1	88.1	65.2	88.5	72.0	94.2
Max	57.7	88.3	64.8	88.8	72.5	94.2
BoW	36.9	71.9	47.9	80.0	53.4	85.3
FV	39.1	69.8	55.6	81.3	58.5	83.8
$VLAD$	45.3	77.3	57.4	84.7	64.7	89.2

Table 5.3: Comparison of different local feature aggregation methods on split 1 of UCF101 and HMDB51.

5.4.3 Experimental settings

We compare two networks, VGG16 and Inception-BN, for the local feature extraction. (We use networks trained by Wang *et al.* [128, 129].) We further compare the outputs from the last five layers from each network as our features. Table 5.1 shows the layer names of each network and the correspondent feature dimensions. We divide the layers into two categories: fully-connected (FC) layers and convolution (Conv) layers (pooling layers are treated as Conv layers). FC layers have significantly more parameters and are thus more likely to overfit the training data than the Conv layers. As shown, VGG16 has three FC layers while Inception-BN only has one.

Following the scheme of [106, 129], we evenly sample 25 frames and flow clips for each video. For each frame/clip, we perform 10x data augmentation by cropping the 4 corners and center along with horizontal flipping. A single feature is computed for each frame/clip by averaging over the augmented data. This results in a set of 25 local features for each video. The dimensions of local features extracted from different network/layer combinations are shown in Table 5.1.

We compare a number of local feature aggregation methods ranging from simple mean and maximum pooling to more complex feature encoding methods such as Bag of words (BoW), Vector of Locally Aggregated Descriptors ($VLAD$) and Fisher Vector (FV) encoding. In order to incorporate global temporal information, we divide each video into three parts and perform the aggregation separately. That is, the first eight, middle nine and final eight of the 25 local features are separately aggregated and then concatenated to form the final global feature. This increases the final feature dimension by three. After concatenation, we perform a square root normalization and L2 normalization as in [67] on the global feature.

We use support vector machines (SVMs) to map (classify) the global features to video-level labels. We use a chi-square kernel and $C = 100$ as in [67] except for the FV and VLAD aggregated features where we use a linear kernel as suggested in [123]. Note that while we use SVMs to predict the video action labels, other mappings/classifiers, such as a shallow neural network, could also be used.

The spatial-net and temporal-net prediction scores of the two-stream network are fused with weights 1 and 1.5, respectively, as in [129].

# of samples	Spatial CNNs (%)		Temporal CNNs (%)		Two-stream (%)	
	HMDB51	UCF101	HMDB51	UCF101	HMDB51	UCF101
3	52.5	85.6	54.9	82.4	64.6	91.6
9	56.1	87.4	62.2	87.7	70.9	93.5
15	56.9	88.2	64.4	88.5	72.3	93.8
21	57.1	88.1	64.8	88.6	71.8	94.1
25	57.7	88.3	64.8	88.8	72.5	94.2
Max	57.6	88.4	65.3	88.9	72.4	94.3

Table 5.4: Number of samples per video versus accuracy on split 1 of UCF101 and HMDB51.

5.5 Evaluation

In this section, we experimentally explore the design choices posed in the Introduction using the UCF101 and HMDB51 datasets.

UCF101 is composed of realistic action videos from YouTube. It contains $13,320$ video clips distributed among 101 action classes. HMDB51 includes $6,766$ video clips of 51 actions extracted from a wide range of sources, such as online videos and movies. UCF101 and HMDB51 both have a standard three split evaluation protocol. We report the average recognition accuracies over the three splits.

Our default configuration uses the outputs of the global_pool layer in the Inception-BN network as the local features due to this layer's low dimension (3072 dimensions with global information encoding). It also uses maximum pooling to aggregate the local features to form the global features.

5.5.1 From which layer(s) should the local features be extracted?

We conduct experiments using both VGG16 and Inception-BN to explore which layers are optimal for extracting the local features. The video-level action classification accuracies on split 1 of UCF101 using different layers are shown in Table 5.2.

Layer L-2 from Inception-BN and layer L-4 from VGG16 give the best performance. These are the final convolution layers in each network which suggests the following three reasons for their superior performance. First, the convolution layers have far fewer parameters compared to the fully connected layers and thus are less likely to overfit the training data that has false label assignment problem. Second, the fully connected layers do not preserve spatial information while the convolution layers do. Third, the later convolution layers encode more global (spatial) information than the earlier ones. We conclude that extracting the local features from the final convolution layers is the optimal choice. We believe this finding helps explain why several recent works [24, 97, 132] also choose the output of the final convolution layer for further processing.

Compared to the results of Wang *et al*. [129], from which we get the pre-trained networks, we can see that our approach do improve the performance on both spatial-net and temporal-net. However, the improvements from spatial networks are much larger. This larger improvement may be because that, in training local feature extractors, the inputs for spatial net are single

frames while the input for temporal net are video clips with 10 stacked frames. Smaller inputs lead to larger chance of false label assignment hence larger performance gap compared to our global feature approach.

Previous work [65, 132, 140] on using local features from networks pre-trained using the ImageNet dataset show that combining features from multiple layers can improve the overall performance significantly. We investigated combining features from multiple layers but found no improvement. This difference shows that fine-tuning brings some new characteristics to the local features.

In the remaining experiments, we use the output of the global_pool layer from the Inception-BN network as it achieves the best performance.

5.5.2 What is the optimal aggregation strategy?

We consider six aggregation methods on split 1 of the the UCF101 and HMDB51 datasets.

Given n local features, each of which has a dimension of d, the six different aggregation methods are as follows:

- $Mean$ computes the mean value of the n local features along each dimension.
- Max selects the maximum value along each dimension.
- $Mean\_Std$, inspired by Fisher Vector encoding, computes the mean and standard deviation along each dimension.
- BoW quantizes each of the n local features as one of k codewords using a codebook generated through k-means clustering.
- $VLAD$ is similar to BoW but encodes the distance between each of the n local features and the assigned codewords.
- FV models the distribution of the local features using a Gaussian mixture model (GMM) with k components and computes the mean and standard deviation of the weighted difference between the n local features and these k components.

For those feature aggregation methods that require clustering, we project each local feature into 256 dimensions using PCA and set the number of clusters as 256. This is similar to what suggested in [132] except we don't break the local features into multiple subfeatures.

As can be seen in Table 5.3, maximum pooling (Max) achieves the best overall performance (two-stream network results). This result is different from that of [65] where mean pooling ($Mean$) performs better than maximum pooling (Max). It is also interesting that $Mean\_std$ consistently performs better than $Mean$. The more complicated encoding methods, BoW, FV and $VLAD$, all perform much worse than simple pooling. We conjecture that extracting a larger number of local features for each video and dividing the features into lower dimension subfeatures as in [132] would likely improve the performance of the more complicated methods. This would, however, incur excessive computational cost and limit practical application.

We use maximum pooling in the remainder of the experiments.

	HMDB51	UCF101
IDT	57.2	85.9
Two-stream	59.4	88.0
MIFS	65.1	89.1
C3D	-	90.4
TDD	65.9	91.5
Action Transformations	62.0	92.4
LTC	67.2	92.7
Depth2Action	68.2	93.0
Key Volume Mining	63.3	93.1
Two-stream Fusion	69.2	93.5
TSN	68.5	94.0
Deep Quantization	-	95.2
TLE	71.1	95.6
DOVF (ours)	71.7	94.9
DOVF + MIFS (ours)	75.0	95.3

Table 5.5: Comparison with state-of-the-art. Mean classification accuracy on UCF101 and HMDB51 over three splits.

5.5.3 How densely should the local features be extracted?

We vary the number of local features extracted from each video between 3 and 25. We also experiment with using the maximum number (Max) by extracting features for every frame/clip (for optical flow, we use a sliding window with step size equal to 1). The videos in HMDB51 and UCF101 contain 92 and 185 frames on average, respectively.

The results in Table 5.4 show that, after a threshold of around 15, the number of sampled frames/clips does not have much of an effect on performance. Sampling 25 frames/clips achieves similar performance to using them all. This is consistent with the observations in [65] and is likely due to the high level of redundancy between frames. However, since the videos in UCF101 and HMDB51 are quite short and the datasets are small, these results should be taken with a bit of caution. Also, using attention model [137] to select frame should improve the overall performance. However, attention model greatly increases the model size, hence make it much more difficult to train.

5.5.4 Comparison with state-of-the-art

Table 5.5 compares our best performance with the state-of-the-art. We improve upon TSN [129], which forms the basis of our approach, by around 3% and 1% on HMDB51 and UCF101, respectively. Our results are also much better than both traditional IDT based methods [67] and the original Two-stream CNNs [106]. In comparison to TLE [24] and Deep Quantization [97], our maximum pooling achieves similar performance to their more complex bilinear models and FV-VAE framework. We also show the results of fusing our prediction scores with those from

MIFS [2] using late fusion. For HMDB51, the improvement from fusing with MIFS is very significant and is more than 3%. The improvement for UCF101 is smaller as the accuracy is already saturated.

5.6 Conclusion

In this chapter, we propose an effective method for deriving global video features from local features extracted using CNNs. We study a set of design choices such as which layer(s) to extract the features from, how to aggregate them and how densely to sample them. Based on a set of experiments on the UCF101 and HMDB51 datasets, we conclude that 1) the local features should be extracted from the final convolution layer; 2) maximum pooling works better than other feature aggregation methods including those which need further encoding; and 3) a sparse sampling of around 15 frames/clips per video is sufficient. While we propose plausible explanations for these conclusions, further investigation into DOVF is warranted. Also, the current two-stage approach only corrects the mistakes after it happens, we believe that a better way would be directly mapping a whole video into the global label, or so called end-to-end learning. Our future work will focus on these directions.

[2]We download the the prediction scores of MIFS from here

Chapter 6

Feature: Convolutional Independent Subspace Analysis for Trajectories (ConvISA)

6.1 Introduction

In this chapter, we present an unsupervised local descriptors learning methods. Despite its superior performance compared to handcrafted methods, we did not use it for MED because of the high cost in processing local features. We present it here in the hope that it can provide lessons for future unsupervised global feature learning development. Also, it has the theoretical value of showing the connection between generating features using traditional handcrafted methods and CNN-based methods.

Recent progress on the problem of action recognition mainly comes from the improvements of features, which can be categorized into two broad classes: 1) more traditional hand-crafted local features [124, 125] and their corresponding BoW encoding methods [94]; and 2) learning based features that are mainly inspired by the success of CNNs for image recognition [54, 58, 106] and of RNNs for speech recognition [34, 35, 87]. In this chapter, we design algorithms to combine the merits of both methodologies.

Trajectory based features, especially IDT [126], are the state-of-the-art hand-crafted features that have dominated action recognition in videos over the last few years. Compared to other hand-crafted motion features, IDT performs better in that it models long-term motion information and has a MBH descriptor that is robust to camera motion. This long-term motion information modeling, as shown in [54, 106], is very hard to learn in a CNN framework. Despite its superiority, IDT, somewhat surprisingly, relies on simple hand-crafted local descriptors such as HOG and HOF [81]. In contrast, for image and speech recognition [58, 87], data-driven approaches have consistently demonstrated their superiority and have been gradually replacing the traditional hand-crafted methods.

These revolutionary changes are largely are largely enabled by the easy access of neural network algorithms, large scale labelled data, and powerful parallel machines. Learning video features for action recognition, however, has proven to be quite a challenge due to its intrinsic

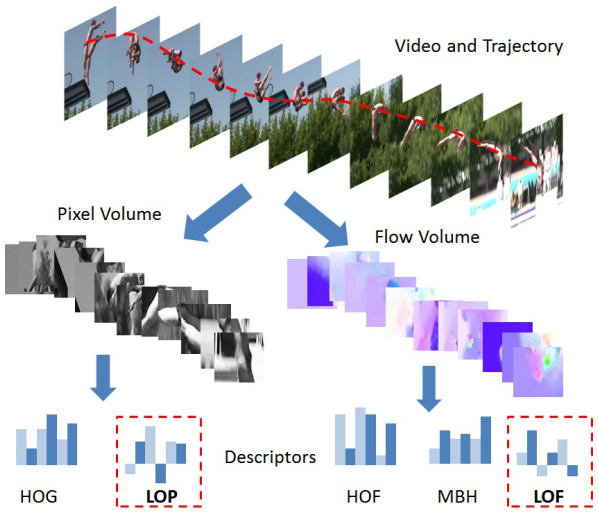


Figure 6.1: Illustration of our novel local video descriptors. LOP and LOF describe gray pixel and optical flow volumes, respectively. They resemble HOG/HOF/MBH in a data-driven learning framework.

high dimensionality, the lack of training data, and the difficulty in processing large-scale video data [54, 87, 106]. With limited training data and computational power, the learned features are generally not discriminative enough and perform worse than IDT, especially among datasets that have few training instances. Recent approaches [87, 106] circumvent these problems by learning from sampled frames or very short video clips, as well as using weakly labelled data. However, video-level label information can be incomplete or even missing at the frame/clip-level and leads to false label assignment, which can be even worse for weakly labelled data [54]. In other words, the imprecise frame/clip-level labels populated from video labels are usually too noisy for learning powerful models. With better labelled data, neural network algorithms can give superior results. Unfortunately, accurately labelled video data is very expensive to obtain.

Though we see the value in developing fully automatically learned global video features using labeled training data, in this chapter we propose to revisit the traditional local feature pipeline and unsupervised feature learning methods. By doing so, we hope to connect both data-independent and data-driven approaches and combine their strengths. Inspired by the two-stream CNNs [106] and ConvISA [72], we introduce a two-stream ISA-IDT to learn both visual appearance and motion information in an unsupervised manner. As shown in Figure 6.1, instead of learning from frames or short video clips, we learn from much smaller primitives – video volumes that follow the trajectories detected by IDT. The learned descriptors, called LOP (Learned descriptors of Pixel) and LOF (Learned descriptors of optical Flow), improve the best performing hand-crafted descriptors by using an unsupervised data-driven method. Our proposed architecture has several attractive properties:

- Compared to full video learning, small video volumes lie in a much lower dimensional space, hence they are computationally efficient to learn and apply.
- Unsupervised learning avoids the costly work of collecting labelled data and the false label assignment problem among current supervised video learning settings.

	Feature Training Time	Need Label	HMDB51	UCF101
Ours	$\sim$2 hours / 1 CPU	No	**61.5%**	**88.3%**
IDT	$\sim$ 1 hour / 1 CPU	No	57.2 %	85.9%
Two-stream CNNs	$\sim$ 1 day / 4 GPUs	Yes	59.4%	88.0%

Table 6.1: Performance comparison of our approach with IDT and two-stream CNNs.

- Through learning from video volumes defined by IDT, the resulting descriptors can be seamlessly combined with hand-crafted descriptors and boost the overall performance.
- By following the traditional local feature pipeline, we can easily utilize techniques developed for traditional local descriptors to improve our data-driven descriptors.

Although the idea of unsupervised video feature learning sounds appealing, it is, in fact, a very challenging problem. It introduces several novel problems that have not been sufficiently studied in the literature. The first one, of course, is the challenge of achieving high accuracy. For our algorithm to be useful, it needs to at least as good as IDT. This is by no means easy. For example, after years of research efforts, in unsupervised image feature learning, SIFT was still the best feature in PASCAL VOC challenges 2012 ([28]). The second challenge, which is unique to video feature learning, is how to learn to describe optical flow data in an unsupervised way. Research in the past [27, 69, 106, 126] show that the optical flow feature is an essential part of motion representation. To the best of our knowledge, we are the first to deal with unsupervised optical flow feature learning.

Before revealing how we address the above mentioned challenges, let us first show that our algorithm indeed outperforms IDT. We conduct experiments on HMDB51 and UCF101, as in [106]. Table 6.1 compares the model training times and accuracy of our method to IDT and two-stream CNN ([106]), a state-of-the-art CNN approach. For both IDT and two-steam CNN, several improvements have been proposed since they were first introduced, but we compare results from the original papers as most of the improvements can also be applied to our method. Later in this thesis, we will have more complete comparisons to the state-of-the-art. In Table 6.1, we show that the training time of our approach is several orders of magnitude shorter than two-stream CNNs. Two-stream CNNs need about 1 day to train a model on 4 Titan-X GPUs while our method only needs about 2 hours on 1 CPU. IDT feature training only needs around 1 hour on 1 CPU because the only part that requires learning is the codebook training. In terms of accuracy, our method outperforms two-stream CNNs on HMDB51 and has similar results on UCF101 despite the fact that it was trained on less data and does not need any labels to train the feature extraction module. Our results are also significantly better than the results of IDT.

In the remainder of this chapter, we first provide more background information about video features. We then describe the relationship between handcrafted features and CNN-based features in detail, followed by the descriptions of our two-stream ISA-IDT algorithm. After that, we conduct experiments and show more detailed comparisons of our method to the state-of-the-art methods. Further discussions including potential improvements are provided at the end.

6.2 Related Work

Features and encoding methods are the major sources of breakthroughs in conventional video representations. Among them, trajectory based approaches [53, 126], especially the IDT [125, 126], are the basis of current state-of-the-art hand-crafted algorithms. These trajectory-based methods are designed to address the flaws of image-extended video features. Their superior performance validates the need for a unique representation of motion features.

There have been many studies attempting to improve IDT due to its popularity. Peng *et al*. [93] enhanced the performance of IDT by increasing codebook sizes and fusing multiple coding methods. Sapienza *et al*. [101] explored ways to sub-sample and generate vocabularies for DT features. Hoai & Zisserman [36] achieved state-of-the-art performance on several action recognition datasets by using three techniques including data augmentation, modeling score distribution over video subsequences, and capturing the relationship among action classes. Fernando *et al*. [30] modeled the evolution of appearance in the video and achieved state-of-the-art results on the Hollywood2 dataset. [67] proposed to extract features from videos with multiple playback speeds to achieve speed invariances. However, none of them dealt with the fact that IDT relies on very simple, hand-crafted descriptors. In contrast, many data-driven approaches have demonstrated their modeling power in image recognition [58] and are quickly replacing traditional hand-crafted methods.

Motivated by this success of CNNs in image recognition, researchers are working intensely towards developing CNN equivalents for learning video features. Several accomplishments have been reported from using CNNs for action recognition in videos [121, 130, 140]. Karpathy *et al*. [54] trained CNNs through one million weakly labelled YouTube videos and reported moderate success while using it as a feature extractor. Simonyan & Zisserman [106] demonstrated a result competitive to IDT [126] through training deep CNNs using both sampled frames and stacked optical flows. Wang *et al*. [127] use the outputs of two-stream CNNs as features to replace HOG and achieved state-of-the-art results on HMDB51 and UCF101 datasets. All of the above relied on a large amount of labels which are expensive to get and generally perform worse than hand-crafted features among small datasets.

There have been a limited number of studies regarding unsupervised methods for learning video features. Among them the Independent Component Analysis (ICA) [38] was the first approach to learn representations of videos in an unsupervised way. Le *et al*. [72] addressed the issue using stacked ConvISA. Srivastava *et al*. [111] applied unsupervised feature learning through long-short term memory. Since these methods rely purely on pixel data, they struggled to capture motion information and generally performed no better than state-of-the-art hand-crafted methods. Also, the network structures of these methods, because they are designed for pixel data, cannot be directly used in learning motion features.

There are also several attempts at connecting the traditional feature encoding pipeline to the neural network frameworks. Vladyslav *et al*. [114] studied the structural similarities between Fisher vectors and neural networks and proposed to jointly optimize Fisher vectors and the classifier. Richard and Gall [99] converted the BoW model into an equivalent recurrent neural network and trained the BoW model and classifier together. Both above approaches focus on the end-to-end training of CNNs and again require labels and significantly increase the model training time. Instead, we emphasize the convoltional-pooling structure of CNNs rather than

their training methods. Jarrett *et al*. [43] also pointed out the connection between handcrafted features and CNNs. However, they focus on image feature learning, which is inherently different from video feature learning. They also did not explicitly explain what linear and non-linear operators these handcrafted features have and how to map them into a CNN framework.

This study overcomes many limitations from previous works by designing and adapting unsupervised feature learning methods to video and optical flow volumes detected by IDT. Our new learning paradigm does not rely on any labels, hence can work well among small datasets. It is better at capturing motion information due to our enhanced approaches to model optical flow information, and can use feature enhancing techniques developed for hand-crafted descriptors, as illustrated by MIFS.

6.3 Improved Dense Trajectory

IDT improves DT feature [125] through explicitly estimating camera motions and removing trajectories generated by them. It relies on histogram-based descriptors, which are computed within space-time volumes aligned with a trajectory to encode the appearance and motion information. The size of the volume is $s \times s$ pixels and l frames long, which corresponds to the input size of stacked ISA. To embed structure information, the volume is subdivided into discrete spatio-temporal grids of size $s_\tau \times s_\tau \times l_\pi$. The default size of volume and grid for IDT are $s = 32$, $l = 15$, $s_\tau = 2$ and $l_\pi = 3$.

6.4 The Convolution-Pooling Architecture

In this section we first define the convolution-pooling structure and then compare IDT with CNN-based video features. We highlight their structural similarities by showing that they are both features generated by deep convolution-pooling cascade with two key elements: convolution and pooling layers.

We define a convolution-pooling cascade as any single, iterative or recursive implementation of the following sequence of operations:

$$c(x) = f(w \otimes x)$$

$$p(x) = g(c(x))$$

where $w \otimes x$ is a three-dimensional convolution of a *filter* w with the $N \times M \times T$ video blocks x and $f()$ is any component-wise operation that is often non-linear. $w \otimes x$, and as a consequence $c(x)$ also have size $N \times M \times T$. (In practice, convolution may result in size shrinking). $g()$ is a *pooling* function that results in a shrinking of the argument and operates on any $N \times M \times T$ input to generate a $J \times K \times L$ output $p(x)$, where $J \leq N$, $K \leq M$, and $L \leq M$.

6.4.1 Handcrafted video features

A typical handcrafted video feature extraction procedure is often composed of two stages of convolution and pooling. The first stage purely relies on handcrafted filters and generates de-

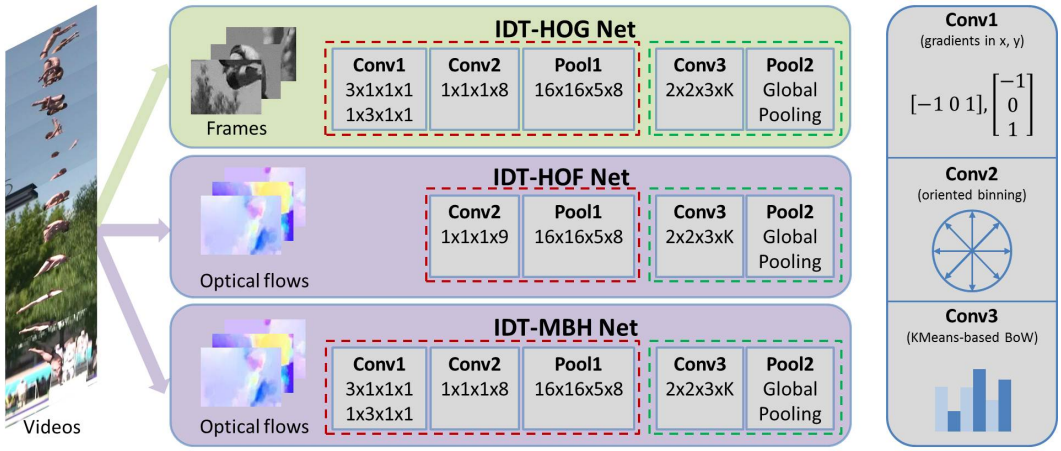


Figure 6.2: Schematic description of IDT as procedures of multiple convolution and pooling operations. Dashed red and green boxes represent the procedure of generating handcrafted descriptors and BoW encoding, respectively. In each operation, the first three numbers are the receptive field sizes in space and time (x, y, t) and the last number indicates the size of output channels.

scriptors from local data. The second one often uses filters learned from unsupervised methods to encode the descriptors generated from the first stage and pool them together to get global features. For example, shown in Figure 6.2 is a schematic description of three HOG-based IDT descriptors, each of which contains two stages of convolution and pooling (marked by dashed red and green boxes, respectively) including a total of three distinct convolution and two distinct pooling operations. Among them, **Conv1** uses two gradient filters as w and with:

$$f(x) = x.$$

Conv2 is an oriented soft binning, which can be approximated with w being the unit directional vectors and f being non-linear activation functions such as rectified linear unit ([58]):

$$f(x) = \max(x, 0).$$

Conv3 is BoW, which uses KMeans centroids as w and a softmax function ([99]) as f:

$$f_k(x) = \frac{\exp(x_k)}{\sum_j \exp(x_j)},$$

where k is the kth centroid. **Pool1** is a local sum pooling:

$$g_{x,y,t}(x) = \sum_{j,l,m \in [1,d]} x_{xj,yl,tm},$$

where d is the pool size in space and time and x, y, t are the space and time locations where $g()$ is applied. **Pool2** is a global sum pooling:

$$g(x) = \sum_{x,y,t} x_{x,y,t}.$$

Using above key operators, the **IDT-HOG Net** represents the procedure of generating a BoW encoded HOG feature from stacked frames. At the first stage, the stacked frames are convolved with two gradient filters followed by 8 oriented binning filters and one spatio-temporal sum pooling. During the second stage, the descriptors from the first stage are convolved with K binning filters learned using KMeans and pooled together afterwards. The **IDT-HOF Net** and **IDT-MBH Net** represent the procedures of generating KMeans encoded HOF and MBH features, respectively, from stacked optical flows. **IDT-MBH Net** is similar to the **IDT-HOG Net** except taking optical flows as inputs instead of pixels. **IDT-HOF Net** removes **Conv1** and using 9 oriented binning filters instead of 8. Note that although we use KMeans encoding as an example, other encoding methods such as Fisher Vector and VLAD have similar procedures ([99, 114]). For simplicity, we leave out the feature detection step, which can be viewed as another convolution with binary activation function. The main strength of this pipeline is that it is computationally efficient because of the layer-wise training and does not need labels to train the feature extraction module because it is meant to minimize reconstruction error. Its limitations lie in the first stage of the structure (dashed red box) in which it uses fixed parameters and structures for different sources of data.

6.4.2 Comparison with CNN-based video features

Needless to say CNNs employ convolution-pooling architecture. In CNNs, the non-linear activation is generally given by $f(x) = \tanh(x)$, $f(x) = (1 + e^{-x})^{-1}$ or $f(x) = \max(x, 0)$. The pooling functions are local average or maximum pooling, for example,

$$g_{x,y,t}(x) = \max_{j,l,m \in [1,d]} x_{xj,yl,tm},$$

where d is the pool size in space and time. The parameters of the model are the filters w. These are learned by minimizing a loss function, typically defined by

$$\min_{w} \sum_{i=1}^{n} ||y_i - h(w, x_i)||^2$$

where $h(w, \cdot)$ is the full convolution-pooling architecture that takes x as inputs. As can be noted above, the loss function requires the labels y of the training data.

Comparing the above two procedures, it is clear that their differences are not so much structural, but rather in how to get the network parameters. With this understanding, we try to answer the question of how to design a video feature learning algorithm that balances efficiency and effectiveness. At first, one might try performing end-to-end training on the network structure in Figure 6.2. However, this training again requires labels and large computational resources. In addition, results from [114] and [99] show that directly applying end-to-end learning on the traditional handcrafted pipelines would not bring large performance gains. So instead we keep the stage-wise unsupervised training to avoid the costly labeling and training. We address the limitations of handcrafted features by proposing a two-stream ISA-IDT to replace the handcrafted filters and enhance the proposed algorithms with two well motivated improvements.

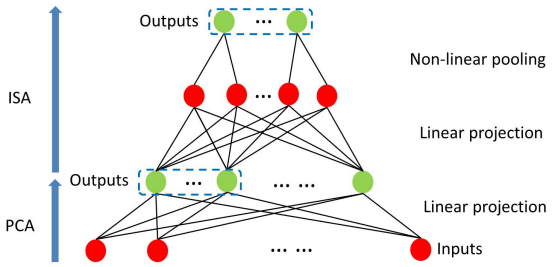


Figure 6.3: The neural network architecture of an ISA network with PCA preprocessing. The dashed blue boxes represent the outputs of our model.

6.5 Two-stream ISA-IDT

In this section, we will describe two-stream ISA-IDT in detail and its structures for both appearance (pixel) and motion (optical flow) stream learning ([106]).

As illustrated in Figure 6.3, an ISA ([39]) is a unsupervised feature learning method that can be described as a two-layered network within convolution-pooling architecture with: $f(x) = x^2$ and $g(x) = \sqrt{x}$. Specifically, let matrix $W \in \mathbb{R}^{m \times n}$ and matrix $V \in \mathbb{R}^{d \times m}$ denote the parameters of the first and second layers of ISA respectively. n is the dimension of the inputs and d is the dimension of outputs. m is the number of latent variables between the first layer and the second layer. Typically $d \leq m \leq n$. The matrix W is learned from data with orthogonal constraint $WW^\top = I$. Therefore we call W the projection matrix because it projects the data into a lower dimensional space where the data dimension are orthogonal to each other. The matrix V groups the dimensions to which W maps the data. The grouping criteria is designed to minimize the number of groups to represent each data point. $V_{ij} = 1$ if the j-th output variable of the first layer is in the i-th group, otherwise $V_{ij} = 0$. Therefore we call V the grouping matrix. Given an input pattern $X^t \in \mathbb{R}^n$, the activation of i-th output unit of the second layer is $p_i(X^t; W, V)$ defined by

$$p_i(X^t; W, V) \triangleq \sqrt{\sum_{k=1}^{m} V_{ik} (\sum_{j=1}^{n} W_{kj} X_j^t)^2} \,. \tag{6.1}$$

ISA enforces the activation of the output unit to be sparse. To achieve the sparse activation, it minimizes the following loss function defined on T training instances:

$$\begin{aligned} \min_{W} \quad & \sum_{t=1}^{T} \sum_{i=1}^{d} p_i(X^t; W, V,) \\ \text{s.t.} \quad & WW^\top = I \,. \end{aligned} \tag{6.2}$$

Another way to interpret ISA is from sparse coding framework. Let $\mathcal{G} = [\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_d]$ denote the variable group indexes defined by V, that is, $j \in \mathcal{G}_i$ if and only if $V_{i,j} = 1$. $|\mathcal{G}_i|$ defines group size, which is generally the same across groups.

As in group LASSO ([31]), for any vector $\boldsymbol{a} \in \mathbb{R}^m$, we defined the group ℓ_1-norm $\|\boldsymbol{a}\|_{\mathcal{G},1}$ as

$$\|\boldsymbol{a}\|_{\mathcal{G},1} \triangleq \sum_{i=1}^{d} \sqrt{\sum_{j \in \mathcal{G}_i} \boldsymbol{a}_j^2} \, .$$

We can write $p_i(X^t; W, V)$ as

$$p_i(X^t; W, V) = \|WX^t\|_{\mathcal{G},1} \, .$$

Denote $\boldsymbol{\alpha}_t = WX^t$, since $WW^\top = I$, we have

$$X^t = W^\dagger \boldsymbol{\alpha}_t \, ,$$

where $W^\dagger$ is the Moore–Penrose pseudo inverse of W. Eq. (6.2) can be re-formulated as a sparse coding method that

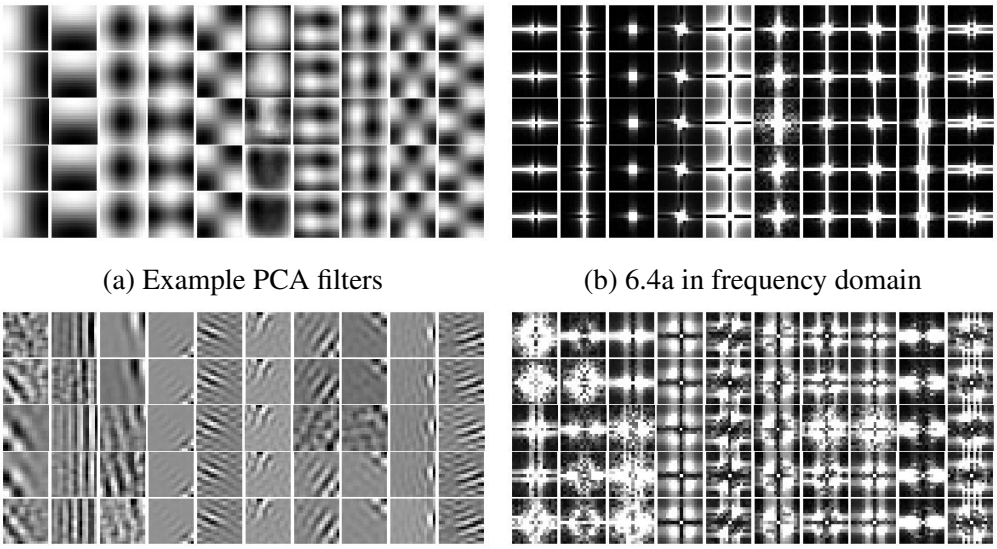
$$\begin{aligned} \min_{W, \boldsymbol{\alpha_t}} \quad & \sum_{t=1}^{T} \|\boldsymbol{\alpha}_t\|_{\mathcal{G},1} \\ \text{s.t.} \quad & (W^\dagger)^\top W^\dagger = I \qquad\qquad X^t = W^\dagger \boldsymbol{\alpha}_t \end{aligned} \tag{6.3}$$

Based on Eq. (6.3), ISA is essentially searching a group-sparse representation $\boldsymbol{\alpha}_t$ of the input signal X_t. The matrix $W^\dagger$ is the dictionaries of sparse coding. The orthogonal constraint of $W^\dagger$ makes the learned components maximally independent.

We select ISA as our unsupervised learning method because it is one of the best unsupervised feature learning methods [39]. ISA also connects to to the popular group Lasso algorithms.

Figure 6.4 visualizes, in both original and frequency domains, example filters learned from ISA and PCA models. As illustrated in Figure 6.4b and 6.4d , the ISA model learns more complex filters that capture higher-frequency information while PCA captures lower-frequency information. To have a more complete frequency coverage, we combine the outputs of ISA with an equivalent number of top outputs from PCA. As will be shown in the experimental section, our enhanced method, denoted by ISA+, significantly outperforms individual PCA or ISA model.

To reflect the different characteristics of different data sources, we design different network structures for pixel and optical flow data. Our learned descriptor for appearance stream LOP is learned by directly applying an ISA+ model to a stack of video frames and implicitly learning temporal pooling. Our learned descriptor of motion stream LOF is trained by applying an ISA+ model to each individual optical flow frame and explicitly performing a temporal pooling afterwards. This design difference of the network structures reflects our observation that pixel data has high temporal correlation while optical flow data often has much less temporal correlation due to the estimation error. As a result, it is much easier to learn temporally consistent appearance filters than temporally consistent motion filters. As shown in Figure 6.5, in which we show some example images and optical flow patches and the filters learned in both structures. From the columns in Figure 6.5a and 6.5b, it is clear that image patches are consistent across frames while optical flow patches have large temporal variation. Quantitatively, we estimate a 0.8014 pixels correlation while only 0.2808 for optical flow correlation by using the Pearson



(a) Example PCA filters

(b) 6.4a in frequency domain

(c) Example ISA filters

(d) 6.4c in frequency domain

Figure 6.4: Example filters learned from our ISA-IDT model. For all figures, y-axis represents same component at different time step and x-axis represents different components. In frequency visualization, zero-frequency component are centered in the figure.

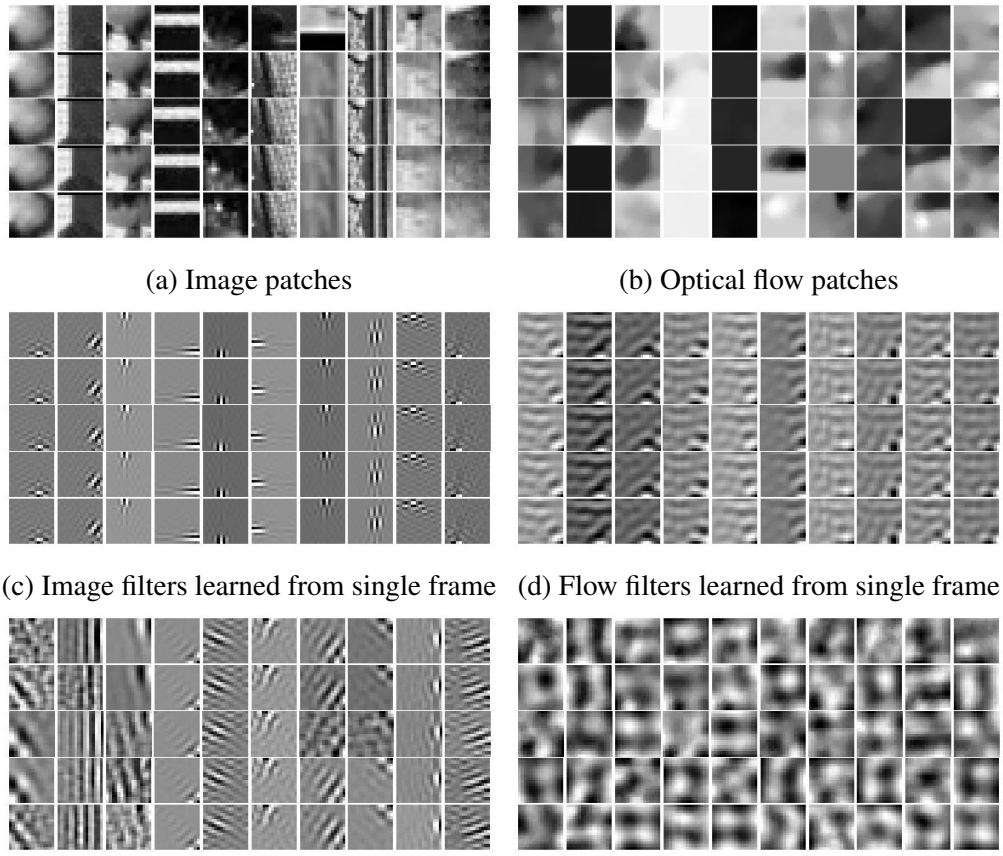
product-moment correlation coefficients to measure their temporal correlation from 100000 random sampled HMDB51 trackets. As a result, the learned appearance filters (Figure 6.5e) are temporally consistent and the learned flow filters are quite chaotic (Figure 6.5f). We suspect that a better optical flow will have higher correlation, but we have not explored this direction further because the Farneback optical flow we used has shown to be the best optical flow for IDT [125]. To discriminate the implicit temporal pooling of LOP from the explicit temporal pooling for optical flow data, we call it temporal projection.

6.6 Experiments

In the following section, we first show that our ISA+ model performs significantly better than either ISA or PCA. We then empirically demonstrate that custom designed network structures for pixel and optical flow data are necessary. After that, we compare our methods to the state-of-the-art video features in both descriptors and overall performance. We conduct experiments on HMDB51 and UCF101 datasets.

6.6.1 Experimental settings

As in [126], IDT features are extracted using 15 frame tracking, camera motion stabilization and RootSIFT normalization and described by Trajectory, HOG, HOF, MBH, LOP and LOF descriptors. Two-stream ISA-IDT models are trained on 200000 IDT trackets for each stream of data. For both PCA and ISA, we keep the filter size the same as in the handcrafted descriptors and use



(a) Image patches

(b) Optical flow patches

(c) Image filters learned from single frame

(d) Flow filters learned from single frame

(e) Image filters learned from 5 stacked frames

(f) Flow filters learned from 5 stacked frames

Figure 6.5: Example inputs and filters learned from our ISA models. For each figure, the y-axis represents same component at different time steps and the x-axis represents different component expect on Figure 6.5c and 6.5d, we replicate the filters 5 times for visualization purpose.

a pooling size of 10 for ISA. Another PCA is used to reduce the dimensionality of the resulting descriptors by a factor of two. For Fisher Vector encoding, we map the raw descriptors into a Gaussian Mixture Model with 256 Gaussians trained from a set of 256000 randomly sampled data points. After encoding, we attach the normalized space-time location information to the encoded descriptors as suggested in [67]. Power and ℓ_2 normalization are also used before concatenating different types of descriptors into a video based representation. For classification, we use a linear SVM classifier with a fixed $C = 100$ as recommended by [126] and the one-versus-all approach is used for multi-class classification scenario. We still need labels for training SVM classifiers, what we avoided is the need of labels for training the feature extraction model, which, because of its much larger parameter space, requires a much larger number of labels.

	PCA	ISA	ISA+
HMDB51	58.1%	58.4%	**61.5%**
UCF101	85.9%	86.2%	**88.3%**

Table 6.2: Comparison of different unsupervised feature learning methods.

	LOG		LOF	
	Projection	Pooling	Projection	Pooling
HMDB51	**52.4%**	44.3%	46.2%	**59.5%**
UCF101	**80.0%**	73.5%	79.6%	**84.8%**

Table 6.3: Comparison of temporal projection and temporal pooling.

6.6.2 ISA+ is better than individual PCA or ISA models

Table 6.2 compares our ISA+ model with individual PCA and ISA models. First, comparing PCA and ISA, we observe that, surprisingly, a simple PCA model can get similar results to a much more complex ISA model. These results demonstrate that PCA can learn good features when the number of features to generate is small. By combining the PCA and ISA outputs, we get more than 3% improvement on HMDB51 and 2% on UCF101. The improvements are significant given that these improvements are on the combined results of appearance and motion models where the potential for improvement is smaller.

6.6.3 Temporal projection versus temporal pooling

In Table 6.3, we compare the results of temporal projection and temporal pooling. As evidenced by the results of both datasets, for appearance modeling, temporal projection is better than temporal pooling, and for motion modeling, temporal pooling performs much better than temporal projection. Furthermore, if we compare single frame image filters (Figure 6.5c) to filters learned using 5 frame stacks (Figure 6.5e) , we can see that adding temporal variation can help to learn more complex filters. A potential improvement, therefore, is to explicitly enforce temporal coherence for optical flow filter learning and learn the temporal pooling for optical flow data.

6.6.4 Performance comparison of individual descriptors

	Appearance Descriptors				Motion Descriptors				
	HOG	S-CNNs	LOG	LOG	HOF	MBH	T-CNNs	LOF	LOF
HMDB51	42.0%	N.A.	47.2	**52.4%**	49.8%	52.4%	55.4%	51.0	**59.5%**
UCF101	72.4%	72.8%	79.3	**80.0%**	74.6%	81.4%	81.2%	81.2	**84.8%**

Table 6.4: Comparison of our proposed descriptors to IDT and two-stream CNNs.

In Table 6.4, we compare our learned descriptors LOG and LOF to the video descriptors

from IDT and spatial (S-CNNs) and temporal (T-CNNs) CNNs [1] from [106]. On the appearance descriptors, an impressive performance improvement of more than 10% over HOG, from 42.0% to 52.4% is achieved by LOG on HMDB51. For UCF101, LOG also gets more than 7% improvement over HOG and Spatial CNNs despite the fact that Spatial CNNs utilize additional training data. The same trend can be observed on motion descriptors. LOF outperforms other descriptors by more than 4% on HMDB51 and more than 3% on UCF101. Although it may not surprise that LOG outperforms HOG since it has been shown that unsupervised learned appearance descriptors can outperform handcrafted descriptors. However, as far as we know, we are the first to show that unsupervised motion descriptors (LOF) can outperform MBH, which is currently the best handcrafted motion descriptor. On the other hand, if we simply adopt the ConvISA [72] structures that we designed for learning from pixels, we get worse results (indicated by LOF (ConvISA)) than MBH. These results again show that unsupervised optical flow feature learning is quite difficult.

6.7 Conclusions

Contrary to the current trend of learning video features using end-to-end deep CNNs, which is computationally demanding and label intensive, we propose in this chapter to revisit the traditional local feature pipeline and combine the merits of both handcrafted and CNN approaches. As an example, we present a video feature learning algorithm called two-stream ConvISA that has better performance, lower computational expense than current state-of-the-art methods and does not require labels. We show that filters learned in an unsupervised fashion, when incorporated in convolution-pooling structures that are custom designed for pixel and optical flow data, can outperform supervised end-to-end networks. This result serves as a reminder that the design choices in handcrafted features may still have many useful properties which could be potentially incorporated into future deep action recognition networks. Currently, we haven't used this algorithm for MED because, although it is fast to learn the model, it is much slower than two-stream CNNs in applying the model. This slow application is mostly because extracting the tracklets is very expensive. Future work would be explicitly enforcing temporal consistency for optical flow feature learning and developing a deeper and better unsupervised learning methods. We would also like to explore end-to-end fine-tuning given the unsupervised learned networks, which is less expensive than training from scratch.

[1] The first split results from [106], pre-trained on Imagenet and trained HMDB51 and UCF101 together (multi-task learning)

Chapter 7

Feature Encoding: Space-time Extended Descriptors (STED)

7.1 Introduction

This chapter addresses the problem of encoding space-time information into the video representations. The spatial pyramid and its variants have been very popular feature models due to their success in balancing spatial location encoding and spatial invariance. Although it seems straightforward to extend spatial pyramid to the temporal domain (spatio-temporal pyramid), the large spatio-temporal diversity of unconstrained videos and the resulting significantly higher dimensional representations make it impractical. Instead, we introduce the space-time extended descriptor (STED), a simple but efficient way to include the spatio-temporal location into the video features. Parts of this chapter have been published in [61, 62].

MED is a difficult task since on-line videos are subject to large visual diversity. Robust to such variability, the BoW [23] model has been used extensively in representing videos. A BoW can be summarized as an encoding step and a pooling step [14]. Traditional pooling discards the local feature position information in the video space. However, this spatio-temporal information has been proven to be a discriminative clue [70]. Indeed, discriminative motion information is not equally distributed in the frame space as shown in Figure 7.1. To benefit from this information, spatial pyramids [70, 71] divides a video using fixed grids and pools the features locally into each grid cell. Spatial pyramids can be easily extended to spatio-temporal pyramid (STP) to encode the order of actions or events, another important cue for video representation. For example, to perform an action called "sitting down", we need to gradually bend our knees and lower our body, while "standing up" is the reverse. Despite its usefulness, STP is not as effective as its spatial sibling on still images due to the fact that actions in unconstrained videos will have much more dramatic spatio-temporal variance than still images. For example, as illustrated in Figure 7.2, we see the person doing a cartwheel moves through out the frame whereas a still image would center the person in the frame. In this case, STP, which commits too much to the artificial boundaries and may lead to a performance drop. Another major criticism of spatial pyramid is that it generates features with dimensions that are orders of magnitude higher than the spatio-temporal invariant representations and hence make it computationally expensive to

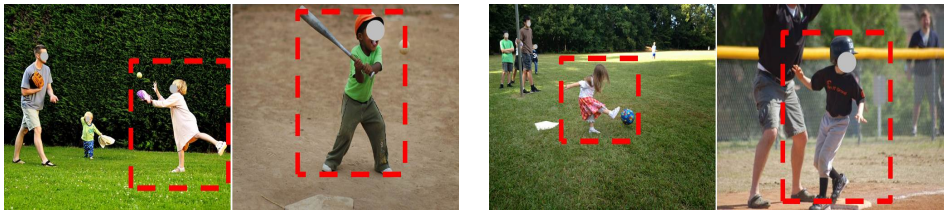


Figure 7.1: “Catch” and “hit” are likely to be distinguished by upper-bodies, especially hands while “kick” and “run” are more easily distinguished by legs.

Inter-videos Intra-video

Figure 7.2: In different videos, actions localization can be subject to variation due to camera viewpoint change. But, even within a single video sequence, the action area can change among frames.

process. A more effective and efficient space-time encoding method is therefore critical for video representations in retrieving videos from large data collections.

In this chapter, we propose to take advantage of the spatio-temporal discriminative information with an emphasis on retaining the spatio-temporal robustness while also controlling dimension explosion. Beyond standard spatial pooling which uses fixed segmentation grids, our proposed STED augments the feature descriptor with spatio-temporal location information. By simultaneously encoding motion and location information, we remove the necessity of a complicated pooling stage and the danger of committing to artificial boundaries. Experimental results on several benchmark datasets show the advantages of STED over STP.

In the remainder of this chapter, we start by providing more background information about spatial pyramids and its variants. We then compare STP and STED in detail. After that, an evaluation of our method is performed and demonstrates STED’s superiority over STP. Further discussions including potential improvements are given at the end.

7.2 Related Work

Video retrieval research is conducted in a diverse setting where emphasis includes low-level and high-level feature design [51, 108], multi-modality fusion [5, 65] and multi-modality retrieval models [48]. Here we focus on reviewing low-level features design and encoding for latter experimental comparison. There has been a large amount of work in building representations that keep spatial information of image patterns. Among them, spatial pyramid matching [71] is the most popular one. However, building spatial pyramids requires dimensions that are orders

of magnitude higher than the original spatial invariant representations and hence make it less suitable for high dimensional encoding methods such as FV [96] and VLAD [6]. Spatial FV [57] and spatial augmentation [83, 100] provide more compact representations to encode spatial information and show similar performance as spatial pyramid methods. Few approaches consider encoding global temporal information into video representations. Oneata *et al*. [91] shows that better action recognition performance can be achieved by dividing videos into two parts and encoding each one separately. Codella *et al*. [20] use temporal pyramids for event detection. They use n temporal segments, where n incrementally increases from 1 to 10.

7.3 Space-time Encoding Methods

In the following, we reformulate the spatial pyramid model and then extend this formulation to describe the STP and STED.

Let $D = \{d_1(\phi_1, x_1, y_1), d_2(\phi_2, x_2, y_2), ..., d_M(\phi_M, x_M, y_M)\}$ be a set of local feature descriptors extracted from a video. Each d_i contains the appearance/motion description ϕ_i, and the normalized pixel location (x_i, y_i) at which feature i is centered. We denote encoding function $g : \phi_i \rightarrow \mathbb{R}^K$ as a local feature encoding scheme such as sparse-coding or locality encoding. Note that here g solely relies on the appearance/motion portion of the descriptors. We further denote by $G = \{G_1, ..., G_n\}$ a set of grid cells. Each G_j is a binary matrix indicating which pixels are active, $G_j \in 0, 1^{(s_x \times s_y)}$, (s_x, s_y) being the image size. Based on those definitions, we express the average spatial pooling operation as

$$X_j = \frac{1}{n} \sum_{i=1}^{M} G^j_{(x_i, y_i)} \times g(\phi_i). \tag{7.1}$$

Basically, for each grid, average spatial pooling is the sum of all the encoded vector belonging to this grid and divide by a constant representing the number of grids.

7.3.1 Spatio-Temporal Pyramid (STP)

For temporal expansion, we simply add the temporal location into formula 7.1. That is

$$X_j = \frac{1}{n} \sum_{i=1}^{M} G^j_{(x_i, y_i, t_i)} \times g(\phi_i). \tag{7.2}$$

Note that by having one more location dimension, the resulting number of grid cells in G, hence number of dimension in feature X, will be orders of magnitude higher than the original spatial pyramid method. This may not be computationaly affordable for high dimensional representation such as FV. For example, if we want to represent an Improved Dense Trajectory (IDT) [125] feature (dimension of ϕ_i is 426) using a FV representation with 256 Gaussian mixture models, then a $k \times k \times l$ with $k \in (1, 2)$ and $l \in (1, 3)$ STP can result in a representation about 4.4 million dimensions. Whereas for the image case, if we use SIFT with the same configuration, then the resulting dimension will be only a quarter of a million dimension.

	UCF50 (mAcc. %)		HMDB51 (mAcc. %)		Hollywood2 (mAP %)		Olympic (mAP %)	
l	Single	Pyramid	Single	Pyramid	Single	Pyramid	Single	Pyramid
1	**92.7**		59.6		65.8		**89.8**	
2	92.3	92.6	61.0	61.3	66.3	**67.4**	87.8	89.3
4	91.6	92.3	60.5	**61.6**	65.2	67.0	85.2	87.6
8	90.4	92.1	58.1	**61.6**	62.2	65.7	83.8	83.8

Table 7.1: Comparison of different temporal pyramid levels for STP.

7.3.2 Space-Time Extended Descriptor (STED)

For STED, instead of using space-time information to locate the grid cells G, we use it to encode the feature, i.e., the mapping function g now includes the space-time information as inputs. Formally, we have,

$$X = \frac{1}{n}\sum_{i=1}^{M} g(\phi_i, x_i, y_i, t_i). \tag{7.3}$$

One advantage of STED is that we avoid having to commit to artificial grid boundaries to define the spatial pooling regions, which can lead to very divergent representations for similar actions happening in different space-time location. Another advantage is that the dimension of STED is only slightly higher than the original space-time invariant representation and is much lower than STP. Again taking the IDT + FV setting for example, for each video, STED generates a feature with only about 0.21 million dimensions, which is about 20 times lower than the STP representation.

7.4 Experiments

7.4.1 Experimental Setting

IDT with FV encoding [126] represents a current state-of-the-art for most real-world action recognition datasets. Therefore, we use it to evaluate our method. Note that although we use FV, our method can be applied to any quantization and pooling method such as VLAD [6]. Our baseline method uses the same settings as in [126]. These settings include the IDT feature extraction, FV representation and a linear SVM classifier.

IDT features are extracted using 15 frame tracking, camera motion stabilization with human masking and RootSIFT normalization and described by Trajectory, HOG, HOF and MBH descriptors. We use PCA to reduce the dimensionality of these descriptors by a factor of two. For FV representation, we map the raw feature descriptors into a Gaussian Mixture Model with 256 Gaussians trained from a set of randomly sampled 256000 data points. Power and L2 normalization are also used before concatenating different types of descriptors into a video based representation. For classification, we use a linear SVM classifier with a fixed C=100 as recommended by [126] and the one-versus-all approach is used for multi-class classification scenario.

STED	UCF50 (mAcc. %)	HMDB51 (mAcc. %)	Hollywood2 (mAP %)	Olympic (mAP %)	EK100 (mAP %)	EK10 (mAP %)
w/o	91.5	59.0	64.6	89.5	29.74	15.28
w	93.0	62.1	67.0	89.8	29.77	16.18

Table 7.2: Performance of STED.

For STP, we use $k \times k \times l$, k is 1 and 2 and $l \in (1, 2, 4, 8)$ grid cells. For STED, we attached the normalized 3 dimensional location to each descriptor as described in section 7.3.2.

7.4.2 Datasets

We use UCF50, HMDB51, Hollywood2, Olympic Sports, and MEDTEST2014 for evaluation.

7.4.3 Experimental results

Spatial-Temporal Pyramids (STP)

In Table 7.1, we compare the performance of the STP at different temporal pyramid levels l. $l = 1$ corresponds to results only using a 1×1 and a 2×2 spatial pyramid pooling. From Table 7.1, we can see that, due to the large spatio-temporal diversity of unconstrained videos, the usefulness of STP is inconclusive. For some datasets such as HMDB51 and Hollywood2, it provides significant performance improvement while for other datasets such as UCF50 and Olympic Sports, it hurts the performance. Further division (from level 2 to 4 or 8) almost always results in worse performance with the exception of the pyramids in the HMDB51 dataset. These results show that a straightforward extension of spatial pyramids may hurt the performance. Also, due to the high dimension, the computational cost for processing these encoded vectors are high.

Space-time Extension Descriptor (STED)

From Table 7.2, we see that, unlike STP, STED consistently improves the performance for all datasets. It is worth mentioning that both HMDB51 and Hollywood2 are very challenging datasets, more than 3% absolute improvement over space-time invariant representation is quite a notable gain. For Olympics dataset, because on average, each class only contains 8 testing examples, the improvement may not be statistically meaningful. Similarly, for MEDTEST2014 EK100 and EK10, the improvements from STED is not as significant as in other action recognition tasks. This smaller improvements may result from the larger variances of videos hence more difficult in deciding the influence of space-time location information.

7.5 Conclusions and Discussions

In this chapter, we proposed STED, a simple way to encode into local descriptors with spatio-temporal information. Despite it simplicity, STED is a much more effective and efficient way

to encode space-time location than previous methods. By simultaneously coding appearance, motion and location, STED avoids the danger of committing to artificial grid boundaries that define the spatial-temporal pooling regions and hence is better in dealing with unconstrained video that have large spatial-temporal motion diversity. We compared STED with STP, a straightforward way to extend spatial pyramids, and showed that STED generates representations with much lower dimension while achieving similar or better results. A potential improvement is to determine the optimal weighting for appearance/motion and location.

Chapter 8

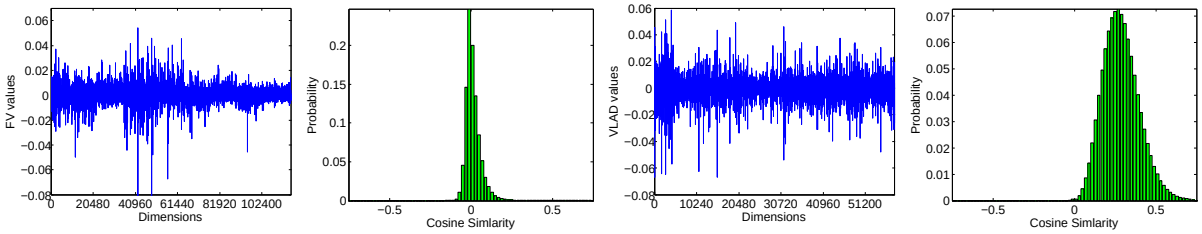
Feature Encoding: Ranking Normalization (RNorm)

8.1 Introduction

FV [95] and VLAD [46] are the current state-of-the-art local feature encoding methods that have been widely used to encode various features such as IDT[126] and CNN features [7, 25, 132].

However, when FV was first introduced by Jaakkola and Haussler [42] and applied to image classification by Perronnin and Dance [95], it did not attract a great deal of attention as it had not shown its superiority over BoW [95]. It is only after the introduction of power normalization (PNorm) [96] that significantly improves the performance of FV and thus make it into many researchers best practice list. The same phenomenon [46] was also witnessed for VLAD and intra normalization (INorm) [6]. The reason that FV and VLAD did not outperform BoW is that they suffer from two critical problems: **sparse distributions** and **bursty distributions**.

The sparse distribution problem [96] is where most values in the high dimensional FV encoded vectors are close to zero. It comes from the fact that, as the codebook (cluster) size increases, fewer local descriptors are assigned to each codebook. For each encoded vector, fewer descriptor assignments mean that some of components will have small and imprecise statistics. After ℓ_2 normalization, these numbers are inevitably close to zero. These sparse high dimensional vectors would lead to imprecise visual similarity measurement [96]. In Figure 8.1a and



(a) Example FV (b) FV Cosine dist. (c) Example VLAD (d) VLAD Cosine dist.

Figure 8.1: An example FV (resp. VLAD) and the pair-wise cosine similarity distribution of ℓ_2 normalized FV (resp. VLAD) encoded vectors from Hollywood2 dataset.

Figure 8.2: Repeated patterns from a brick wall (left) and camera motions (right).

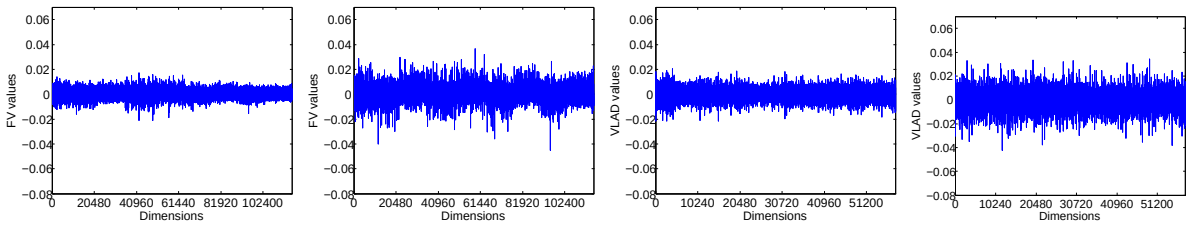
8.1b, we show a randomly sampled FV from Hollywood2 dataset and the cosine similarity distribution of all the ℓ_2 normalized FV encoded vectors in the whole dataset. As can be seen, the FVs are indeed sparse and their pair-wise cosine similarities are, as expected, clustered around zero.

The bursty distribution problem was first discovered by Hervé *et al*. [45] in BoW encoding and later observed by Arandjelovic and Zisserman [6] in VLAD encoding as well. Different from the sparse distribution problem that is caused by the encoding process, the bursty distribution problem comes from the data itself. It occurs when repeated patterns in an image or a video produce a few artificially large blocks (i.e. sum of residuals within a coarse cluster) in the encoded vectors. Examples of such patterns, as illustrated in Figure 8.2, are a brick wall in an image and camera motions in videos. Unlike the sparse distribution problem, these repeated patterns may not generate sparse vectors. while both VLAD and FV suffer from the sparse distribution problem, the bursty problem often happens in VLAD. For example, the VLAD vector in Figure 8.1c is much denser than the vector in Figure 8.1a (note the difference of scale in x-axis); the center of the pair-wise cosine similarity distribution (Figure 8.1d) of all vectors in the dataset is also far away from zero. However, because the encoded dimensions from these repeated patterns have large numerical values yet carry very little useful information, the similarity measures computed from the encoded vectors are still imprecise.

As it is a characteristic of the data rather than the encoding process, the bursty distribution problem should also exist in FV encoded vectors. Therefore, FV has both sparse and bursty distribution problems while VLAD only has the bursty distribution problem. This may explain the observations [132] that, without proper normalization, sometimes FV performs worse than VLAD even though it encodes more information than VLAD does.

The sparse and bursty distribution problems are often considered as two different problems as they are stemmed from different causes and generate different feature patterns. However, they share similar characteristics. For example, both of them lead to imprecise distance measure. Even the solutions have similar smoothing effects. In Figure 8.3, we plot the results of PNorm and INorm of the example FV and VLAD vectors in Figure 8.1a and 8.1c. As illustrated, both PNorm and INorm suppress large values and enlarge smaller ones.

A large amount of empirical evidence [6, 96] shows that both PNorm and INorm can significantly improve the performance of FV and VLAD. The unexpected effectiveness of these simple smoothing techniques set us thinking, wondering the reasons behind the success. If the smooth-



(a) PNorm (FV) (b) INorm (FV) (c) PNorm (VLAD) (d) INorm (VLAD)

Figure 8.3: The smoothing effects of PNorm and Inorm on the FV (8.3a and 8.3b) and VLAD (8.3c and 8.3d) encoded vectors from Figure 8.1a and 8.1c, respectively.

ing (compression) is so useful, there must be a large mismatch between the numerical values and ideal values of the encoded vectors. Here we define the ideal value of a dimension as how much it should contribute to the similarity measurement. In other words, if suppressing a large numerical value can get better performance, it means that the numerical value over-estimates its importance in determining the distance between two feature vectors. Similarly, the usefulness of increasing small numerical values suggests that these values may not be as negligible in distance measure as the numerical values specify. This is what we call the **value-mismatch problem**. It summarizes the reasons (problems) that cause the necessity of a normalization technique.

The value-mismatch problem is inherent in the unsupervised feature learning scheme. Without supervision, we really do not know how much useful information each pattern (dimension) carries. Some of the patterns, even if they only contain a small region of an image or a video hence having small values after encoding, can be very important in determining the class of the image or video. Some of the dimensions with large numerical values can be useless in classification. This situation is true even for end-to-end learned CNN features. When CNN features are applied to another task where fine-tuning is not available or not applicable [25, 132], regardless of their numerical values, it is difficult to tell how useful each dimension is for the target task.

As we do not know how important each dimension is, the PNorm and INorm can only help alleviate the value-mismatch problem. It is often difficult to decide how much dispersal each task requires. In most cases, even after certain suppression or dispersion, the numerical values in each dimension still do not represent their ideal values. Such being the case, we argue that a better way to solve the value-mismatch problem is to treat all dimensions with equal importance, namely, to use the rank to replace the original values of FV and VLAD encoded features. We call this new normalization method rank normalization (RNorm).

RNorm ranks all video representations in a dataset along each dimension and uses the normalized rankings in place of the original video representations for the subsequent classification. Since it only needs a ranking operation, it is computationally efficient and parameter free. Our experiments also show that RNorm can significantly outperform PNorm and INorm. Our empirical results also show that an approximate ranking is enough for RNorm and it can be applied to not only FV/VLAD, but also to local descriptors.

In the remainder of this paper, we first review some relevant works about the improvement of feature encoding, mostly associated with FV and VLAD. We then detail the encoding methods, baseline method, and evaluation benchmarks we used. Next, we describe the proposed method

and demonstrate its performance gain over the baseline approach (composed of IDT, FV, and SVMs) on six human video classification datasets. After that, we describe the proposed approximate ranking method and extend the usage of RNorm to local descriptors. Finally, we conclude our work on feature encoding.

8.2 Related Work

Features and encoding methods are the major sources of breakthroughs in conventional video representations. Among them the trajectory based approaches [53, 126], especially the DT, IDT [125, 126], and FV encoding, are the basis of current state-of-the-art algorithms.

FV and VLAD are very similar encoding methods [6] and both have been popular for image and video classification [6, 85, 96, 126, 132]. In the original scheme [44, 95], they either do not require post processing [95] or use only ℓ_2 normalization [44]. Although ℓ_2 normalization can reduce the influence of background information and transform the linear kernel into an ℓ_2 similarity measurement [96], it does not disperse the data. As a result, the original FV encoding method showed inconclusive results compared to other state-of-the-art encoding methods [95]. It is the introduction of PNorm [6, 96] which significantly improved the performance of those encoding methods and thus makes them useful in practices. PNorm helps alleviate the problem of sparse and bursty distribution of FV and VLAD. Later on, as a special design for VLAD, Arandjelovic and Zisserman [6] proposed INorm to further reduce the bursty distribution problem of VLADs. Jégou and Chum [44] used PCA to decorrelate a low dimensional representation and adopted multiple clustering to reduce the quantization errors for VLADs. Nonetheless, those methods only alleviated the burstiness problem [45]. The sparsity of the encoded descriptors still depends on that of the original data. Unlike the above two approaches, RNorm normalizes each dimension of FV and VLAD encoded features to evenly distribute the energy to each dimension, regardless of how sparse the original data are.

8.3 Background

In this section, we detail some background information of the FV and VLAD encoding methods, PNorm and INorm, ℓ_2 normalized baseline method, and evaluation benchmarks.

8.3.1 FV and VLAD

After getting the IDT local descriptors, FV and VLAD are applied to do the encoding. For FV, we first trained a Gaussian Mixture Model (GMM) with 256 Gaussians for each type of descriptor and then map all the descriptors in that type to the model. Let $\gamma_t(i)$ be the soft assignment of descriptors x_t to Gaussian u_i:

$$\gamma_t(i) = \frac{w_i u_i(x_t)}{\sum_{j=1}^{K} w_j u_j(x_t)} \tag{8.1}$$

where w_i is the mixture weight of Gaussian u_i and K is the number of Gaussians. Let D be the dimension of the descriptors x_t; and $\mathcal{G}^X_{\mu,i}$ and $\mathcal{G}^X_{\sigma,i}$ are the D-dimensional gradient with respect

to the mean μ_i and standard deviation σ_i of u_i, respectively:

$$\mathcal{G}^X_{\mu,i} = \frac{1}{T\sqrt{\pi_i}} \sum_{t=1}^{T} \gamma_t(i) \left(\frac{x_t - \mu_i}{\sigma_i} \right), \tag{8.2}$$

$$\mathcal{G}^X_{\sigma,i} = \frac{1}{T\sqrt{2\pi_i}} \sum_{t=1}^{T} \gamma_t(i) \left[\frac{(x_t - \mu_i)^2}{\sigma_i^2} - 1 \right], \tag{8.3}$$

where T is the number of descriptors. The final encoded vector is the concatenation of $\mathcal{G}^X_{\mu,i}$ and $\mathcal{G}^X_{\sigma,i}$ for $i = 1...K$ and is therefore $2KD$ dimensional. More detailed descriptions of FV can be found in [96].

The VLAD encoding is similar to FV encoding. The differences are VLAD often uses K-means clustering instead of GMMs and only records the residuals between the descriptors and the cluster means. That is to say, VLAD only has the $\mathcal{G}^X_{\mu,i}$ part and the final dimension would be KD.

8.3.2 PNorm and INorm

Although it is customary to use PNorm for FV and INorm for VLAD, they have very similar effects, as can be seen in Figure 8.3. PNorm has the following element-wise operation:

$$f(z) = sign(z)|z|^\alpha, \tag{8.4}$$

where $0 \leq \alpha \leq 1$. Without further specification, this paper uses PNorm with $\alpha = 0.5$, which is the most commonly used one and also named as signed square root (SSR) normalization [6]. From this formula, we can tell that PNorm is essentially a smoothing function that suppresses large numerical values and enlarges small ones.

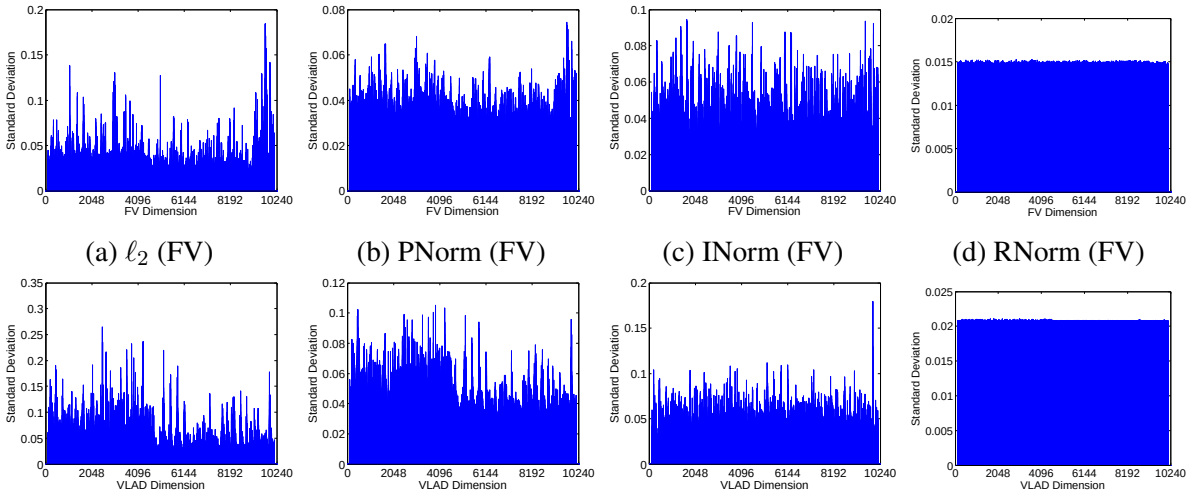
The INorm, which ℓ_2 normalizes each VLAD block independently, has a very similar smoothing effect. However, from Figure 8.3, we can see that PNorm has a stronger normalization effect than INorm has. In general, we believe that PNorm is better in dealing with the value-mismatch problem than INorm because PNorm can smooth out both the problems that happen in individual dimension and the whole block while INorm can only deal with the block-wise value-mismatch problem. As mentioned before, a block here means a part of vector that represents one cluster. For example, if we use 256 k-mean clusters for VLAD, then each VLAD vector has 256 blocks.

8.3.3 Evaluation benchmarks

For action recognition, we use both Hollywood2 and Olympic Sports datasts. For the MED task, we evalute on both MEDTEST13 and MEDTEST14 datasets.

8.3.4 Experimental settings

We follow the experimental settings in [67]. More specifically, we use IDT features extracted using 15 frame tracking and camera motion stabilization. PCA is utilized to reduce the dimensionality of IDT descriptors by a factor of two. After reduction, the local descriptors are augmented



(a) ℓ_2 (FV) (b) PNorm (FV) (c) INorm (FV) (d) RNorm (FV)

(e) ℓ_2 (VLAD) (f) PNorm (VLAD) (g) INorm (VLAD) (h) RNorm (VLAD)

Figure 8.4: The effect of various normalization methods to the FV and VLAD encoded vectors (Note that the scales are different).

with three-dimensional normalized location information [67]. FV (resp. VLAD) encoding maps the raw descriptors into a GMM (resp. K-means) with 256 clusters trained from a set of 256000 randomly sampled data points. Classification in conducted by a "one versus rest" linear SVM classifier with a fixed $C = 100$ as in [126] for fair comparison. For PNorm, we use SSR unless otherwise stated.

8.4 Rank Normalization (RNorm)

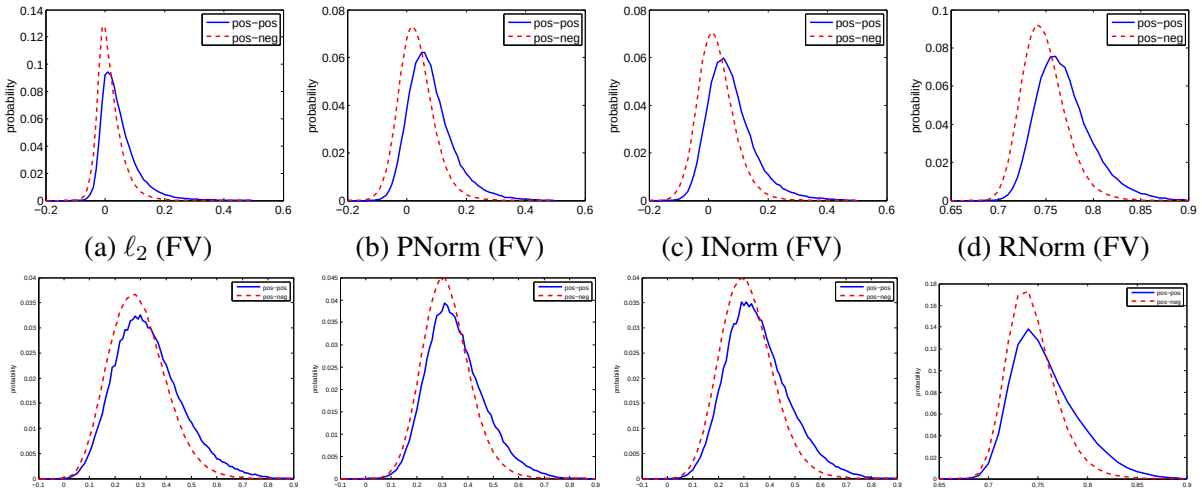
RNorm applies to each dimension of the FV and VLAD encoded vectors the following function:

$$f(z) = rank(z)/N,$$

where $rank(z)$ is z's position after sorting along the dimension of all N FV or VLAD vectors in a dataset. In the following, we will provide our experimental analysis about RNorm.

8.4.1 Qualitative analysis

Figure 8.4 shows the standard deviation (i.e. energy) of the values for each dimension of FV and VLAD encoded vectors across all the videos in Hollywood2 dataset. It can be observed that the energy is strongly concentrated around only a few dimensions in the ℓ_2 normalized encoded vectors(8.4a and 8.4e). These peaks strongly influence the similarity scores. PNorm (8.4b and 8.4f) and INorm (8.4c and 8.4g) indeed manage to discount their effect. However, even with PNorm and INorm, it is clear that the distributions are still peaky. RNorm (8.4d and 8.4h) completely alleviates this effect and evenly distributes the energy across different dimensions. If we compare PNorm and INorm, we would observe that PNorm has a better smoothing effect



(a) ℓ_2 (FV) (b) PNorm (FV) (c) INorm (FV) (d) RNorm (FV)

(e) ℓ_2 (VLAD) (f) PNorm (VLAD) (g) INorm (VLAD) (h) RNorm (VLAD)

Figure 8.5: The cosine similarities of videos under different normalization methods. The blue and plain lines are the distributions of pair-wise cosine similarities between positive samples, and the red and dashed lines show the distributions of pair-wise cosine similarities between positive and negative samples.

(note that the scales are different). Also, by comparing Figure 8.4e and Figure 8.4g, we can see that INorm can suppress large blocks. However, INorm also creates artificially large dimension and cannot preserve the order of the original values. These characteristics again show that INorm is not as desirable as PNorm or RNorm for smoothing normalization.

The improvements on similarity measurement brought by these normalization methods are also obvious. In Figure 8.5, we visualize the distribution of pair-wise cosine similarity between positive/positive and positive/negative samples. First, by comparing Figure 8.5b, 8.5c and 8.5d to 8.5a, we observe that, for FV, after normalizations, the data samples are much more separable. Instead of clustering around zero, the normalized vectors have much broader range of similarity distributions. Second, if we compare the Figure 8.5a to 8.5e, we can see that, unlike FV, where the cosine similarities cluster around zero, the similarity distribution center of VLAD encoded vectors are far away from zero. This observation again confirms that VLAD encoded vectors are not as sparse as FV encoded vectors. It also explains why the improvements for VLAD are much more subtle. Comparing Figure 8.5e, 8.5f, 8.5g and 8.5h, we can see that why the improvement for VLAD aren't as great.

In Figure 8.6, we also compare the PNorm and RNorm on the first dimension of FV encoded vector on the Hollywood2 dataset. The x-axis shows the original values and the y-axis are the corresponding values after normalization [1]. For a better visualization, we normalize all the curves so that their values are between -1 and 1. For example, PNorm returns the original values when $\alpha = 1$ and becomes a step function as $\alpha = 0$. When $0 < \alpha < 1$, PNorm carries out a mapping like a sigmoid activation function. It magnifies those values around zeros and allows them to possess more of the y-axis space. RNorm has a similar effect as PNorm except it is not

[1] INorm cannot be visualized in this way as it does not preserve the order of the values.

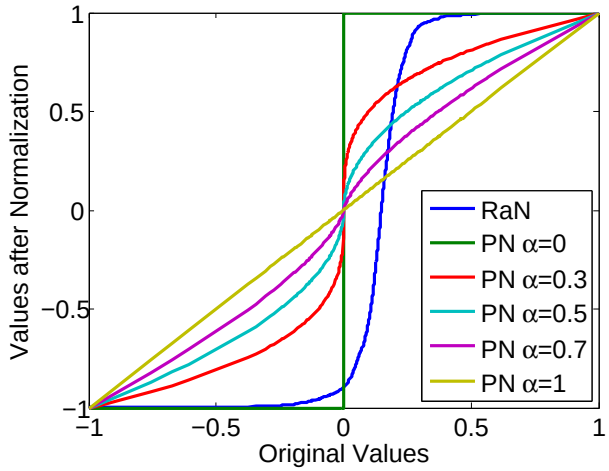


Figure 8.6: Comparison of the effects of PNorm (PN) and RNorm (RaN) on the first dimension of FV in Hollywood2. For a better visualization, we normalize all the curves so that their values are between -1 and 1.

constrained to maintain sign.

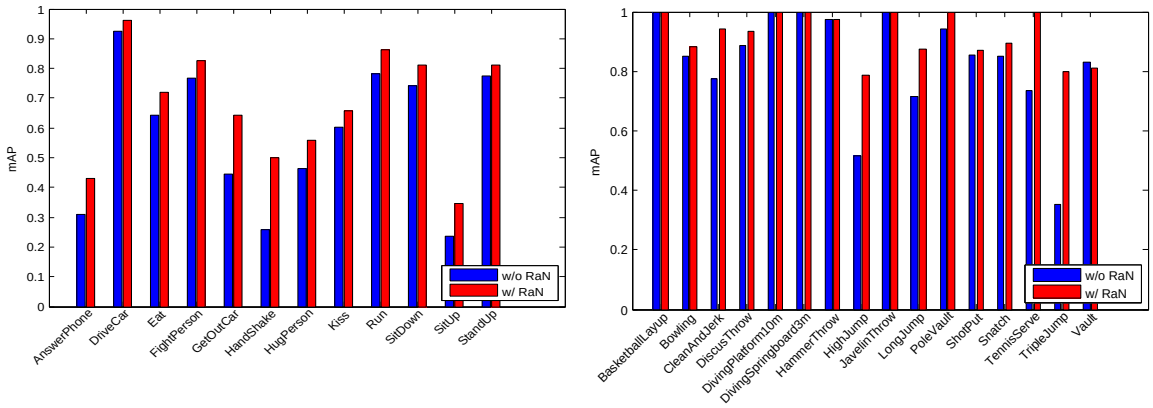
8.4.2 Quantitative analysis

In Table 8.1, we compare the results of different normalization methods on both Hollywood2 and Olympic Sports datasets. ℓ_2 normalization serves as a baseline method. First, if we compare the baseline results of FV and VLAD, we can see that only using ℓ_2 normalization, FV performs worse than VLAD. These results are consistent with our hypothesis that FV suffers more severe value-mismatch problem than VLAD has. After normalization, FV significantly outperforms VLAD, which again confirm our analysis that with better normalization, FV can outperform VLAD. Second, if we compare PNorm with INorm, we can see that PNorm always achieves better results than INorm, which is in line with our observation in Figure 8.4 that INorm is not as good as PNorm. Finally, RNorm consistently outperforms all other normalization methods, especially for FV encoding. Compared to the baseline method in FV, RNorm gives about 10% absolute improvement on a difficult dataset like Hollywood2; on the relatively easy one like Olympic Sports that has less potential to explore, RNorm still manages to improve by more than 9%, absolutely.

Figure 8.7 shows the per-class comparison of with and without RNorm. Most remarkably, RNorm improves the baseline method on all 12 actions from the Hollywood2 dataset. On some of the hard classes like 'HandShake', RNorm improves the baseline results by more than 15%. A similar trend can be seen in the Olympic Sports dataset. Compared to the baseline method, RNorm either improves or delivers similar results on 15 out of 16 actions. These per-class performance comparisons show that our improvements are robust and significant.

	Hollywood2 (%)		Olympic Sports (%)	
	FV	VLAD	FV	VLAD
Baseline (ℓ_2)	57.9	59.6	83.0	87.5
PNorm	66.1	62.2	90.1	88.6
INorm	65.7	61.6	89.0	88.5
RNorm	**67.7**	**62.4**	**92.3**	**88.9**

Table 8.1: Performance comparison of different normalization methods on action recognition datasets.



(a) Hollywood2

(b) Olympic Sports

Figure 8.7: Per-class performance comparison of the baseline performances with and without RNorm.

8.4.3 Results on MED

Table 8.2 lists the overall MAP on all four datasets of MED. The baseline method is the same as in action recognition tasks. First, on all four datasets, PNorm, INorm and RNorm all notably improve the baseline results. RNorm still consistently outperforms other methods. It improves the baseline method by around 3% on both EK10 and EK100 tasks. It is worth emphasizing that MED is such a challenging task that 3% of absolute performance improvement is significant. These results demonstrate that RNorm is robust across tasks with different difficulty levels and video types.

	MEDTEST13 (%)		MEDTEST14 (%)	
	EK10	EK100	EK10	EK100
Baseline (ℓ_2)	17.0	33.6	12.0	26.2
PNorm	19.3	36.3	14.9	29.0
INorm	19.1	35.8	14.5	28.9
RNorm	**20.2**	**36.6**	**15.4**	**29.3**

Table 8.2: Performance Comparison on the MED task.

S	Hollywood2 (%)	Olympic Sports (%)
2	43.8 ± 24.0	31.6 ± 27.5
4	67.0 ± 0.4	91.6 ± 1.1
6	67.6 ± 0.2	92.1 ± 0.9
10	67.5 ± 0.3	92.6 ± 0.4
50	67.6 ± 0.1	92.7 ± 0.2
100	67.7 ± 0.1	92.7 ± 0.1

Table 8.3: Comparison of different seed size S for RNorm. Each experiment is repeated 10 times and the mean values and standard deviations are given.

8.5 Approximate Ranking

The exact ranking we have used so far requires each vector to compare all other vectors in the database. Although computationally this is not a problem as we have efficient sorting algorithms; in practice, we may not have all the data available to compute the rank. For example, in on-line learning scenario. This problem motivates us to develop an approximate ranking algorithm. The algorithm is similar to product quantization [47] except it quantize each dimension separately. First, we randomly choose a small subset of vectors as our seed vectors. These seed vectors serve as the "codebook" for computing the rank of a new vector. Given the seed vectors, each dimension is first sorted, and a rank is assigned to each dimension of a seed vector. When a new vector is given, for each dimension, the nearest neighbor in the seed vectors is found, and the rank of the nearest neighbor is applied to the current dimension of the new vector. For example, if we have S=2, then each dimension of the target vectors will have value that is either 0 or 1, depending on whether it is closer to the smaller or larger values of the same dimension in the two seed vectors. Similarly, seed vectors with size S=4 means that we quantize each dimension into a value range of 0 to 3, which also implies that we only need 2 bits rather than 32 bits to store each dimension. This is significant in terms of saving storage space. Let us see how the accuracy change with respect to the approximate ranking. We conduct experiments with different S on both Hollywood2 and Olympic Sports datasets using FV encoded vectors. The results are shown in Table 8.3, in which we report the mean values and standard deviations of 10 repeated experiments for each configuration. Surprisingly, a seed size S as small as 4 is good enough to achieve a result that is almost as high as precise ranking. That is to say, with approximate ranking, we can save significant amount of storage space with minimal performance drop.

8.6 Local Descriptors Normalization

So far, we have been discussing the use of different normalization methods to address the value-mismatch problem of FV and VLAD encoded features. In this section, we extend the usage of these normalization methods to local descriptors, where the value-mismatch problem also exists. As we have discussed before, the value-mismatch problem originates from the unsupervised/handcrafted feature generation mechanism, where we do not know which dimension is more useful in similarity measurement. As the local descriptors are handcrafted, they also suffer

	Hollywood2	Olympic Sports
Baseline (ℓ_2)	57.9%	83.0%
PNorm	58.2%	83.7%
INorm	58.0%	83.3%
RNorm	58.1%	84.0%

Table 8.4: Performance comparison of different normalization methods on local descriptors.

from the value-mismatch problem. This mismatch is mostly because of the repeated patterns in images and videos. Table 8.4 shows the effects of different normalization methods on local descriptors. For RNorm, we use the approximate ranking with a seed size of 4. Unlike the results in Table 8.1, the improvements here are quite small but consistent. This improvement is because the number of clusters in local descriptors is quite small (8 or 9). However, since we only need 2 bits instead of 32 bits to store each dimension of the local descriptors, RNorm can still be used as an efficient way to store these descriptors.

8.7 Conclusions

In this chapter, we have shown that both FV and VLAD bear the value-mismatch problem, which is caused by the sparse and bursty distribution phenomena. The classic PNorm and INorm can only mitigate this problem. Compared to PNorm and INorm, our proposed RNorm is a better way to handle this problem. It evenly distributes the energy across dimensions while preserving the order of the original values. Experimental results on six real-world datasets also demonstrate that RNorm is better than PNorm and INorm. Furthermore, our proposed approximate ranking significantly improves its computation and storage efficiency while retaining the accuracy. The local descriptor ranking, although only having small but consistent accuracy improvement, can also be used to save storage space.

Chapter 9

Fusion: Double Fusion

9.1 Introduction

In previous chapters, we have talked about several features and encoding methods. In this chapter, we will talk about how to fuse the resulting representations. Fusion is particularly important for MED as the complexity of the problem requires a combination of multiple sources of information. Generally speaking, early fusion and late fusion [109] are the two most popular combination strategies. The former one fuses features before performing classification and the latter one combines outputs of different classifiers. Early fusion can better capture the relationship among features yet is prone to over-fit the training data. Late fusion deals with the over-fitting problem better but each classifier can only see part of the information. In this chapter, we introduce a new fusion scheme named double fusion, which effectively combines early fusion and late fusion to incorporate their advantages and mitigate their disadvantages. Parts of this chapter have been published in [63, 64].

Historically, researchers in image and video retrieval [41, 52, 73, 109, 136] found that a combination of multi-modality information almost always boosts retrieval accuracy. Early fusion combines features before performing classification using methods such as multi-kernel learning [21, 33]. Late fusion combines output of classifiers from different features using methods such as average fusion, committee voting [120] and co-regularized least squared regression [15]. There is no universal conclusion of which strategy is preferred for multimedia content analysis and retrieval. Snoek *et al*. [109] found that early fusion is better than late fusion in TRECVID 2004 semantic indexing task. By studying data on TRECVID 2006 semantic indexing task, Ayache *et al*. [8] found that early fusion gets better results on most of tasks while late fusion is more robust and can handle some harder tasks. To incorporate the advantages of both methods, we introduce a simple yet efficient fusion strategy called double fusion. In double fusion, we first perform early fusion to generate different combinations of features from subsets of features. After that, we train classifiers on each feature or feature combination and use late fusion to combine the outputs of all those classifiers. For example, as shown in Fig. 9.1, we first extract three kinds of features (visual, audio and text) from training and testing videos. After that, pairwise early fusion (visual+audio, visual+text) are carried out in these three features by combining their kernel matrices. In the training step, five classifiers are trained based on five features and feature combinations (visual,

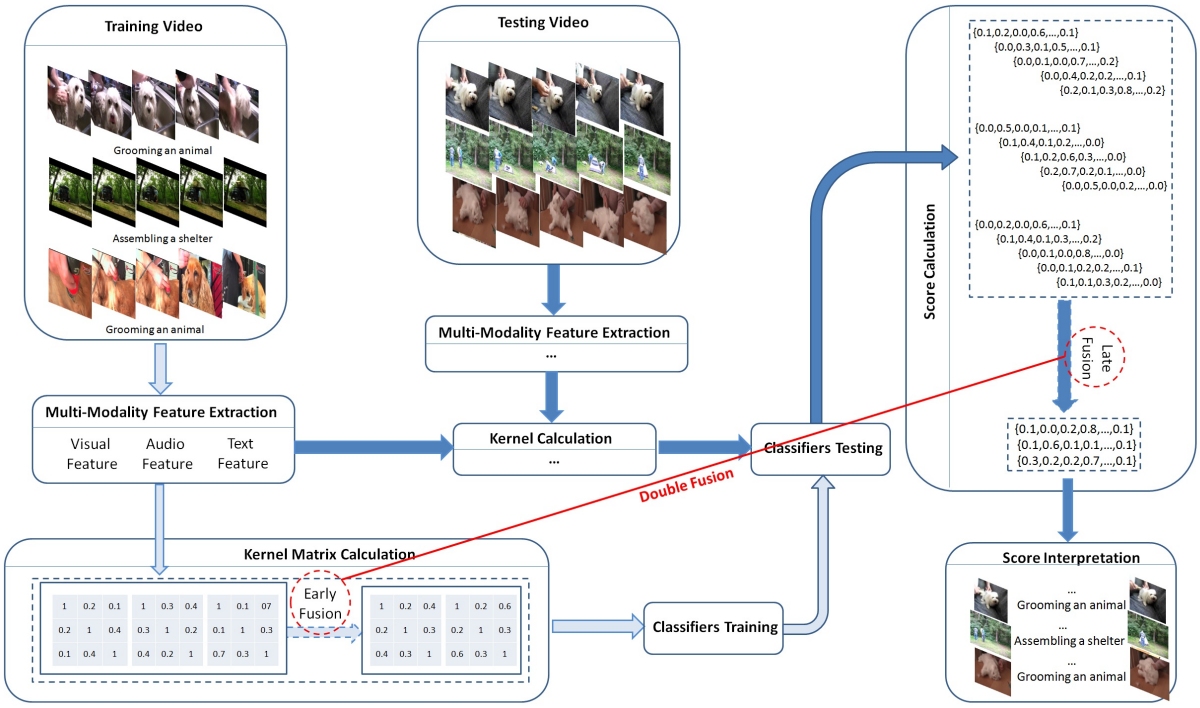


Figure 9.1: The illustration of our MED system.

audio, text, visual+audio, visual+text). For each video, there are five scores indicating how likely it is that this video belongs to the event. In the last step, late fusion is used to fuse five score vectors into one score vector, on which the final interpretation can be done. Experimental results on the TRECVID MED 2010 and MED 2011 data sets with about 1684 hours' video clips for 18 events show the effectiveness of double fusion. For MED 2010 we get a mean minimal normalized detection cost (MNDC) of 0.49, which exceeds the state-of-the-art performance [52] by more than 12%.

The remainder of the chapter is organized as follows. Section 9.2 briefly introduces different fusion strategies that motivate our work. Section 9.3 presents the details of our E-lamp system, including feature representation, BoW scheme, classifiers and fusion schemes. Section 9.4 analyzes experimental results on MED 2010 and MED 2011. We conclude this chapter in Section 9.5.

9.2 Fusion Scheme

Early Fusion [109] is a combination scheme that runs before classification. Both feature fusion and kernel space fusion are examples of early fusion. The main advantage of early fusion is that the classifier can "see" all the features at once and only one learning phase is required. However, due to their different meanings and value ranges, it is often difficult to combine features that are vastly different in feature spaces [109]. Multiple kernel learning (MKL) [21] is one of the most popular early fusion technologies. Its drawback is the curse of high dimensionality, usually

accompanied by limited training data. Our observations show that early fusion will be negatively affected by features that have relatively low performance.

In contrast to early fusion, late fusion [109] happens in prediction score space. After classification, all the features lie in the same score space. Therefore, late fusion is often much easier to perform and does not have the problem of two features which are not compatible. However, in late fusion, because we classify each feature individually, the relationships among different features become more difficult to exploit. Normally, another learning procedure is needed to combine these outputs, but because of the overfitting problem, simply average fusion is often used to combine the output scores and often yields better or at least comparable results than training another classifier for fusion. Compared to early fusion, late fusion is more robust to features that have negative impact.

We introduce a method called double fusion. Double fusion combines early fusion and late fusion. Specifically, for early fusion, we fuse multiple subsets of features by using standard early fusion technologies; for late fusion, we combine outputs of classifiers trained from single and combined features. By using this scheme, we can freely combine different early fusion and late fusion techniques, and get benefits of both methods. Given n features, we first train n classifiers for n individual features. We then perform early fusion to combine some of the features. Because of the explosive number (the number of combinations is $2^n - 1$, n is the number of features) of combinations, it is computationally expensive to explore all possible feature combinations when the feature space is large. Therefore, we select two types of combinations including category-wise combination and all-future combination. In category-wise combination, we map all features belonging to the same categories into a single feature and generate c new classifiers, where c is the number of categories. For example, all visual features belongs to a feature type. In all-feature combination, we just combine all the features as in early fusion. We use two early fusion strategies, i.e., rule-based combination and MKL [21]. Rule-based combination simple averages the kernel matrices. MKL [21] is an extension of average combination that automatically learns the fusion weights for different kernel matrices. Our experimental results show that MKL is only slightly better than average fusion.

9.3 Implementation

9.3.1 System details

As shown in Fig. 9.1, there are four key steps in our system. Step one does feature extraction on visual, textual and audio modality. After modality specific data processing, BoW representation is used to aggregate the local descriptors into video representations. Early fusion is applied in step two after calculating the kernel matrices. In step three, classifiers are trained to get models. After classification, the outputs of different classifiers are combined through late fusion.

Features

Features are critical for video content understanding. In our MED system, we explore four feature modalities including visual, audio, text, and concepts based features.

Visual Features We use five visual features, namely SIFT [79], CSIFT [119], Transformed color SIFT (TSIFT) [119], MoSIFT [16], STIP [69] and GIST [90].

For SIFT, CSIFT and TSIFT features, the harris-laplace key point detector is used to detect interest points. As processing all MED video frames is computationally expensive, we only extract features from key frames generated by a shot boundary detection algorithm. Specifically, the algorithm calculates the color histogram for every five frames and subtracts the histogram with the histogram of the previous frame, if the value is larger than a certain threshold, which is empirically set, the key frame will be a shot boundary. After detecting a shot, we use the frame in the middle of the shot to represent that shot. By using this algorithm, we extracted 114992 key frames from MED 2010 and 364747 key frames from MED 2011 training dataset and 1035417 key frames from MED 2011 test dataset.

While SIFT, CSIFT and TSIFT describe 2D local structure in images, STIP and MoSIFT capture space time volumes where the image values have significant local variations in both space and time. STIP and MoSIFT are different in both interest points detectors and descriptors. STIP uses 3D Harris corner detectors and its interest points are represented in two parts: the first part is a HOG (Histogram of Oriented Gradients; 72 dimensions), which indicates the spatial appearance and the second part is a HOF (Histogram of Optical Flow; 90 dimensions), describing the motion information. MoSIFT uses a Difference of Gaussian (DoG) based detector and is represented by another descriptor which is also concatenated from two parts: the first part is a SIFT (128 dimensions), which indicates the spatial appearance and the second part is also a HOF (128 dimensions).

For the GIST feature, we follow the suggestion from [90] and set the dimension of descriptors to 960. However, because it did not help in improving our final system accuracy, we did not use it for our MED 2011 final submission. The reason that GIST is not helpful for our system is that it is a very weak visual feature that cannot impact the system given the existence of other strong visual features.

Audio Features For the audio features, we used the Automatic Speech Recognition (ASR) features, whose extraction process is as described in [73]. Briefly speaking, a simple speech-to-text system is used to automatically transcribe the audio tracks of the videos. The system is for American English and trained on a variety of audio sources, including Broadcast News and 'Meeting' audio. It has a vocabulary of about 40k words. The outputs of the system consist of word strings. In addition to ASR, we also use MFCC features as in [52] to capture other sound besides spoken word.

Textual Features Following the work of [73], we use the Optical Character Recognition (OCR) feature extracted by the Informedia system to represent the text feature. To extract OCR features, we first identify text blocks based on the assumption that text blocks consist of short edges in vertical and horizontal orientations. Canny filters and morphological operators are used to perform the edge detection. We then use a commercial OCR system to recognize the text within the text blocks. However, because OCR rarely gets a correct and complete word, instead of treating the misspelled word as tokens, we treat each trigram of characters as a token. For example the word "scre∼en" will be split into "scr", "cre", "re∼", "e∼e" and "∼en". This method minimizes the size of the dictionary.

Concepts-based Features In our MED 2011 final submission, we used two high level features. The first one, called SIN346, is trained from the data provided by TRECVID Semantic Indexing (SIN) track. This feature has 346 dimensions that represents 346 concepts in SIN . The second one is a set of face attributes extracted by PittPatt. After recognizing faces in videos, we count the numbers of faces in each video, the maximum number of faces within each frame, the number of frontal faces and other attributes of faces to represent the video.

BoW Representation

After extracting the features described in the previous section, a BoW representation is adopted to cast local features into global video features. First, K-means is used to cluster feature descriptors into a large number of clusters (i.e. 'words'). For visual features, the codebook size is 4096 except GIST, which has 960 dimensions. Second, by mapping these features into their cluster centroids and summing all the mapped local features, we get frame-based representations. We use a soft-weight strategy that selects the ten nearest clusters and assigns a rank weight to them. Finally, we transform the frame-based features into video-based features. For SIFT, CSIFT, TSIFT and GIST, we first normalize feature vectors of each key frame in a video and then sum them together to represent the video. For STIP and MoSIFT, we just sum all the feature descriptors in a video together and normalize it. As for ASR and OCR, we simply count the number of words or tokens found in videos without doing the frame-based feature step. There are a total of 11618 unique words and 180228 unique tokens extracted for ASR and OCR, respectively.

Spatial Pyramid Matching

Since the classic BoW method discards all information about the spatial layout of features, Lazebnik *et al*. [71] proposed the pyramid matching scheme by repeatedly subdividing the image and computing histograms of local features for each sub-regions. We use the same method in our system. Specifically, besides the BoW representation for the whole image, we divided each key frame into 2x2 and 1x3 sub-regions, and computed the BoW representation for each sub-region. The feature dimension for our visual feature vector is 8x4096=32768. We applied this simple and effective method for SIFT, CSIFT,TSIFT, MoSIFT and STIP.

Classifiers

A large variety of classifiers exist for mapping the features into score space. In this chapter, we use two classifiers, i.e. non-linear support vector machine (SVM) and kernel regression (KR) [15]. SVM is one of the most commonly used classifiers due to its simple implementation, low computational cost, relatively mature theory and high performance. In TRECVID MED 2010, most of the teams [52] [41] use SVM as their classifiers. Compared to SVM, KR is a simpler but less popular algorithm. However, our experiment shows that KR consistently outperforms SVM in MED scenario. Therefore, we use KR.

Fusion

Due to the computational cost, in justifying our choices of classifiers and fusion strategies, we only use 7 features (SIFT, CSIFT, MoSIFT, STIP, ASR, OCR, GIST) and conduct experiments on MED10 and MED11 development dataset with a total of 9822 videos. In this set of experiments, only the visual feature set has multiple features, while all other categories are represented by a single feature. By performing a visual feature (SIFT, CSIFT, MoSIFT, STIP, GIST) combination and an all-feature (SIFT, CSIFT, MoSIFT, STIP, ASR, OCR, GIST) combination, we have two combined features and seven single features (SIFT, CSIFT, MoSIFT, STIP, ASR, OCR, GIST). For late fusion, we use two rule-based fusion methods to combine the outputs of the above 9 classifiers. One is an average combination, another one is a weighted combination. The details of the weight calculation will be given in the experimental part.

In our second set of experiments, we use all the features except GIST as it does not contribute to the final performance. For double fusion, we fuse visual features (SIFT, CSIFT, TSIFT and MoSIFT), audio features (ASR, MFCC) and all the features as three early fusions and fuse them with the outputs of classifiers trained on individual features.

9.3.2 Datasets and system environment

We evaluate this work on the MED2010 and MED2011 datasets.

We ran our program on the Carnegie Mellon University Parallel Data Lab cluster, which contains 300 cores and it took us about 272140 CPU hours to extract features and run the BoW mapping.

9.3.3 Evaluation metrics

For historical reasons, we also use the following two evaluation metrics, namely Minimal Normalized Detection Cost (MinNDC) and Mean Maximm F1 (MMF1). As shown in Formula 1, NDC is a normalized weighted sum of miss detection probability P_{MD} and false positive rate P_{FA}. In our case, we use $C_{MD} = 80$ as the weight for miss detection and $C_{FA} = 1$ as the cost for false positive. MinNDC is the minmal value of NDC along the precision-recall curve. Lower MinNDC indicates better performance. These two criteria were the standard evaluation criteria for NIST in TRECVID 2010 and 2011.

$$NDC(S, E) = \frac{C_{MD} * P_{MD} * P_T + C_{FA} * P_{FA} * (1 - P_T)}{MINIMUM(C_{MD} * P_T, C_{FA} * (1 - P_T))} \tag{9.1}$$

9.3.4 Experimental settings

For both SVM and KR, we use a χ^2 kernel and use two-folder cross-validation to select the χ^2 parameter γ and SVM parameter C. γ controls the influence of kernel and C controls the bias variance trade-off of SVM.

The search ranges for both C and γ are 10^{-3} to 10^3, in multiples of 10. We did try small step size search for parameter selection suggested by [102], but could not find statistically significant differences.

Feature	MMNDC %± STD	MMF1%± STD
CSIFT	60.6 ± 0.7	52.5 ± 0.6
SIFT	**60.5 ± 1.4**	**53.3 ± 1.1**
MoSIFT	63.9 ± 1.4	50.6 ± 0.9
STIP	69.1 ± 0.5	48.2 ± 1.8
GIST	82.9 ± 1.5	33.7 ± 0.7
ASR	89.1 ±4.7	22.5 ± 4.1
OCR	85.7 ± 0.1	28.8 ± 0.8

Table 9.1: Comparison of single features on TRECVID MED2010. For MMNDC, lower score indicates better performance; for MMF1, higher score means better performance.

9.4 Results

To get statistically meaningful experiments, for all settings except MED 2011 submission, we repeat our experiments 10 times and calculate the mean and standard deviation of the results for each setting. Because running all the combinations of fusion strategies and classifiers will be computational expensive, we first compare all the classifiers, early fusion and late fusion strategies on MED 2010 and choose the best strategy for each step to perform further experiments on MED 2011.

9.4.1 Single feature comparison

First, we compare the mean MNDC (MMNDC) (lower MMNDC indicates better performance) and MMF1 (higher MMF1 indicates better performance) of single features on MED 2010. As shown in Table 9.1 and Fig. 9.2, the performance of different features varies dramatically from event to event. Generally, four local features including CSIFT, SIFT, MOSIFT and STIP consistently outperform other three features. In these four features, motion based features including MOSIFT and STIP get much better results than static features including SIFT and CSIFT in "Assembling a shelter" event, which has a lot of motion. Interestingly, static features are obviously superior to other features in "Batting a run" event and "Making a cake", because of their relatively monotonous background. Different characteristics and suitable situations for different features shows that our features are complementary to each other. Also, the performances of ASR and OCR features alone are much worse than visual features.

9.4.2 KR versus SVM

We further compared the performance of different classifiers by simply using SIFT, which is the single best feature. From Table 9.2, we can see that, compared to SVM, KR has lower MMNDC and higher MMF1, which indicates that KR is a better classifier for TRECVID MED tasks. From now on, we will use KR as our classifier except MED 2011 submission, in which we will use both KR and SVM classifiers.

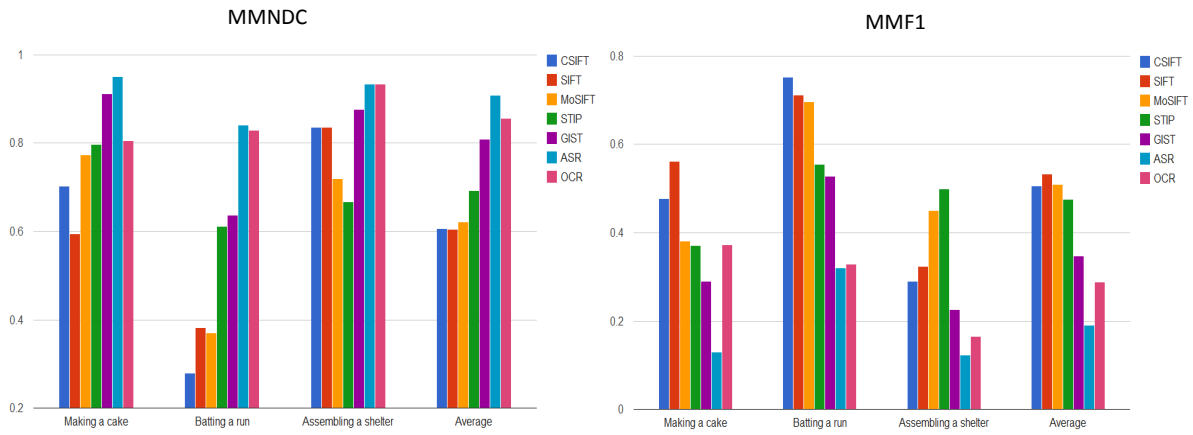


Figure 9.2: Comparison of single feature on TRECVID MED2010. For MMNDC, lower score indicates better performance; for MMF1, higher score means better performance.

	Classifiers		Early Fusion		Late Fusion	
	KR	SVM	MKL	Average Fusion	Weighted Fusion	Average Fusion
MMNDC% ± STD	**60.5 ± 1.4**	62.3 ± 1.1	**50.6 ± 0.8**	50.7 ± 0.6	**52.5 ± 1.5**	57.6 ± 1.9
MMF1% ± STD	**53.3 ± 1.1**	50.7 ± 2.9	**61.4 ± 0.1**	61.2 ± 0.6	**59.7 ± 1.1**	54.4 ± 1.6

Table 9.2: Comparison of classifiers, early fusion and late fusion strategies on TRECVID MED 2010. For MMNDC, lower score indicates better performance; for MMF1, higher score means better performance.

9.4.3 Early Fusion strategies comparison

For early fusion, we test both MKL and early fusion average fusion. The results are shown Table 9.2, in which we can see that MKL only gets comparable results to simple average fusion, this is consistent with what was suggested by [21]. Since MKL is much slower than average fusion, we will use average fusion for our further experiments.

9.4.4 Late Fusion strategies comparison

Table 9.2 shows the results of late fusion using weighted fusion and average fusion. The result of weighted late fusion is much better than the result of average late fusion. This indicates that different features have different contributions to the final results, especially when the performance varies dramatically between features. We will only use the weighted combination for late fusion for further comparison except our MED'11 submission.

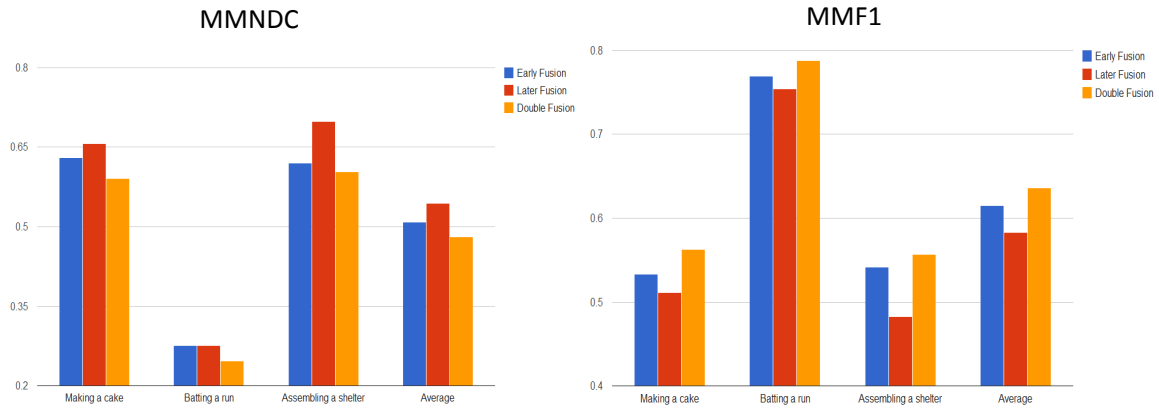


Figure 9.3: Comparison of double fusion with early fusion and late fusion on MED 2010. For MMNDC, lower score indicates better performance; for MMF1, higher score means better performance.

	MED 2010			MED 2011		
	Early Fusion	Late Fusion	Double Fusion	Early Fusion	Late Fusion	Double Fusion
MMNDC% $\pm$ STD	50.6 $\pm$ 0.8	52.5 $\pm$ 1.5	**48.9 $\pm$ 0.7**	65.6 $\pm$ 0.7	68.2 $\pm$ 1.3	**60.6 $\pm$ 0.8**
MMF1% $\pm$ STD	61.4 $\pm$ 0.1	59.7 $\pm$ 1.1	**62.9 $\pm$ 0.6**	41.1 $\pm$ 0.5	37.4 $\pm$ 3.8	**44.3 $\pm$ 0.9**

Table 9.3: Comparison of double fusion with early fusion and late fusion on MED2010. For MMNDC, lower score indicates better performance; for MMF1, higher score means better performance.

9.4.5 Double Fusion versus Early Fusion and Late Fusion

The result of double fusion is shown in Table 9.3 . From the table, we can see that double fusion gives much better results than both early and late fusion. Fig. 9.3 shows that double fusion consistently outperform early fusion and late fusion on all of three events in MED 2010. MED 2011 is much harder and more diverse than MED 2010 since we have 15 events now, but Fig. 9.4 and Fig. 9.5 indicate that double fusion still gets better performance than early fusion and late fusion on 11 of 15 events. For the other 4 events, double fusion gets comparable results to the best method and outperform the worse method. Therefore, double fusion does capture advantages of both early fusion and late fusion.

MED 2011 Submission

In our TRECVID MED 2011 submission [9], we used double fusion and achieved second best in terms of MMNDC. Again, from table 9.4, we can see that double fusion outperform both early fusion and late fusion.

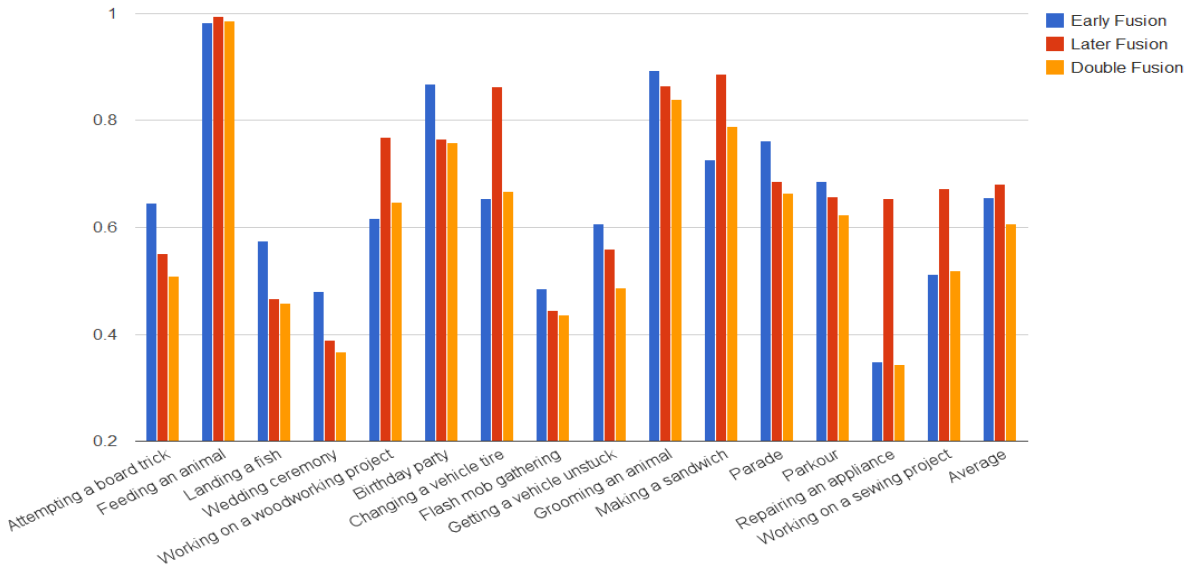


Figure 9.4: Comparison of double fusion with early fusion and late fusion on MED 2011 by suing MMNDC criteria. Lower MMNDC indicates better performances.

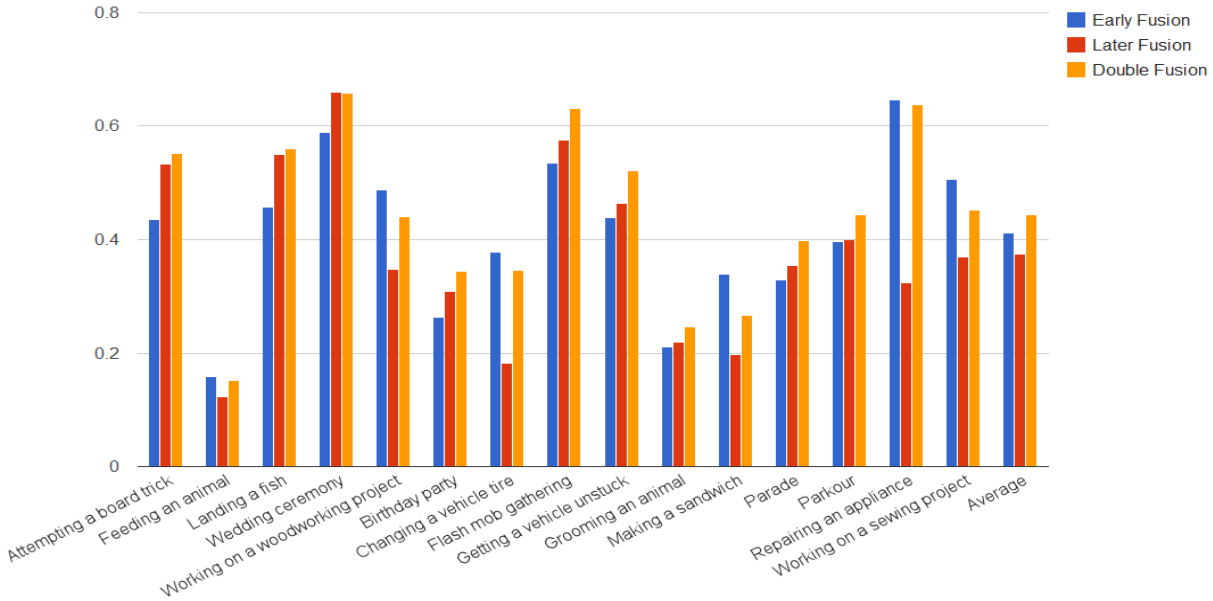


Figure 9.5: Comparison of double fusion with early fusion and late fusion on MED 2011 by using MMF1 criteria. Higher MMF1 indicates better performances.

	Early Fusion	Late Fusion	Double Fusion
SVM	63.2	52.8	51.9
KR	58.5	51.6	50.6

Table 9.4: Comparison of classifiers and fusion methods by using MED'11 dataset. MMNDC is used for evaluation, lower score indicates better performance.

9.5 Conclusions and Discussions

In this chapter, we present an analysis of early fusion and late fusion that combine features of different modalities in MED and introduce a double fusion scheme which combines early fusion and late fusion to incorporate their advantages. Our experiments on about 1684 hours of videos from TRECVID MED 2010 and 2011 showed that this simple strategy is very effective and had a substantial advantage over both early fusion and late fusion strategies. Moreover, we found that weighted combination is better than average combination for late fusion but performs similarly to average combination in early fusion. For historical reasons, the data and features we used in this analysis are relatively outdated. However, the lessons we learned from this analysis still holds for new datasets and features. And the techniques we used will be valuable in evaluating new features and method. We have been using double fusion for TRECVID MED submissions since 2010.

Chapter 10

Postprocessing: Multi-class Iterative Re-ranking (MIR)

10.1 Introduction

In our MED pipeline, we use a traditional "one-versus-all" classifiers, which fails to capture the relationships among multiple event classes. To address this problem, we introduce in this chapter a postprocessing step that applies to the classifiers' predictions. Parts of this chapter have been published in [61].

Our proposed method is inspired by curriculum learning which mimics the human learning scheme and has become popular for image and video classification [12, 17, 19, 49]. Curriculum learning [12, 60] suggests distinguishing easy and typical samples from difficult ones and treating them separately. Traditional curriculum learning methods [11, 12, 60] rely on human or extra resources to define data with difficulty levels (curriculum). Instead, we use freely-available classifiers from other action classes to define the curriculum. We then re-rank the prediction results to promote easy videos and suppress difficult ones. Our proposed method, called Multi-class Iterative Re-ranking (MIR), is easy to implement and training-free.

In the remainder of this chapter, We will first briefly introduce recent studies on capturing relationships among multiple action classes and curriculum learning. We then describe the details of our proposed method and demonstrate its performance gain over the baseline approach for the action recognition task represented by Hollywood2 and Olympic Sports datasets. Next, we report results for several MED tasks. A conclusion and discussion of future works are provided at the end.

10.2 Related Work

Given the encoded features, state-of-the-art methods often use "one versus the rest" SVM, which does not consider the relationships among action classes. To model those relationships, Bergamo & Torresani [13] suggested a meta-class method for identifying related image classes based on misclassification errors from a validation set. Hou *et al*. [37] identified similar class pairs and grouped them together to train "two versus the rest" classifiers. By combining "two versus

the rest" with "one versus the rest" classifiers, they observed significant improvements from baselines. Unlike the aforementioned approaches that require training and modify the predictions for one time only, MIR is training-free and iteratively updates the prediction rankings given those rankings from previous iterations.

Our MIR model is inspired by a new learning paradigm called curriculum learning, proposed by Bengio *et al*. [12]. Curriculum learning rates the difficulty levels for classifying samples and uses the rating as a 'curriculum' to guide learning. This new way of learning, from a human behavioral perspective, is considered similar to human learning in principle [55]. Moreover, just like school curriculum design in the everyday case, it relies on human or other data resources to define the curriculum. Our method instead uses freely available classifier predictions from other action classes to help define the curriculum without human intervention.

10.3 Action Recognition

We use Hollywood2 and Olympic Sports datasets for this evaluation. The reason we only use two instead of all five datasets is that MIR improves ranking rather than classification performance. That is to say, for HMDB51, UCF101 and UCF50 where the datasets where the MAcc is used for evaluation, MIR will not change their results.

We follow the experimental settings in [126]. More specifically, we use IDT features extracted using 15 frame tracking and camera motion stabilization. PCA is utilized to reduce the dimensionality of IDT descriptors by a factor of two. After reduction, the local descriptors are augmented with three-dimensional normalized location information [67]. FV encoding maps the raw descriptors into a Gaussian Mixture Model with 256 Gaussians trained from a set of 256000 randomly sampled data points. Classification in conducted by a 'one versus the rest' linear SVM classifier with a fixed $C = 100$ ([126]).

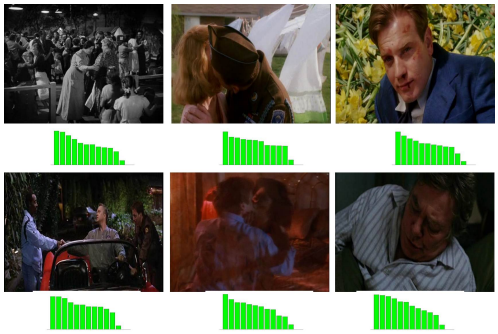
10.4 Multi-class Iterative Re-ranking (MIR)

As mentioned previously, traditional curriculum learning [12] often relies on human or extra data sources to define the curriculum: easy and typical samples versus difficult ones. In this chapter, we instead rely on classifiers of other classes, which are freely available, to define the curriculum. This new way of curriculum definition captures relationships among multiple human activity classes.

As depicted in Figure 10.1, we rank the videos from easy to difficult based on the classifiers' confidences. Obviously, easier videos have a much faster roll-off rate of sorted classification confidences. According to this ranking, we discover that videos that have some combinations of the following three scenarios will be more likely to be ranked on the difficult end of the scale:

- Contains noisy background motions. If the background of a video contains noisy motions and the target action has a small and weak signal, then the target action would likely be obscured. For example, the videos of HandShake actions in 10.1b are obscured by the background motions and much more difficult to detect than those videos of HandShake actions in 10.1a.

(a) **Easy Videos.** The example actions are: HandShake (left), Kiss (middle), HugPerson (right).



(b) **Difficult Videos.** The example actions are: HandShake (left), Kiss (middle), SitUp (right).

Figure 10.1: Illustration of easy and typical videos versus difficult ones. The bar charts show the sorted predictions of all the classifiers to the example videos. The predictions are normalized so that the values are between 0 and 1. As shown, the predictions of typical easy videos often have one or two dominant classes while those predictions of difficult videos often have much smoother score distributions.

- Contains multiple actions. If the subject performs multiple actions, the classifiers would be confused as the video features represent a mixture of the multiple actions. For example, the Kiss video in 10.1b contain both Kiss and Hug actions (appear in the following frames) while the Kiss examples in 10.1a only contain the Kiss action itself.
- Ill-defined actions. If the target action is ill-defined by itself, the video would be more difficult to classify. For example, 'SitUp' action is often encapsulated in the 'StandUp' action, which make it harder to classify.

Based on the difficulty scale, we design MIR to re-rank the predictions of classifiers. The intuition behind MIR is that if videos are more difficult to classify, then their predictions are less reliable and their rank should be lowered; if videos are easy and typical, then the predictions should be more reliable and the videos should be ranked higher. The algorithm to re-rank is easy to implement and fast to run. As shown in Algorithm 1, given a score matrix $P \in \mathbb{R}^{N \times K}$ that contains K classifiers' predictions on N videos. We update each score $P_{i,j}^{(w)}$ iteratively by looking at other classifiers' predictions on the same video and reducing the score using the predictions from the other classifiers. The reduction is carried out by first sorting other classifiers' predictions $\{P_{i,1}^{(w)}, P_{i,2}^{(w)}, \cdots, P_{i,K}^{(w)}\} \backslash P_{i,j}^{(w)}$ in a descending order and then subtracting the weighted sum of the sorted scores from $P_{i,j}$. We use exponentially decaying weights and the weighting coefficient β and the annealing parameter η have been set to 1 and 0.5, respectively, throughout this chapter. These values were determined experimentally. It improves more than 2% over the baseline method on both datasets. We also show the per-class comparison with and without MIR in Figure 10.3. We can see that for Hollywood2, MIR improves upon the baseline results for 11 out of 12 actions, and for Olympic Sports, MIR improves or gets similar results for 14 out of 16 actions. These per-class performance comparisons again show that the improvements from MIR are robust and significant. For those three situations that MIR did worse were the rare cases

Algorithm 1 Multi-class Iterative Re-ranking (MIR)

1: **Input:** The prediction scores of K class $P \in \mathbb{R}^{N\times K}$; Re-ranking annealing parameter η; Re-ranking weighting coefficient $\beta > 0$; Total iteration steps W.
2: **Init:** $P^{(0)} = P$
3: **for** $w = 1, 2, \cdots, W-1$ **do**
4: For any instance index $i \in \{1, 2, \cdots, N\}$ and class index $j \in \{1, 2, \cdots, K\}$,

$$\Delta_{i,j}^{(w)} = \text{sort}(\{P_{i,1}^{(w)}, P_{i,2}^{(w)}, \cdots, P_{i,K}^{(w)}\} \backslash P_{i,j}^{(w)}, \downarrow)$$

$$P_{i,j}^{(w+1)} = P_{i,j}^{(w)} - \eta^{w-1} \sum_{\substack{r=1 \\ r\neq j}}^{K} \mathrm{e}^{-\beta r} \Delta_{i,j}^{(w)}(r)$$

5: **end for**
6: **Output:** $P^{(W)}$

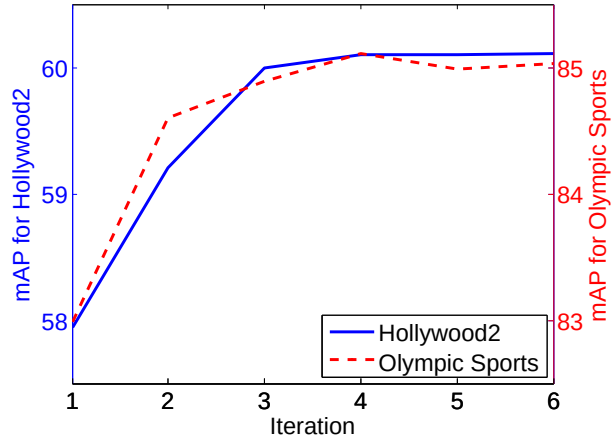


Figure 10.2: MIR performance versus number of iterations.

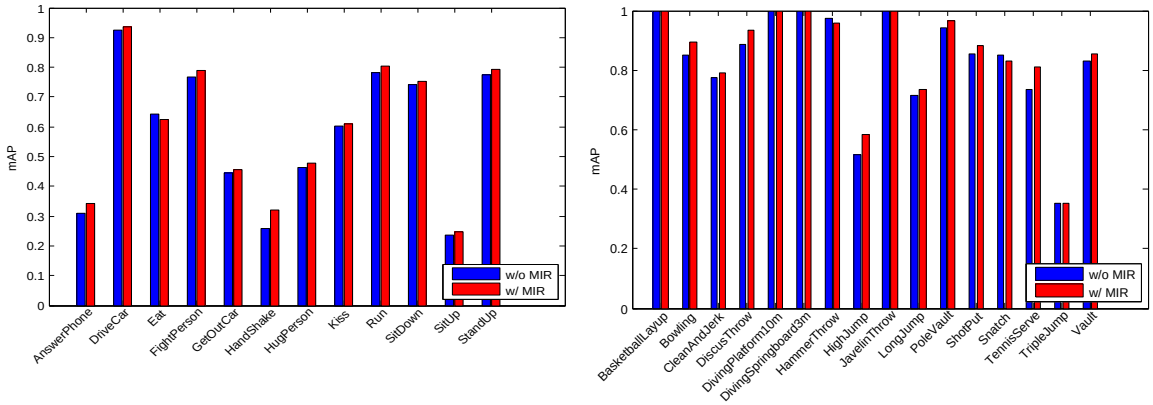
where the reranking causes the drop of the performance. As shown in Figure 10.2, MIR typically converges within 3 or 4 iterations.

10.5 Combined Results: MIFS, RNorm and MIR

10.5.1 Action Recognition

The data, feature encoding methods and classifiers are the same as the settings discussed in section 10.3.

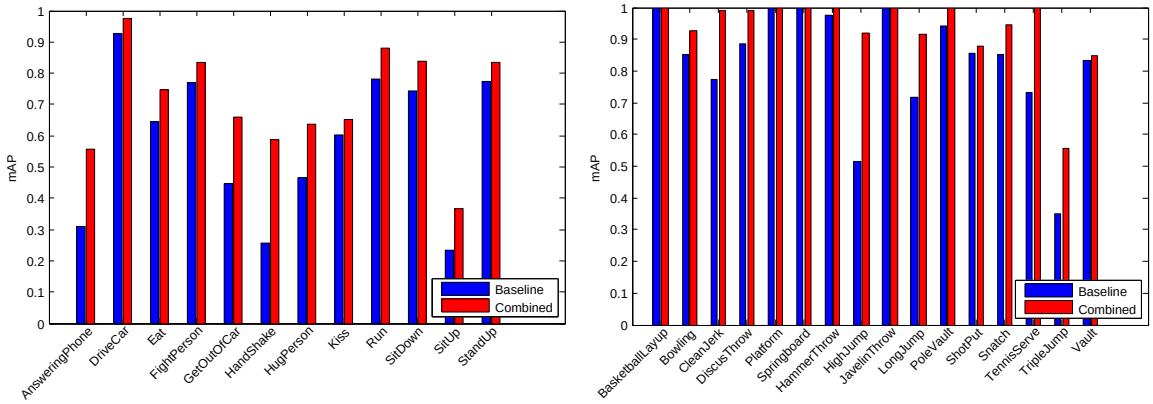
If we combine the proposed MIFS, RNorm and MIR methods, we observe an even more prominent improvement over the baseline method. We improve upon the baseline method by more than 13% and 10% absolute performance improvement on Hollywood2 and Olympic Sports datasets, respectively. A more detailed comparison is provided in Figure 10.4, from which we



(a) Hollywood2 (b) Olympic Sports

Figure 10.3: Per-class performance comparison of the baseline performances with and without MIR.

can see that our proposed methods together improve the baseline method on all the action classes in Hollywood2. For some of the hard classes like 'Hand Shake' and 'Answer Phone', we can get more than 20% absolute improvement. For Olympic Sports dataset, we observe a similar trend and get 9 out 16 classes with perfect predictions.



(a) Hollywood2 (b) Olympic Sports

Figure 10.4: Per-class performance comparison of our combined results to the baseline method. 'Combined' indicates applying MIF, RNorm and MIR to the baseline method.

10.5.2 Multimedia Event Detection

We test on MEDTEST2013 and MEDTEST2014 datasets in both EK100 and EK10 scenarios.

The feature encoding methods and classifiers are the same as the settings discussed in section 10.3.

	MEDTEST13		MEDTEST14	
	EK10	EK100	EK10	EK100
Baseline	17.0	33.6	12.0	26.2
MIR	16.7	34.2	11.3	26.6
Combined	**20.0**	**37.5**	**14.9**	**29.9**

Table 10.1: Performance Comparison on the MED task.

Table 10.1 lists the overall mAP on all four datasets. The baseline method is a conventional IDT representation. First, in EK10 scenario where the baseline performance is unusually low, MIR hurts the performance due to the inaccurate curriculum estimation; in EK100 setting that have comparatively reasonable baseline performance, MIR manages to achieve noticeable improvements, though not as much as the improvements in action recognition tasks where the baseline performances are much higher. These results reveal the fact that for MIR to be useful, the baseline performance should be reasonably accurate. Finally, the combined results of MIFS, RNorm and MIR, improve upon the baseline method by around 4% on EK100 tasks and about 3% on EK10 tasks. It is worth emphasizing that MED is such a challenging task that 3% of absolute performance improvement is significant.

10.6 Conclusions

This chapter introduces a postprocessing step for our MED system to capture the relationships among multiple event classes. MIR iteratively uses the predictions of other classifiers to rank videos from easy to difficult task levels and re-ranks the predictions accordingly. We also show that the combination of MIFS, RNorm and MIR significantly improve the performance of FV on the six real-world datasets we tested and set new state-of-the-art results for two benchmark action datasets including Hollywood2 and Olympic Sports.

Chapter 11

Resource Constrained Multimedia Event Detection

11.1 Introduction

In this chapter, we present a study comparing the cost and efficiency tradeoffs of multiple features for MED. Parts of this chapter have been published in [65, 66].

Features are a critical part of contemporary multimedia and computer vision research. However, their efficacy has not been systematically studied. In this chapter, we evaluate the accuracy and contribution of more than 10 features from different modalities. Contrasting multiple performance metrics including MAP, MinNDC, and $P_{MD}@TER = 12.5$, our study balances the trade-off between accuracy and computational cost. This study provides empirical results for selecting feature sets that are capable for dealing with large-scale data using limited computational resources. Our methods can also be applied to other resource limited multimedia analysis such as selecting/fusing multiple classifiers and different representations of each feature set.

The remaining chapter are organized as follows. We discuss related work in section 11.2 and we elaborate our MED system including features and evaluation metrics in section 11.3. In section 11.4, we discuss experimental results. Finally, we summarize this chapter in section 11.5.

11.2 Related Work

Previous work [84] [116] [26] [77] on MED feature evaluation can be divided in two main categories whether they rely on low-level features or high-level semantic concepts. Yang *et al.* [134] and Tamrakar *et al.* [116] evaluated the individual performance of different low-level visual features (SIFT, STIP, Trajectories...) and their combinations. Meler *et al.* [84], Ebadollahi *et al.* [26] and Liu *et al.* [77] evaluated high-level features' performance on MED. In this work, we include both low-level and high-level features, leading to a more complete comparison. Moreover, most previous work only evaluated each feature's single performance, while we focus more on each feature's contribution to the combined system. We show that the single feature performance, albeit important, does not necessarily reflect its contribution to the overall performance.

In terms of efficiency, most previous works focused on improving one component of a recognition system with faster algorithms. For example, Bay *et al.* [10] introduced SURF as a faster alternative of SIFT. Moosmann *et al.* [86] proposed to use random forest to replace SVM. Jiang [50] conducted an interesting study to evaluate and combine a number of speed-up strategies to get a fast event recognition system. Different from previous work, we offer a resource constrained solution that can be customized by users who have different resource constraints.

11.3 MED System

Given a set of training and testing videos, we first extract features in different modalities from the videos and then train a χ^2 SVM classifier for each feature. Average late fusion is used to combine the prediction results from each feature.

11.3.1 Features

To build a good MED system, it is important to have features that capture various aspects of an event. In our MED system, we explore five different feature modalities which are computed from different sources. Image features capturing appearance information are computed from key-frames. Video features are extracted from videos directly and collect motion information. Audio features characterize acoustic information. Text features and semantic features can borrow domain knowledge from other datasets such as Flickr and give semantically meaningful representations for events.

Image Features: We use three image features that are computed from the keyframes extracted as described in [64]. The three images feature are SIFT, CSIFT, and TCH [119].

After detecting key points using harris-laplace key point detectors from key frames, we use three different feature descriptors to generate SIFT, CSIFT and TCH features, which hopefully are complementary. From the key points' descriptors, a k-means algorithm generates a codebook which has 4096 words for each feature. Next, a soft-mapping strategy, in which we choose the ten nearest clusters and assign a rank weight ($\frac{1}{rank}$) to them, maps key points into the codebook. Spatial pyramid matching as described in [63] compensates for spatial information lost in the bag-of-words representation. We then aggregate the image representation into video representations by averaging all the image representations in one video and normalize the video representation using an L2 normalization.

Video Features: We have three visual video features, namely IDT [126], MoSIFT [16] and STIP [69], which are computed directly from videos. MoSIFT, as a three dimensional extension of SIFT features, uses a Difference of Gaussian (DoG) based detector and is represented by a descriptor combining SIFT and HOF. STIP uses 3D Harris corner detectors and its interest points are represented as the combination of HOG and HOF. After getting the key point descriptors, the same bag-of words and spatial pyramid matching as with image features is adopted to cast the key point representation into a video-level representation.

Audio Features: Audio features are another important resource to detect events in videos. To represent general audio information, we use the Mel-frequency cepstral coefficients (MFCCs) feature. We compute 20 dimensional MFCCs for every 10ms over a 32ms sliding window. Given

the raw features, we compute a 4096 word codebook and aggregate all MFCC features from one video into a 4096 dimensional bag-of-words representation. In addition to MFCC, we also use have ASR features as described in [9] to capture semantic information in audio.

Semantic Features: In our MED system, three semantic features are used. The first one called SIN346 is defined by the TRECVID SIN track. This feature has 346 dimensions representing the 346 concepts in SIN [9]. The second one is the Object Bank feature (ObjBank) introduced by Li et.al. [74], in which we extended the original 176 objects to 1000 objects by using the Imagenet challenge 2012 dataset (ILSVRC2012) [58]. Another semantic feature that is also trained on the ILSVRC2012 dataset is the DCNN feature, in which we trained a Deep Convolutional Neural Network feature using the method introduced by Krizhevsky *et al*. [58] on a NVIDIA Tesla K20m GPU.

Text Features: Following Bao *et al*. [9], we also use Optical Character Recognition (OCR) features to represent the text feature. We use a commercial OCR system is used to recognize the text. As OCR rarely gets a complete word correct, we treat each trigram of characters as a token instead of each whole word as a token. The details can be found in Chapter 9

11.4 Experiments

We evaluate on both MEDTEST2013 and KINDREDTEST datasets. To extract features, we use the PSC blacklight [1] machine, which is a SGI UV 1000cc-NUMA shared-memory system comprising 256 blades. Each blade holds 2 Intel Xeon X7560 (Nehalem) eight-core 2.27 GHz processors. For the GPU, we use a NVIDA Tesla-K20, which has 2496 cores. All GPU measurment are relative to on of these GPUs.

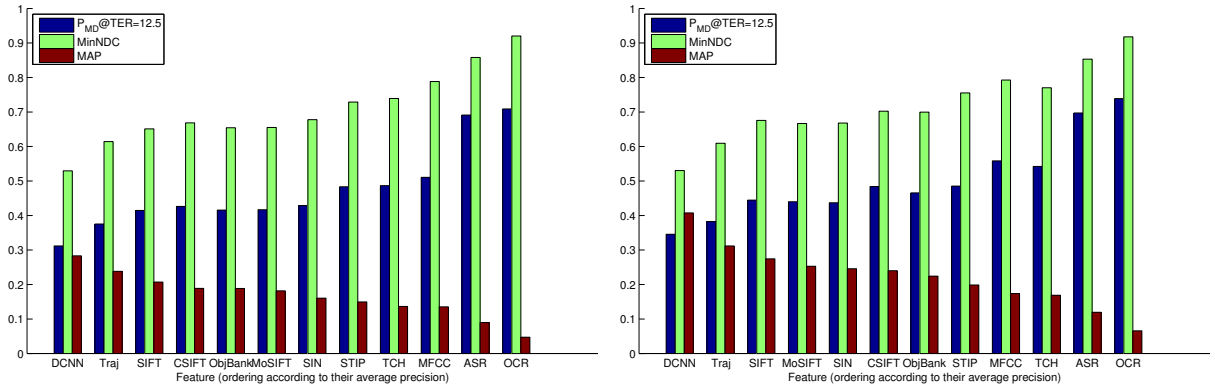
For historical reasons, besides MAP, we also use two other evaluation metrics, namely Minimal Normalized Detection Cost (MinNDC) and $P_{MD}@TER = 12.5$. As shown in Formula 1, NDC is a normalized weighted sum of miss detection probability P_{MD} and false positive rate P_{FA}. In our case, we use $C_{MD} = 80$ as the weight for miss detection and $C_{FA} = 1$ as the cost for false positive. MinNDC is the minmal value of NDC along the precision-recall curve. Lower MinNDC indicates better performance. Another metric related to MinNDC is $P_{MD}@TER = 12.5$, in which $TER = \frac{P_{MD}}{P_{FA}}$ is an indicator of the miss detection rate at a given threshold. These two criteria were the standard evaluation criteria for NIST in TRECVID 2010 and 2011.

$$NDC(S, E) = \frac{C_{MD} * P_{MD} * P_T + C_{FA} * P_{FA} * (1 - P_T)}{MINIMUM(C_{MD} * P_T, C_{FA} * (1 - P_T))} \tag{11.1}$$

11.4.1 Single feature performance and contribution

We study both single and combined features' performance using the three evaluation metrics described in Section 3.2.

Fig. 11.1 shows the single feature accuracies on our MEDTEST2013 and KINDREDTEST sets. We order the features according to their MAPs. From Fig. 11.1, we can see that the ranks are quite consist across different metrics and datasets. Specifically, the top two features that

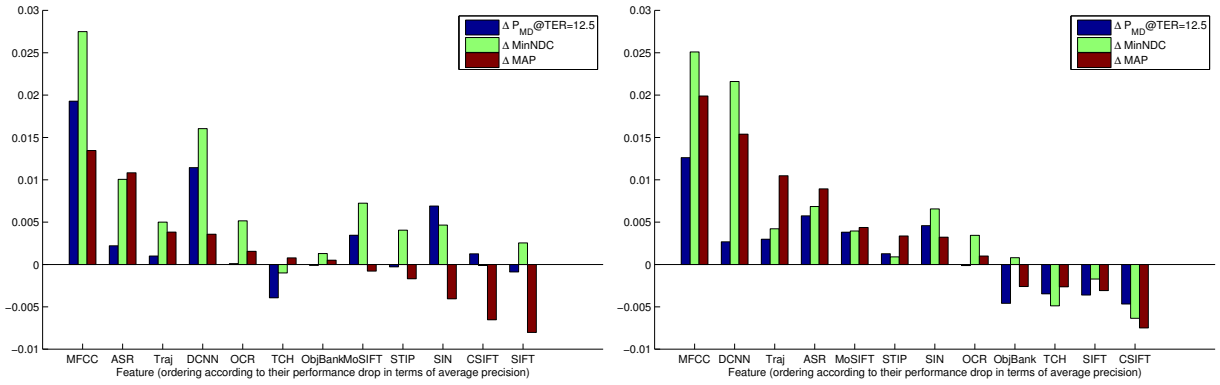


(a) MEDTEST2013 (b) KINDREDTEST

Figure 11.1: Single feature accuracy for both datasets, ranked according to MAP. Lower score corresponds to better performance for MinNDC and $P_{MD}@TER = 12.5$, but higher is better for MAP.

significantly outperform other features are DCNN and Traj; the two features that are worse than other features are ASR and OCR; others have very similar performances and their ranks vary due to their minor performance differences. It is interesting to see that DCNN, a high-level feature, can significantly outperform low-level features. Also, our OCR has higher recognition accuracy than ObjBank and SIN, yet its overall performance is the worst among all eleven features. The reason is that most of video don't contain recognizable text. The significant performance difference between visual and audio features shows that, in unconstrained, visual information is more discriminative than audio information.

To determine the contribution of each feature, we conduct an ablation study. In this study, we first get the results of combining all features, then we remove one feature from the set and recalculate the results. Fig. 11.2 shows the performance drop (leave-one-feature-out accuracy) from removing one feature at a time. This drop shows the importance of each feature to the overall combined feature set. The ranks by performance drop is quite different from the ranks of single feature performance. In this case, higher values are better for all performance measure. The higher the value, the greater the contribution to the overall system. For example, MFCC has a low rank as a single feature accuracy but rank highest in leave-one-feature-out performance. This indicates that MFCC is very complementary to the other features. While SIFT and CSIFT, align with most other features, they combine to reduce MAP because they reduce the overall weight per feature while not contributing additional information in the average late fusion method. More sophisticated fusion methods such as fusion by learning combination weights may be able to avoid this problem but will inevitably give smaller weights to those redundant features. Fig. 11.3 shows the Spearman rank coefficients for all of the features: it indicates MFCC and ASR are very different from the other features. The figure demonstrate how MFCC and ASR are orthogonal to the other features. Although the Spearman rank coefficients also show OCR is very different from the other features, its close to random individual performance indicates its negligible role in the system. Fig. 11.2 demonstrates that single feature accuracy alone does not indicate suitability for inclusion in the combined feature set. However, as long as the leave-one-feature-out accuracy is not negative, inclusion will increase the overall score. Unfortunately, including all features with



(a) MEDTEST2013

(b) KINDREDTEST

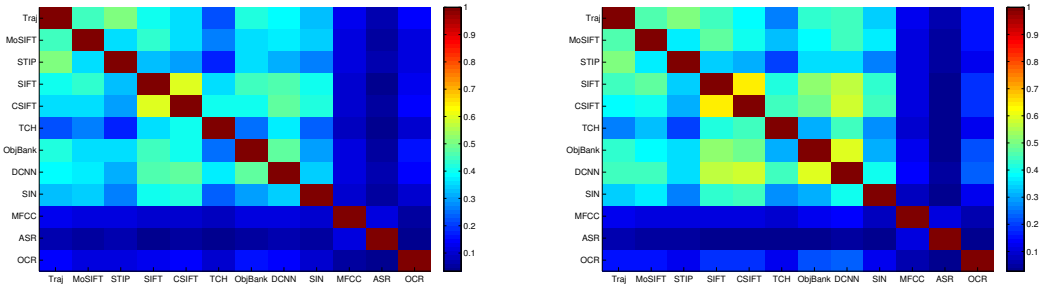
Figure 11.2: Leave-one-out Accuracy for MED, Ranked According to ΔMAP. In all three metrics, higher values means higher *performance drop* when we leave the feature out, hence a higher contribution of the feature to the combined system.

a positive value in Fig. 11.2 will lead to a computationally expensive system that is generally unsuitable for a real world applications.

11.4.2 Performance versus cost trade-off

In order to determine the performance versus cost trade-off, we first determine each feature's computational cost as shown in Table 11.1, which also shows the abbreviation of features for later usage. For each feature, the time is the number of hours to process one hour of video. Let's assume our goal is to process one hour of video in one hour. We then determine, for a given number of CPUs, what the best possible performance is by a brute-force search across all features in Table 11.1. Fig. 11.4 shows the best possible performance given one hour of processing time for the given number of CPUs for all three metrics without using a GPU, thus excluding the DCNN feature. Tables 11.2 and 11.3 show the optimal feature sets for the given number of CPUs. Fig. 11.5 shows the best possible performance given one hour of processing time for the given number of CPUs for all three metrics using one GPU, including the DCNN feature. Tables 11.4 and 11.5 give the optimal feature sets for the given number of CPUs. As we can see from Tables 11.2 to 11.5, the MFCC feature appears in almost all configurations due to its low computational cost (Table 11.1) and relatively high contribution (Fig. 11.2). Although Traj has a high contribution, it does not show up in Tables 11.2 to 11.5 until we have a minimum of 16 CPUs due to its high computational cost. We can see from the tables that the optimal feature sets are very similar for the MEDTEST2013 and the KINDREDTEST, which demonstrates that it is possible to select the optimal feature set from a smaller dataset like KINDREDEST and apply it to a larger dataset like MEDTEST2013. Likewise, these optimal feature sets are fairly similar across the three metrics. Further, we can also see from the figures that we get a diminishing return beyond 32 CPUs. In all cases, we can get more than 92 percent of the best performance by just using 32 cores.

Comparing Fig. 11.2 and Tables 11.4 and 11.5, we can see that in listing the importance



(a) MEDTEST2013 (b) KINDREDTEST

Figure 11.3: Spearman's rank correlation coefficient for features.

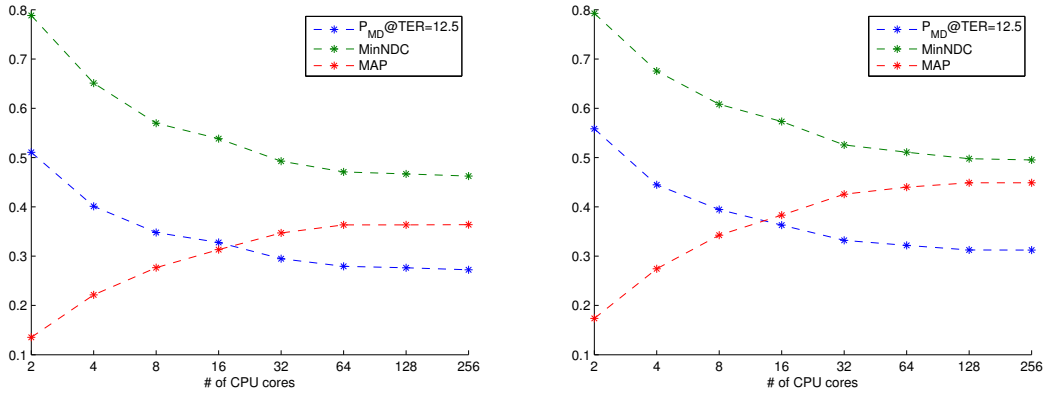
Table 11.1: Computational cost for features.

Features (Abbrev.)	core hours	features (Abbrev.)	core hours
Traj(Tr)	12.38	Objbank(Ob)	28.43
MoSIFT(Mo)	11.23	DCNN(DC)	0.15 GPU
STIP(ST)	10.33	SIN(SIN)	78.92
SIFT(SI)	3.57	MFCC(MF)	1.36
CSIFT(CS)	5.05	ASR(AS)	4.99
TCH(TC)	2.12	OCR(OCR)	1.34

of features, where importance in the table is measured by the ratio of the number of times the feature occurs to the number of possible occurrence given timing constraints, leave-one-feature-out performance is consistent with brute-force search results, hence very predictive in selecting the right feature set. For example, MFCC, DCNN, Traj and ASR, as the top 4 contributing features appear in almost all of the configurations as long as we have enough computational resources in terms of MAP. For other metrics, we have the same basic observation. The cost of computing the leave-one-feature-out accuracy is relatively inexpensive for late fusion, as all the components are already computed. In our system with 12 features, leave-one-feature-out accuracy computation is about 300 times faster than brute-force search.

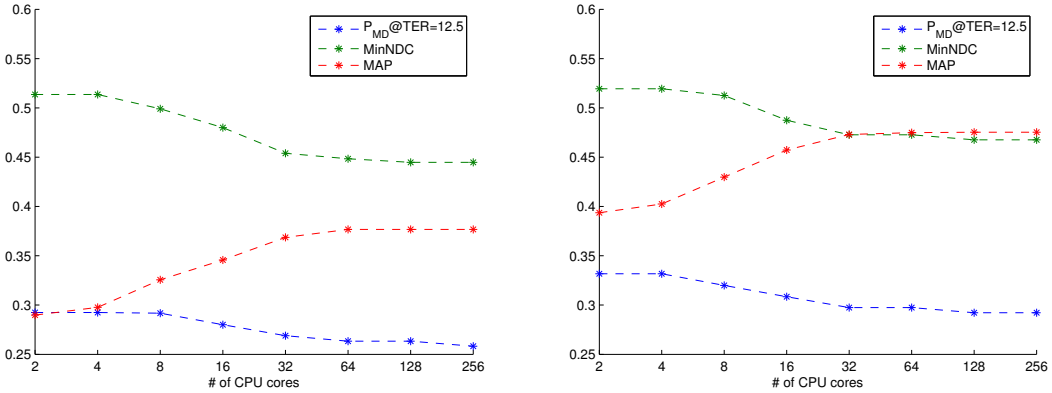
Table 11.2: Resource specific feature sets for MEDTEST2013.

CPUs	Optimal Sets in Real-time Performance		
	MinNDC	$P_{MD}@TER = 12.5$	MAP
2	MF	MF	MF
4	TC MF	SI	TC MF
8	TC SI MF	TC SI MF	TC SI MF
16	Tr TC MF	CS SI AS MF	Tr TC MF
32	Mo TC CS SI AS MF	Mo TC CS SI OCR AS MF	Tr TC SI OCR AS MF
64	Ob ST Mo TC CS AS MF	Ob Mo Tr SI OCR AS MF	Ob Mo Tr TC OCR AS MF
128	Ob ST Mo TC CS OCR AS MF	Ob Mo Tr CS SI OCR AS MF	Ob Mo Tr TC OCR AS MF
256	SIN Ob Mo Tr CS SI OCR AS MF	SIN Ob Mo Tr SI OCR AS MF	SIN Ob Mo Tr TC OCR AS MF



(a) MEDTEST2013 (b) KINDREDTEST

Figure 11.4: Resource specific performance for MED (without DCNN).



(a) MEDTEST2013 (b) KINDREDTEST

Figure 11.5: Resource specific performance for MED (with DCNN on a GPU).

Table 11.3: Resource specific feature sets for KINDREDTEST.

CPUs	Optimal Sets in Real-time Performance		
	MinNDC	$P_{MD}@TER = 12.5$	MAP
2	MF	MF	MF
4	SI	SI	SI
8	TC SI MF	TC SI MF	TC SI MF
16	Tr MF	Tr TC MF	Tr TC MF
32	ST Mo SI AS MF	Tr CS SI OCR AS MF	Tr TC SI AS MF
64	ST Mo Tr SI OCR AS MF	Ob Mo Tr SI OCR AS MF	ST Mo Tr TC SI OCR AS MF
128	SIN ST Mo Tr SI OCR AS MF	SIN ST Mo Tr SI OCR AS MF	SIN Mo Tr TC OCR AS MF
256	SIN Ob ST Mo Tr CS SI OCR AS MF	SIN Ob ST Mo Tr SI OCR AS MF	SIN Mo Tr TC OCR AS MF

11.5 Conclusions and Discussions

In this chapter, we systematically evaluated the performance and contributions of more than 10 multi-modality features for MED. Based on the evaluation and computational cost of feature

Table 11.4: Resource specific feature sets for MEDTEST2013 (with 1 additional GPU).

CPUs	Optimal Sets in Real-time Performance		
	MinNDC	$P_{MD}@TER = 12.5$	MAP
2	DC MF	DC MF	DC MF
4	DC MF	DC MF	DC TC MF
8	DC SI MF	DC SI MF	DC SI MF
16	DC Tr MF	DC TC SI OCR AS MF	DC Tr TC MF
32	DC Tr SI AS MF	DC Mo TC SI OCR AS MF	DC Tr CS OCR AS MF
64	DC Ob ST Mo TC SI AS MF	DC ST Mo Tr TC CS SI OCR AS MF	DC Ob Mo Tr TC OCR AS MF
128	SIN DC ST Mo TC SI OCR AS MF	SIN DC ST Mo SI OCR AS MF	DC Ob Mo Tr TC OCR AS MF
256	SIN DC Ob Mo Tr AS MF	SIN DC ST Mo SI OCR AS MF	DC Ob Mo Tr TC OCR AS MF

Table 11.5: Resource specific feature sets for KINDREDTEST(with 1 additional GPU).

CPUs	Optimal Sets in Real-time Performance		
	MinNDC	$P_{MD}@TER = 12.5$	MAP
2	DC MF	DC MF	DC MF
4	DC MF	DC MF	DC TC MF
8	DC AS MF	DC AS MF	DC SI MF
16	DC Tr MF	DC Tr MF	DC Tr MF
32	DC Tr AS MF	DC Mo Tr OCR AS MF	DC Tr SI AS MF
64	DC Tr AS MF	DC Mo Tr OCR AS MF	DC Mo Tr SI AS MF
128	SIN DC ST Mo Tr SI OCR AS MF	SIN DC Mo Tr OCR AS MF	SIN DC Mo Tr TC OCR AS MF
256	SIN DC ST Mo Tr SI OCR AS MF	SIN DC Mo Tr OCR AS MF	SIN DC Mo Tr TC OCR AS MF

extraction, we select feature sets that have optimal real-time performance under various resource constraints by measuring leave-one-feature-out performance and brute-force search performance.

A particularly important insight from above experiments is that leave-one-feature-out feature performance is very predictive in selecting the right feature set.

We also found that in both datasets and across the three different metrics:

- DCNN and Trajectory features are very useful features in unconstrained video analysis. Especially DCNN, given its semantic and high accuracy characteristics, is a feature that is worth paying a lot of attention to.
- Even a less accurate feature such as MFCC, if it is cheap and complementary to other features, can be very useful.
- By selecting the right features, we can save a large amount of computational cost with a minimum accuracy drop. For example, in our experiments, we can still achieve 92% of optimal performance by using a system that only requires 13% of the original computational cost.

By using the results of this analysis, researchers can use it as a guide to tailor a system based on their resource constraints.

Chapter 12

Conclusions and Future Work

In this thesis, we study the problem of multimedia event detection (MED) in web videos. By introducing a series of improvements on the classic video classification pipeline, we have been able to realize an accurate and customizable MED system. Some of these proposed methods have been incorporated as major components of the CMU E-lamp system, which has achieved the leading performances in the TRECVID Multimedia Event Detection (MED) competition 2011 $\sim$ 2015.

Besides an applicable system, our research also provides a set of insights that we believe would be helpful for the future MED study. These insights are:

- Different from spatial scale invariance, Gaussian smoothing should not be applied to temporal scale invariance (MIFS).
- Due to the heavy redundancy of video frames, we can take single frame/shots as inputs to train CNNs. However, this type of training is suboptimal because the label describes the entire video taken as a whole but not necessarily each frame/clip (DOVF).
- In representing videos, we do not need to represent every frame (DOVF).
- Handcrafted features share a similar process as CNNs (ConvISA).
- Performance tends to improve when we design different network structures for different kinds of data such as optical flows and pixels (ConvISA).
- Space (and time) encoding in video classification is not as useful as space encoding in image classification because of the much larger diversity in video content (STED).
- In space-time information encoding, instead of using spatial pyramid matching, a better way is to extend features with spatio-temporal location information, which can dramatically reduce the dimensionality of the final features and keep the performance the same (STED).
- Replacing the original FV values with their instance-wise ranking can fundamentally address the sparse and bursty distribution problem of FV and lead to significant performance improvements (RNorm).
- Combining both early fusion and late fusion together can do better than either one of them alone, especially in those cases where late fusion outperform early fusion (Double Fusion).

- Incorporating the information from other events into event classification can significantly improve the performance of the MED system (MIR).
- In choosing the right feature to incorporate into the system, an ablation study can be much more informative than looking at each feature independently.

Given the complexity of our task and the significance of our improvements, we believe that our algorithms and the above lessons could be generalized to other tasks and better methods. For example, our double fusion methods can be used on tasks such as image segmentation [78], text retrieval [138], and action recognition in Inferred data [32], etc. It can also be extended to neural network scenario [131]. Similarly, our MIFS algorithm has been extended to deep learning scenario [135].

In the near future, a promising direction is to improve the speed of two-stream CNNs. Currently, we rely on traditional local optical flow estimation methods to compute motion information for the CNNs. This two-stage pipeline is sub-optimal for the following reasons:

- The pre-computation of optical flow is time consuming and storage demanding compared to the CNN step. Even extracted on GPUs, optical flow calculation has been the major speed bottleneck of the current two-stream approaches, which learn to encode appearance and motion information in two separate CNNs.
- Traditional optical flow estimation is completely independent of action recognition and is prone to error. Because it is not end-to-end trainable, we cannot extract motion information that is optimal for the tasks.

The primary goal, therefore, is to move toward end-to-end learning by incorporating the optical flow estimation into the CNN framework. I hope that, by taking consecutive video frames as inputs, CNNs learn the temporal relationships among pixels and use the relationships to predict action classes.

Another very promising direction to do joint multi-stream learning. Currently, we train two-stream CNN separately. Can we train them jointly? Can we do multiple stream training? For example, adding audio information into the joint training? It would be interesting if we can provide insights for these questions.

The above enhancements, if successful, will great improve the performance (speed and accuracy) of MED and general video classification.

Appendix A

Proof

Here we give the details of proofs in the main text. Our proofs are based on the following Bernstein's Matrix Inequaltiy.

Lemma 1 (Bernstein's Matrix Inequality) *Let* $\mathbf{x}_i \in \mathbb{R}^{p\times 1}$, $\|\mathbf{x}_i\|^2 \leq B$. $S = \mathbf{x}_1{\mathbf{x}_1}^{\mathrm{T}} + \cdots {\mathbf{x}_n}{\mathbf{x}_n}^{\mathrm{T}}$. *Then with probability at least* $1-\delta$,

$$\|S - \mathrm{E}\{S\}\| \leq \sqrt{2B\,\|\mathrm{E}\{S\}\|\log(2p/\delta)} + \frac{B}{3}\log(2p/\delta)\,. \tag{A.1}$$

A.0.1 Proof of Theorem 1

For the i-th row, j-th column of P,

$$\begin{aligned}
|P_{i,j}| =& [\boldsymbol{\alpha}_i(t_j+\tau) - \boldsymbol{\alpha}_i(t_j)] \leq 2 && \text{(A.2)}\\
\|P_j\|^2 \leq& 4k && \text{(A.3)}\\
\mathrm{E}\{P_{i,j}^2\} \leq& 2(1+c)\exp(-\gamma/\tau) && \text{(A.4)}\\
\mathrm{E}\{P_{i,j}^2\} \geq& 2\exp(-\gamma/\tau) && \text{(A.5)}\\
\mathrm{E}\{P_{i,j}P_{k,j}\} =& 0 \quad i \neq k && \text{(A.6)}\\
\lambda_{\max}\{\mathrm{E}\{P_j{P_j}^{\mathrm{T}}\}\} =& \frac{1}{T}\lambda_{\max}\{\mathrm{E}\{PP^{\mathrm{T}}\}\} \leq 2(1+c)\exp(-\gamma_1/\tau) && \text{(A.7)}\\
\lambda_{\min}\{\mathrm{E}\{P_j{P_j}^{\mathrm{T}}\}\} =& \frac{1}{T}\lambda_{\min}\{\mathrm{E}\{PP^{\mathrm{T}}\}\} \geq 2\exp(-\gamma_k/\tau)\,. && \text{(A.8)}
\end{aligned}$$

By Bernstein's Matrix inequality (Theorem 1), with probability at least $1-\delta$, we have

$$\begin{aligned}
4T\Delta_\tau \triangleq \left\|PP^{\mathrm{T}} - \mathrm{E}\{PP^{\mathrm{T}}\}\right\| \leq& \sqrt{2\times 4k\times 2T(1+c)\exp(-\gamma_1/\tau)\log(2k/\delta)} + \frac{4k}{3}\log(2k/\delta)\\
=& 4\sqrt{kT(1+c)\exp(-\gamma_1/\tau)\log(2k/\delta)} + \frac{4}{3}k\log(2k/\delta)\\
\leq& 4\sqrt{kT(1+c)\log(2k/\delta)} + \frac{4}{3}k\log(2k/\delta) && \text{(A.9)}
\end{aligned}$$

When

$$T \geq \frac{1}{9(1+c)}k\log(2k/\delta) \tag{A.10}$$

we have

$$\Delta_\tau \leq 2\sqrt{k\frac{1}{T}(1+c)\log(2k/\delta)} \tag{A.11}$$

Therefore, when T large enough,

$$\beta(PP^{\mathrm{T}}) \leq \frac{(1+c)\exp(-\gamma_1/\tau)+\Delta_\tau}{\exp(-\gamma_k/\tau)-\Delta_\tau}\ . \tag{A.12}$$

A lower bound on $\beta(PP^{\mathrm{T}})$ could be given similarly by changing Δ_τ to $-\Delta_\tau$.

A.0.2 Proof of Theorem 2

The proof is similar to Theorem 1, except that P_i is sampled from m different distribution. To borrow the proof in Theorem 1, the distribution of P_i has m components. The i-th component is sampled from $i\tau$ skip with probability $T_i/\sum_j T_j = T_i/T$ where $T = \sum_j T_j$ is the total number of features. Based on this observation, we have:

$$|P_{i,j}| \leq 2 \tag{A.13}$$

$$\|P_j\|^2 \leq 4k \tag{A.14}$$

$$\mathrm{E}\{P_{i,j}^2\} \leq \sum_i \frac{T_i}{T} 2(1+c)\exp(-\gamma/\tau_i) \tag{A.15}$$

$$\mathrm{E}\{P_{i,j}^2\} \geq \sum_i \frac{T_i}{T} 2\exp(-\gamma/\tau_i) \tag{A.16}$$

$$\mathrm{E}\{P_{i,j}P_{k,j}\} = 0 \quad i \neq k \tag{A.17}$$

$$\lambda_{\max}\{\mathrm{E}\{P_j{P_j}^{\mathrm{T}}\}\} = \frac{1}{T}\lambda_{\max}\{\mathrm{E}\{PP^{\mathrm{T}}\}\} \leq \sum_i \frac{T_i}{T} 2(1+c)\exp(-\gamma_1/\tau_i) \tag{A.18}$$

$$\lambda_{\min}\{\mathrm{E}\{P_j{P_j}^{\mathrm{T}}\}\} = \frac{1}{T}\lambda_{\min}\{\mathrm{E}\{PP^{\mathrm{T}}\}\} \geq \sum_i \frac{T_i}{T} 2\exp(-\gamma_k/\tau_i) \tag{A.19}$$

From matrix concentration,

$$\begin{aligned} 4T\Delta_\tau \triangleq \left\|PP^{\mathrm{T}} - \mathrm{E}\{PP^{\mathrm{T}}\}\right\| &\leq \sqrt{2\times 4k\times[\sum_i T_i 2(1+c)\exp(-\gamma/\tau_i)]\log(2k/\delta)} + \frac{4k}{3}\log(2k/\delta) \\ &= 4\sqrt{k[\sum_i T_i(1+c)\exp(-\gamma/\tau_i)]\log(2k/\delta)} + \frac{4}{3}k\log(2k/\delta) \\ &\leq 4\sqrt{k(1+c)T\log(2k/\delta)} + \frac{4}{3}k\log(2k/\delta) \end{aligned} \tag{A.20}$$

Appendix B

Detailed Results

Table B.1 B.2 B.3 B.4 B.5 B.6 B.7 show the detailed MIFS results of seven datasets we use in this paper. We mask the best result of each class in bold and calculate the number of classes that have the best results for each method. It is interesting to observe, from action recognition datasets, that similar patterns as in each dataset still hold within each class. That is to say, different actions can have very different scale requirements; the performances of coarse scale features are limited by the difficulty of optical flow calculation and tracking. Our future work will focus on selecting the right scale for each action type and improving the quality of optical flow calcuation and tracking at coase scale. For MED results, we can see that MIFS does not only perform significantly better over the baseline method in terms of mAP, but also dominates in terms of number of event that performs better.

Table B.1: Detailed results of HMDB51 for MIFS with different scale levels. Accuracy (%) is used for evaluation. The best accuracy is marked in bold.

Action Name	L=0	L=1	L=2	L=3	L=4	L=5
BrushHair	92.22	**94.44**	92.22	92.22	93.33	93.33
Cartwheel	56.67	63.33	**64.44**	61.11	63.33	63.33
Catch	78.89	**81.11**	80.00	**81.11**	77.78	78.89
Chew	70.00	71.11	72.22	74.44	**80.00**	74.44
Clap	**85.56**	82.22	77.78	77.78	73.33	78.89
Climb	82.22	86.67	91.11	90.00	**92.22**	91.11
ClimbStairs	50.00	**56.67**	**56.67**	53.33	54.44	53.33
Dive	60.00	**62.22**	56.67	57.78	60.00	56.67
DrawSword	54.44	55.56	55.56	55.56	**57.78**	**57.78**
Dribble	85.56	**86.67**	**86.67**	85.56	84.44	84.44
Drink	67.78	66.67	**68.89**	67.78	66.67	67.78
Eat	**56.67**	52.22	48.89	51.11	48.89	52.22
FallFloor	**47.78**	46.67	45.56	44.44	42.22	45.56
Fencing	56.67	62.22	63.33	60.00	**64.44**	62.22
FlicFlac	84.44	**85.56**	82.22	83.33	83.33	80.00
Golf	**98.89**	**98.89**	**98.89**	**98.89**	**98.89**	**98.89**

Continued on next page

TableB.1 – continued from previous page

Action Name	L=0	L=1	L=2	L=3	L=4	L=5
Handstand	75.56	75.56	77.78	77.78	**80.00**	**80.00**
Hit	60.00	56.67	63.33	65.56	68.89	**70.00**
Hug	80.00	82.22	82.22	82.22	**84.44**	81.11
Jump	44.44	50.00	51.11	50.00	**52.22**	46.67
Kick	37.78	40.00	44.44	46.67	**47.78**	42.22
KickBall	**61.11**	57.78	55.56	60.00	58.89	**61.11**
Kiss	77.78	**83.33**	82.22	81.11	75.56	82.22
Laugh	75.56	77.78	80.00	78.89	80.00	**82.22**
Pick	30.00	**42.22**	41.11	37.78	35.56	37.78
Pour	85.56	86.67	88.89	90.00	90.00	**91.11**
Pullup	97.78	98.89	**100.00**	**100.00**	**100.00**	**100.00**
Punch	38.89	**44.44**	41.11	43.33	43.33	43.33
Push	58.89	62.22	66.67	68.89	**68.89**	68.89
Pushup	**92.22**	87.78	88.89	87.78	88.89	88.89
RideBike	77.78	78.89	**81.11**	77.78	**81.11**	78.89
RideHorse	71.11	75.56	80.00	81.11	80.00	**84.44**
Run	**44.44**	43.33	36.67	35.56	38.89	34.44
ShakeHands	66.67	68.89	70.00	70.00	**72.22**	68.89
ShootBall	81.11	**86.67**	85.56	85.56	85.56	85.56
ShootBow	76.67	90.00	90.00	**91.11**	**91.11**	**91.11**
ShootGun	48.89	51.11	56.67	56.67	**60.00**	58.89
Sit	80.00	81.11	81.11	81.11	81.11	**84.44**
Situp	80.00	86.67	86.67	88.89	**90.00**	**90.00**
Smile	37.78	42.22	41.11	**45.56**	43.33	42.22
Smoke	44.44	54.44	**62.22**	60.00	56.67	55.56
Somersault	**71.11**	68.89	67.78	64.44	67.78	70.00
Stand	65.56	70.00	**77.78**	74.44	74.44	73.33
SwingBaseball	10.00	16.67	15.56	18.89	**22.22**	21.11
Sword	33.33	34.44	**35.56**	33.33	34.44	34.44
SwordExercise	18.89	18.89	25.56	25.56	**26.67**	21.11
Talk	73.33	67.78	70.00	**74.44**	73.33	72.22
Throw	5.56	4.44	**8.89**	4.44	4.44	4.44
Turn	71.11	74.44	74.44	73.33	73.33	**78.89**
Walk	57.78	57.78	53.33	**58.89**	53.33	56.67
Wave	10.00	14.44	12.22	14.44	12.22	**15.56**
Mean	62.14	64.40	65.03	65.10	**65.45**	65.42
NumOfBest	8	11	11	7	**18**	14

Table B.2: Detailed results of Hollywood2 for MIFS with different scale levels. Average precision (AP) (%) is used for evaluation. The best AP is marked in bold.

Action Name	L=0	L=1	L=2	L=3	L=4	L=5
AnswerPhone	34.84	41.00	39.25	**42.72**	40.55	38.45
DriveCar	95.27	95.98	96.21	**96.48**	96.34	96.22
Eat	70.76	73.10	73.10	**73.83**	70.28	70.91
FightPerson	**83.64**	82.85	82.66	82.09	82.84	82.61
GetOutCar	63.41	63.90	62.53	63.03	**64.18**	63.89
HandShake	**57.72**	52.91	52.83	49.14	50.82	48.38
HugPerson	54.49	55.23	**58.76**	58.15	56.56	54.17
Kiss	63.06	63.49	64.32	65.12	**65.19**	64.78
Run	85.88	86.07	86.10	86.10	**86.26**	85.97
SitDown	79.55	80.45	82.14	81.68	81.36	**82.30**
SitUp	35.43	35.69	**37.98**	36.51	34.20	36.92
StandUp	79.68	79.87	79.10	80.99	80.61	**81.03**
Mean	66.98	67.55	67.92	**67.99**	67.43	67.14
numOfBest	2	0	2	**3**	**3**	2

Table B.3: Detailed results of UCF101 for MIFS with different scale levels. Accuracy (%) is used for evaluation. The best accuracy is marked in bold.

Action Name	L=0	L=1	L=2	L=3	L=4	L=5
ApplyEyeMakeup	83.46	85.97	86.63	89.14	**89.90**	86.77
ApplyLipstick	62.54	67.73	65.57	68.66	**70.82**	69.78
Archery	75.03	**78.54**	77.60	76.72	77.60	76.72
BabyCrawling	83.25	86.11	84.30	**87.16**	83.64	83.64
BalanceBeam	89.25	92.47	92.47	93.55	**94.62**	**94.62**
BandMarching	91.51	87.67	92.32	93.09	**95.42**	**95.42**
BaseballPitch	86.41	86.41	89.52	89.52	**90.29**	**90.29**
Basketball	78.15	79.96	78.95	80.97	80.13	**81.02**
BasketballDunk	**98.33**	**98.33**	**98.33**	**98.33**	**98.33**	**98.33**
BenchPress	**100.00**	**100.00**	**100.00**	99.21	**100.00**	99.21
Biking	96.39	**97.27**	96.39	95.52	96.39	**97.27**
Billiards	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
BlowDryHair	73.61	75.13	74.34	76.15	**77.82**	76.11
BlowingCandles	83.96	89.29	**90.40**	**90.40**	88.28	**90.40**
BodyWeightSquats	96.67	**97.78**	**97.78**	**97.78**	96.74	**97.78**
Bowling	**96.12**	**96.12**	**96.12**	**96.12**	**96.12**	**96.12**
BoxingPunchingBag	88.47	89.96	**95.65**	94.29	94.29	94.16
BoxingSpeedBag	94.59	**94.66**	92.93	**94.66**	93.06	**94.66**
BreastStroke	**98.81**	97.66	96.47	96.47	96.47	95.32
BrushingTeeth	66.50	**70.37**	65.52	66.50	65.58	62.64

Continued on next page

Table B.3 – continued from previous page

Action Name	L=0	L=1	L=2	L=3	L=4	L=5
CleanAndJerk	93.74	94.85	93.74	96.87	95.86	**97.88**
CliffDiving	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
CricketBowling	83.70	84.49	**87.06**	84.55	85.28	84.48
CricketShot	90.87	90.78	92.42	90.97	93.78	**93.88**
CuttingInKitchen	72.62	81.32	79.42	79.23	80.67	**81.44**
Diving	99.12	**100.00**	99.12	99.12	99.12	99.12
Drumming	83.68	86.65	87.42	**91.82**	88.95	88.90
Fencing	98.81	**100.00**	**100.00**	**100.00**	98.81	98.81
FieldHockeyPenalty	89.44	93.06	94.72	93.89	**95.56**	91.94
FloorGymnastics	92.62	93.52	95.34	96.27	**96.29**	96.27
FrisbeeCatch	86.01	89.77	88.79	**91.65**	91.57	90.67
FrontCrawl	79.51	80.37	79.47	80.37	**81.27**	**81.27**
GolfSwing	97.50	**98.33**	**98.33**	**98.33**	**98.33**	**98.33**
Haircut	53.36	56.33	55.75	58.18	**60.36**	60.20
Hammering	55.50	47.69	53.26	57.47	58.69	**61.51**
HammerThrow	**99.26**	**99.26**	**99.26**	**99.26**	**99.26**	**99.26**
HandstandPushups	**94.63**	91.96	93.15	94.34	93.15	93.15
HandstandWalking	62.99	**75.97**	72.84	65.37	70.36	68.19
HeadMassage	84.47	85.33	86.12	86.14	**86.95**	**86.95**
HighJump	85.80	**86.88**	85.80	85.80	83.20	85.25
HorseRace	96.97	**100.00**	**100.00**	99.05	98.10	98.10
HorseRiding	90.15	92.58	**94.02**	93.34	93.26	92.58
HulaHoop	**97.11**	**97.11**	**97.11**	**97.11**	96.16	**97.11**
IceDancing	98.48	98.48	98.48	**99.28**	**99.28**	**99.28**
JavelinThrow	74.36	74.29	71.26	73.35	**75.43**	70.19
JugglingBalls	87.39	**92.30**	90.27	91.11	91.11	92.09
JumpingJack	91.69	92.62	92.62	94.44	**95.37**	**95.37**
JumpRope	93.86	96.49	**97.37**	**97.37**	**97.37**	**97.37**
Kayaking	77.01	77.94	**79.07**	77.21	76.29	76.29
Knitting	91.45	94.39	**96.24**	**96.24**	95.26	95.26
LongJump	**85.65**	80.58	81.56	81.56	83.15	81.56
Lunges	74.11	77.69	78.54	78.54	**79.44**	**79.44**
MilitaryParade	**97.19**	95.42	94.41	92.61	92.61	93.53
Mixing	87.76	85.51	86.58	**88.08**	87.32	85.84
MoppingFloor	58.66	**61.54**	61.03	60.14	55.05	56.09
Nunchucks	**61.61**	60.84	57.32	58.04	57.14	57.08
ParallelBars	98.20	98.20	**100.00**	98.20	98.99	98.09
PizzaTossing	61.13	66.32	66.18	**70.36**	67.19	64.02
PlayingCello	89.57	91.75	93.22	93.20	**93.96**	**93.96**
PlayingDaf	90.75	90.75	90.75	90.75	90.75	**92.23**
PlayingDhol	**99.19**	**99.19**	**99.19**	97.80	98.49	97.81

Continued on next page

Table B.3 – continued from previous page

Action Name	L=0	L=1	L=2	L=3	L=4	L=5
PlayingFlute	77.98	77.18	76.49	**78.08**	77.38	76.49
PlayingGuitar	85.05	87.81	88.56	90.64	90.64	**92.14**
PlayingPiano	91.67	91.67	95.24	**96.43**	**96.43**	**96.43**
PlayingSitar	94.01	94.73	**96.97**	94.76	95.49	96.25
PlayingTabla	96.84	**98.92**	**98.92**	**98.92**	**98.92**	96.84
PlayingViolin	84.52	85.71	85.71	**88.10**	**88.10**	85.71
PoleVault	98.33	97.59	**99.17**	97.57	98.31	97.57
PommelHorse	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
PullUps	92.86	**97.62**	95.24	96.43	96.43	**97.62**
Punch	**90.17**	**90.17**	**90.17**	**90.17**	**90.17**	**90.17**
PushUps	82.30	86.98	85.71	**89.29**	**89.29**	**89.29**
Rafting	**80.53**	79.49	80.47	77.50	78.48	79.49
RockClimbingIndoor	91.74	**93.41**	93.37	92.52	92.52	91.66
RopeClimbing	85.20	89.06	89.06	**92.04**	91.03	89.06
Rowing	88.62	92.91	92.91	93.84	93.84	**94.59**
SalsaSpin	95.73	97.44	**99.15**	97.36	97.36	96.74
ShavingBeard	75.68	**76.45**	76.37	75.60	76.37	73.27
Shotput	69.79	**74.73**	72.04	69.27	69.50	71.90
SkateBoarding	**93.13**	90.15	91.10	90.15	90.15	90.15
Skiing	82.98	85.46	84.58	87.24	85.50	**87.28**
Skijet	**96.43**	95.24	92.86	89.29	91.67	92.86
SkyDiving	92.87	90.82	92.87	92.87	93.85	**94.83**
SoccerJuggling	84.64	83.54	**86.64**	86.56	86.56	85.79
SoccerPenalty	92.64	93.46	**95.10**	94.27	94.27	94.27
StillRings	96.84	**97.92**	**97.92**	**97.92**	**97.92**	**97.92**
SumoWrestling	95.96	95.99	96.97	96.97	**97.98**	**97.98**
Surfing	90.60	93.16	**94.87**	94.02	93.16	93.16
Swing	92.33	**96.03**	**96.03**	95.24	95.24	**96.03**
TableTennisShot	**91.77**	**91.77**	90.94	91.11	90.94	90.94
TaiChi	86.90	**88.10**	**88.10**	**88.10**	**88.10**	**88.10**
TennisSwing	94.56	95.92	93.88	**96.58**	95.24	93.88
ThrowDiscus	93.62	93.64	94.57	94.57	**95.47**	**95.47**
TrampolineJumping	92.71	95.83	95.83	**96.88**	**96.88**	**96.88**
Typing	82.30	86.37	**89.55**	87.70	87.70	87.70
UnevenBars	91.82	91.82	91.96	**94.20**	91.96	**94.20**
VolleyballSpiking	99.10	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
WalkingWithDog	77.21	**81.05**	79.14	78.16	79.11	80.99
WallPushups	82.91	89.68	**90.48**	89.52	**90.48**	89.52
WritingOnBoard	92.95	94.75	**96.56**	96.40	95.50	95.66
YoYo	78.49	**79.37**	75.56	76.48	74.58	77.41
Mean	87.26	88.65	88.81	89.05	**89.08**	88.98

Continued on next page

Table B.3 – continued from previous page

Action Name	L=0	L=1	L=2	L=3	L=4	L=5
NumOfBest	19	35	36	33	37	**44**

Table B.4: Detailed results of UCF50 for MIFS with different scale levels. Accuracy (%) is used for evaluation. The best accuracy is marked in bold.

Action Name	L=0	L=1	L=2	L=3	L=4	L=5
BaseballPitch	90.15	91.87	92.53	93.10	93.68	**94.34**
Basketball	84.29	87.80	87.47	**89.77**	88.10	86.63
BenchPress	**99.20**	**99.20**	**99.20**	**99.20**	**99.20**	**99.20**
Biking	94.53	**96.33**	**96.33**	94.33	95.33	95.33
Billiards	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
BreastStroke	99.00	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
CleanAndJerk	99.00	99.00	**100.00**	**100.00**	**100.00**	**100.00**
Diving	99.43	**100.00**	99.43	**100.00**	99.43	99.43
Drumming	86.90	89.33	92.05	91.52	**93.06**	92.39
Fencing	99.00	99.00	**100.00**	**100.00**	99.00	99.00
GolfSwing	96.00	98.67	98.67	99.33	**100.00**	**100.00**
HighJump	93.80	**94.80**	94.80	93.80	94.00	93.00
HorseRace	**99.00**	98.33	98.33	98.33	98.67	98.67
HorseRiding	93.50	94.00	**95.43**	**95.43**	94.36	94.36
HulaHoop	**97.00**	96.87	96.87	96.87	96.20	96.87
JavelinThrow	84.20	85.33	84.67	86.13	**86.80**	85.13
JugglingBalls	**96.30**	94.43	94.43	95.10	94.10	95.10
JumpRope	95.71	96.14	97.43	**98.43**	**98.43**	97.43
JumpingJack	91.43	94.43	95.43	96.00	96.00	**97.00**
Kayaking	90.95	90.38	91.62	**92.19**	**92.19**	91.62
Lunges	79.07	83.20	84.44	**86.49**	85.69	85.93
MilitaryParade	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
Mixing	**96.00**	93.69	93.52	94.19	94.86	94.19
Nunchucks	70.06	**71.98**	70.42	66.71	65.58	65.01
PizzaTossing	74.40	75.10	**76.10**	75.60	74.40	75.30
PlayingGuitar	96.83	98.86	98.86	**99.43**	**99.43**	**99.43**
a PlayingPiano	97.00	98.00	**99.00**	**99.00**	**99.00**	98.00
PlayingTabla	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
PlayingViolin	**97.00**	96.00	96.00	**97.00**	**97.00**	**97.00**
PoleVault	97.48	**98.56**	97.56	96.56	96.56	96.56
PommelHorse	99.00	99.00	**100.00**	99.00	**100.00**	**100.00**
PullUps	**98.00**	97.00	97.00	97.00	97.00	**98.00**
Punch	91.06	91.63	91.06	91.63	92.06	**92.63**
PushUps	91.00	90.00	90.00	92.00	92.00	**94.00**
RockClimbingIndoor	92.57	93.90	93.05	**95.86**	95.29	94.76

Continued on next page

Table B.4 – continued from previous page

Action Name	L=0	L=1	L=2	L=3	L=4	L=5
RopeClimbing	85.40	92.96	93.46	**95.96**	94.46	92.96
Rowing	92.76	**94.76**	**94.76**	**94.76**	**94.76**	**94.76**
SalsaSpin	**98.33**	97.43	97.43	97.43	96.43	97.43
SkateBoarding	85.00	**88.00**	87.00	87.67	86.00	86.00
Skiing	93.12	93.96	94.51	94.46	**95.36**	93.27
Skijet	**94.00**	93.00	**94.00**	93.00	93.00	93.00
SoccerJuggling	87.40	92.55	92.55	**93.62**	93.05	93.05
Swing	93.33	93.87	**95.20**	**95.20**	**95.20**	**95.20**
TaiChi	92.00	**93.00**	92.00	92.00	**93.00**	**93.00**
TennisSwing	95.81	96.48	96.48	94.76	**97.05**	96.48
ThrowDiscus	96.43	**97.43**	**97.43**	**97.43**	**97.43**	**97.43**
TrampolineJumping	95.00	96.00	98.00	98.00	**99.00**	98.00
VolleyballSpiking	98.33	99.00	99.00	**100.00**	99.00	**100.00**
WalkingWithDog	87.07	88.73	89.87	**90.73**	89.40	88.87
YoYo	88.67	**89.13**	88.60	87.27	88.27	88.27
Mean	93.03	93.98	94.24	**94.45**	94.38	94.28
numOfBest	12	15	16	**24**	22	20

Table B.5: Detailed results of Olympic Sports for MIFS with different scale levels. Average precision (AP) (%) is used for evaluation. The best AP is marked in bold.

Action Name	L=0	L=1	L=2	L=3	L=4	L=5
BasketballLayup	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
Bowling	**89.11**	86.49	86.69	87.21	86.28	87.16
CleanJerk	92.43	99.09	99.09	**100.00**	**100.00**	**100.00**
DiscusThrow	84.01	87.51	89.79	91.84	**94.41**	92.98
DivingPlatform10m	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
DivingSpringboard3m	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
HammerThrow	96.59	**97.50**	**97.50**	**97.50**	**97.50**	**97.50**
HighJump	**86.63**	78.59	81.19	80.30	80.51	80.14
JavelinThrow	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
LongJump	81.79	**91.51**	80.00	80.00	71.67	85.56
PoleVault	95.83	**98.61**	95.03	97.05	**98.61**	**98.61**
ShotPut	83.29	**88.11**	85.26	81.55	84.09	82.68
Snatch	70.46	83.74	89.55	88.78	**91.22**	88.78
TennisServe	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
TripleJump	74.70	**95.00**	83.04	77.50	59.64	64.36
Vault	82.25	80.69	80.14	80.59	81.40	**82.27**
Mean	89.82	**92.93**	91.70	91.40	90.33	91.25
numOfBest	7	**10**	6	7	**10**	9

Table B.6: Detailed results of MIFS and baseline methods on TRECVID MEDTEST13 dataset. Average precision (AP) (%) is used for evaluation. The best AP is marked in bold.

	EK10		EK100	
Event ID & Name	Baseline	MIFS($L = 3$)	Baseline	MIFS($L = 3$)
E006: Birthday party	9.40	**10.66**	32.38	**35.59**
E007: Changing a vehicle tire	10.22	**12.76**	41.18	**44.97**
E008: Flash mob gathering	48.88	**52.46**	66.36	**68.73**
E009: Getting a vehicle unstuck	9.94	**11.21**	46.11	**52.04**
E010: Grooming an animal	10.88	**13.81**	31.32	**34.45**
E011: Making a sandwich	6.83	**6.99**	23.51	**25.26**
E012: Parade	38.51	**40.88**	53.56	**58.65**
E013: Parkour	68.22	**72.57**	74.93	**77.15**
E014: Repairing an appliance	**22.42**	20.52	**47.44**	45.58
E015: Working on a sewing project	11.99	**13.63**	38.16	**38.87**
E021: Attempting a bike trick	**7.48**	7.29	11.08	**11.66**
E022: Cleaning an appliance	3.90	**4.10**	21.81	**22.50**
E023: Dog Show	32.38	**41.93**	64.75	**68.75**
E024: Giving directions to a location	1.25	**1.77**	11.18	**15.75**
E025: Marriage proposal	**0.55**	0.54	**12.67**	12.04
E026: Renovating a home	4.78	**5.29**	**13.18**	13.07
E027: Rock climbing	19.11	**20.48**	22.60	**26.05**
E028: Town hall meeting	22.39	**22.81**	35.84	**36.38**
E029: Winning a race without a vehicle	16.73	**18.62**	**21.07**	17.79
E030: Working on a metal crafts project	**9.78**	8.09	16.63	**21.29**
Mean	17.78	**19.32**	34.27	36.33
numOfBest	4	**16**	4	**16**

Table B.7: Detailed results of MIFS and baseline methods on TRECVID MEDTEST14 dataset. Average precision (AP) (%) is used for evaluation. The best AP is marked in bold.

	EK10		EK100	
Event ID & Name	Baseline	MIFS($L = 3$)	Baseline	MIFS($L = 3$)
E021: Attempting a bike trick	**7.47**	7.26	11.17	**11.76**
E022: Cleaning an appliance	4.01	**4.38**	**23.67**	23.02
E023: Dog Show	28.92	**39.34**	66.17	**69.91**
E024: Giving directions to a location	1.41	**2.16**	8.31	**13.91**
E025: Marriage proposal	0.60	**0.66**	**12.87**	12.31
E026: Renovating a home	4.80	**5.31**	13.19	**13.27**
E027: Rock climbing	19.07	**20.47**	22.69	**26.08**
E028: Town hall meeting	22.46	**22.91**	**39.94**	36.93
E029: Winning a race without a vehicle	16.78	**18.68**	**21.11**	17.84
E030: Working on a metal crafts project	**10.32**	8.45	21.44	**22.49**

Continued on next page

Table B.7 – continued from previous page

	EK10		EK100	
Event ID & Name	Baseline	MIFS($L = 3$)	Baseline	MIFS($L = 3$)
E031: Beekeeping	35.91	**38.71**	63.42	**65.46**
E032: Wedding shower	7.79	**10.04**	22.61	**25.39**
E033: Non-motorized vehicle repair	21.47	**22.64**	35.34	**38.34**
E034: Fixing musical instrument	**5.08**	4.84	**23.29**	22.99
E035: Horse riding competition	28.47	**33.70**	34.69	**37.24**
E036: Felling a tree	8.49	**11.14**	19.18	**26.57**
E037: Parking a vehicle	9.56	**14.21**	27.44	**31.78**
E038: Playing fetch	0.94	**1.22**	6.96	**7.08**
E039: Tailgating	7.75	**12.55**	46.49	**46.71**
E040: Tuning musical instrument	14.07	**19.97**	27.51	**31.20**
Mean	12.77	**14.93**	27.37	**29.01**
numOfBest	3	**17**	5	**15**

Bibliography

[1] http://www.psc.edu/index.php/computing-resources/blacklight. 11.4

[2] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016. 2.3

[3] Edward H Adelson, Charles H Anderson, James R Bergen, Peter J Burt, and Joan M Ogden. Pyramid methods in image processing. *RCA engineer*, 1984. 2.1, 4.2

[4] JK Aggarwal and Michael S Ryoo. Human activity analysis: A review. *ACM Computing Surveys (CSUR)*, 2011. 4.2

[5] Arnon Amir, Marco Berg, Shih-Fu Chang, Winston Hsu, Giridharan Iyengar, Ching-Yung Lin, et al. Ibm research trecvid-2003 video retrieval system. *NIST TRECVID*, 2003. 7.2

[6] Relja Arandjelovic and Andrew Zisserman. All about vlad. In *CVPR*, 2013. 2.2, 7.2, 7.4.1, 8.1, 8.1, 8.1, 8.2, 8.3.2

[7] Relja Arandjelović, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. *arXiv preprint arXiv:1511.07247*, 2015. 2.2, 8.1

[8] Stéphane Ayache, Georges Quénot, and Jérôme Gensel. *Classifier fusion for SVM-based multimedia semantic indexing*. Springer, 2007. 2.4, 9.1

[9] Lei Bao, Shoou-I Yu, Zhen-zhong Lan, Arnold Overwijk, Qin Jin, Brian Langner, Michael Garbus, Susanne Burger, Florian Metze, and Alexander Hauptmann. Informedia@ trecvid 2011. 2011. 9.4.5, 11.3.1

[10] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *ECCV*, pages 404–417. Springer, 2006. 11.2

[11] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends in Machine Learning*, 2(1):1–127, 2009. 10.1

[12] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, 2009. 10.1, 10.2, 10.4

[13] Alessandro Bergamo and Lorenzo Torresani. Meta-class features for large-scale object categorization on a budget. In *CVPR*, 2012. 2.3, 10.2

[14] Y-L Boureau, Francis Bach, Yann LeCun, and Jean Ponce. Learning mid-level features for recognition. In *CVPR*, 2010. 7.1

[15] Ulf Brefeld, Thomas Gärtner, Tobias Scheffer, and Stefan Wrobel. Efficient co-regularised least squares regression. In *ICML*, 2006. 2.4, 9.1, 9.3.1

[16] Ming-yu Chen and Alexander Hauptmann. Mosift: Recognizing human actions in surveillance videos. 2009. 9.3.1, 11.3.1

[17] Xinlei Chen and Abhinav Gupta. Webly supervised learning of convolutional networks. *arXiv preprint arXiv:1505.01554*, 2015. 10.1

[18] Wesley Chu and Tsau Young Lin. *Foundations and advances in data mining*, volume 180. Springer, 2005. 4.4.1

[19] Mircea Cimpoi, Subhransu Maji, and Andrea Vedaldi. Deep filter banks for texture recognition and segmentation. In *CVPR*, 2015. 10.1

[20] Noel CF Codella, Apostol Natsev, Gang Hua, Matthew Hill, Liangliang Cao, Leiguang Gong, and John R Smith. Video event detection using temporal pyramids of visual semantics with kernel optimization and model subspace boosting. In *ICME*, 2012. 2.2, 7.2

[21] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. L 2 regularization for learning kernels. In *UAI*, 2009. 2.4, 9.1, 9.2, 9.4.3

[22] Trevor Darrell and Alex Pentland. Space-time gestures. In *CVPR*, 1993. 4.2

[23] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 1.1

[24] Ali Diba, Vivek Sharma, and Luc Van Gool. Deep temporal linear encoding networks. *arXiv preprint arXiv:1611.06678*, 2016. 2.1, 5.2, 5.4.2, 5.5.1, 5.5.4

[25] Mandar Dixit, Si Chen, Dashan Gao, Nikhil Rasiwasia, and Nuno Vasconcelos. Scene classification with semantic fisher vectors. In *CVPR*, 2015. 2.2, 8.1, 8.1

[26] Shahram Ebadollahi, Lexing Xie, Shih-Fu Chang, and John R Smith. Visual event detection using multi-dimensional concept dynamics. In *ICME*, 2006. 2.1, 11.2

[27] Alexei A Efros, Alexander C Berg, Greg Mori, Jitendra Malik, et al. Recognizing action at a distance. In *ICCV*, 2003. 6.1

[28] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 2014. 6.1

[29] Rob Fergus, Yair Weiss, and Antonio Torralba. Semi-supervised learning in gigantic image collections. In *NIPS*, 2009. 2.3

[30] Basura Fernando, Efstratios Gavves, José Oramas, Amir Ghodrati, Tinne Tuytelaars, and Leuven Belgium. Modeling video evolution for action recognition. In *CVPR*, 2015. 5.2, 6.2

[31] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A note on the group lasso and a sparse group lasso. *arXiv preprint arXiv:1001.0736*, 2010. 6.5

[32] Chenqiang Gao, Yinhe Du, Jiang Liu, Luyu Yang, and Deyu Meng. A new dataset and evaluation for infrared action recognition. In *CCF Chinese Conference on Computer Vi-*

sion. Springer, 2015. 12

[33] Peter Gehler and Sebastian Nowozin. On feature combination for multiclass object classification. In *CVPR*, 2009. 2.4, 9.1

[34] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *ICML*, 2014. 6.1

[35] Alex Graves, AR Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, 2013. 6.1

[36] Minh Hoai and Andrew Zisserman. Improving human action recognition using score distribution and ranking. In *ACCV*, 2014. 5.2, 6.2

[37] Rui Hou, Amir Roshan Zamir, Rahul Sukthankar, and Mubarak Shah. Damn–discriminative and mutually nearest: Exploiting pairwise category proximity for video action recognition. In *ECCV*. 2014. 2.3, 10.2

[38] Jarmo Hurri and Aapo Hyvärinen. Simple-cell-like receptive fields maximize temporal coherence in natural video. *Neural Computation*, 15(3):663–691, 2003. 6.2

[39] Aapo Hyvärinen, Jarmo Hurri, and Patrick O Hoyer. *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision.*, volume 39. Springer Science & Business Media, 2009. 6.5, 6.5

[40] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. *arXiv preprint arXiv:1612.01925*, 2016. 2.1

[41] Giridharan Iyengar and Harriet J Nock. Discriminative model fusion for semantic concept detection and annotation in video. In *ACMMM*, 2003. 2.4, 9.1, 9.3.1

[42] Tommi S Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. *NIPS*, 1999. 8.1

[43] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, 2009. 6.2

[44] Hervé Jégou and Ondřej Chum. Negative evidences and co-occurences in image retrieval: The benefit of pca and whitening. In *ECCV*. 2012. 2.2, 8.2

[45] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. On the burstiness of visual elements. In *CVPR*, 2009. 2.2, 8.1, 8.2

[46] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010. 2.2, 8.1

[47] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33 (1):117–128, 2011. 8.5

[48] Lu Jiang, Teruko Mitamura, Shoou-I Yu, and Alexander G Hauptmann. Zero-example event search using multimodal pseudo relevance feedback. In *ICMR*, 2014. 7.2

[49] Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G Hauptmann. Self-paced curriculum learning. In *AAAI*, 2015. 10.1

[50] Yu-Gang Jiang. Super: Towards real-time event recognition in internet videos. In *ICMR*, page 7. ACM, 2012. 11.2

[51] Yu-Gang Jiang, Chong-Wah Ngo, and Jun Yang. Towards optimal bag-of-features for object categorization and semantic video retrieval. In *CIVR*, 2007. 7.2

[52] Yu-Gang Jiang, Xiaohong Zeng, Guangnan Ye, Dan Ellis, Shih-Fu Chang, Subhabrata Bhattacharya, and Mubarak Shah. Columbia-ucf trecvid2010 multimedia event detection: Combining multiple modalities, contextual concepts, and temporal matching. In *TRECVID*, volume 20, pages 21–32, 2010. 2.4, 9.1, 9.3.1, 9.3.1

[53] Yu-Gang Jiang, Qi Dai, Xiangyang Xue, Wei Liu, and Chong-Wah Ngo. Trajectory-based modeling of human actions with motion reference points. In *ECCV*. 2012. 2.1, 4.2, 5.2, 6.2, 8.2

[54] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 2.1, 4.2, 5.2, 6.1, 6.2

[55] Faisal Khan, Bilge Mutlu, and Xiaojin Zhu. How do humans teach: On curriculum learning and teaching dimension. In *NIPS*, 2011. 10.2

[56] Jan J Koenderink. The structure of images. *Biological cybernetics*, 50(5):363–370, 1984. 4.1

[57] Josip Krapac, Jakob Verbeek, and Frédéric Jurie. Modeling spatial layout with fisher vectors for image categorization. In *ICCV*, 2011. 2.2, 7.2

[58] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 6.1, 6.2, 6.4.1, 11.3.1

[59] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *ICCV*, 2011. 3.2.2

[60] M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *NIPS*, 2010. 10.1

[61] Zhenzhong Lan. Learn to recognize actions through neural networks. In *MM*, 2015. 7.1, 10.1

[62] Zhenzhong Lan and Alexander G Hauptmann. Beyond spatial pyramid matching: Space-time extended descriptor for action recognition. *arXiv preprint arXiv:1510.04565*, 2015. 1.1, 4.1, 7.1

[63] Zhenzhong Lan, Lei Bao, Shoou-I Yu, Wei Liu, and Alexander G Hauptmann. Double fusion for multimedia event detection. In *MMM*. 2012. 9.1, 11.3.1

[64] Zhenzhong Lan, Lei Bao, Shoou-I Yu, Wei Liu, and Alexander G Hauptmann. Multimedia classification and event detection using double fusion. *Multimedia Tools and Applications*, 2013. 9.1, 11.3.1

[65] Zhenzhong Lan, Lu Jiang, Shoou-I Yu, Shourabh Rawat, Yang Cai, Chenqiang Gao, Shicheng Xu, Haoquan Shen, Xuanchong Li, Yipei Wang, et al. Cmu-informedia at

trecvid 2013 multimedia event detection. In *Proceedings of the TRECVID 2013 Workshop*, volume 1, page 5, 2013. 1, 5.1, 5.5.1, 5.5.2, 5.5.3, 7.2, 11.1

[66] Zhenzhong Lan, Yi Yang, Nicolas Ballas, Shoou-I Yu, and Alexander Haputmann. Resource constrained multimedia event detection. In *MMM*, 2014. 11.1

[67] Zhenzhong Lan, Ming Lin, Xuanchong Li, Alexander G Hauptmann, and Bhiksha Raj. Beyond gaussian pyramid: Multi-skip feature stacking for action recognition. In *CVPR*, 2015. 5.2, 5.4.3, 5.5.4, 6.2, 6.6.1, 8.3.4, 10.3

[68] Zhenzhong Lan, Yi Zhu, and Alexander G Hauptmann. Deep local video feature for action recognition. *arXiv preprint arXiv:1701.07368*, 2017. 2.1

[69] Ivan Laptev. On space-time interest points. *IJCV*, 64(2-3):107–123, 2005. 2.1, 4.1, 4.2, 4.3, 6.1, 9.3.1, 11.3.1

[70] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008. 7.1

[71] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 2.2, 7.1, 7.2, 9.3.1

[72] Quoc V Le, Will Y Zou, Serena Y Yeung, and Andrew Y Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *CVPR*, 2011. 6.1, 6.2, 6.6.4

[73] Huan Li, Lei Bao, Arnold Overwijk, Wei Liu, Long-Fei Zhang, Shoou-I Yu, Ming-Yu Chen, Florian Metze, and Alexander Hauptmann. Informedia@ trecvid 2010. 2010. 2.4, 9.1, 9.3.1

[74] Li-Jia Li, Hao Su, Li Fei-Fei, and Eric P Xing. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *Advances in neural information processing systems*, pages 1378–1386, 2010. 11.3.1

[75] Tony Lindeberg. Detecting salient blob-like image structures and their scales with a scale-space primal sketch: a method for focus-of-attention. *IJCV*, 11(3):283–318, 1993. 2.1

[76] Tony Lindeberg and Bart M ter Haar Romeny. *Linear scale-space I: Basic theory*. Springer, 1994. 2.1, 4.1, 4.2

[77] Jingen Liu, Qian Yu, Omar Javed, Saad Ali, Amir Tamrakar, Ajay Divakaran, Hui Cheng, and Harpreet S Sawhney. Video event recognition using concept attributes. In *WACV*, pages 339–346, 2013. 2.1, 11.2

[78] Wei Liu, Andrew Rabinovich, and Alexander C Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015. 12

[79] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2): 91–110, 2004. 2.1, 4.1, 9.3.1

[80] David Marr. Vision: A computational investigation into the human representation and processing of visual information. *Inc., New York, NY*, pages 2–46, 1982. 4.1

[81] Marcin Marszalek, Ivan Laptev, and Cordelia Schmid. Actions in context. In *CVPR*, 2009.

3.2.2, 6.1

[82] Pyry Matikainen, Martial Hebert, and Rahul Sukthankar. Trajectons: Action recognition through the motion analysis of tracked features. In *ICCV Workshops*, 2009. 2.1, 4.2

[83] Sancho McCann and David G Lowe. Spatially local coding for object recognition. In *ACCV*. 2013. 2.2, 7.2

[84] Michele Merler, Bert Huang, Lexing Xie, Gang Hua, and Apostol Natsev. Semantic model vectors for complex video event recognition. *TMM*, 2012. 2.1, 11.2

[85] Ionuţ Mironică, Bogdan Ionescu, Jasper Uijlings, and Nicu Sebe. Fisher kernel temporal variation-based relevance feedback for video retrieval. *Computer Vision and Image Understanding*, 2015. 2.2, 8.2

[86] Frank Moosmann, Eric Nowak, and Frederic Jurie. Randomized clustering forests for image classification. *PAMI*, 30(9):1632–1646, 2008. 11.2

[87] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. *arXiv preprint arXiv:1503.08909*, 2015. 2.1, 6.1

[88] Joe Yue-Hei Ng, Jonghyun Choi, Jan Neumann, and Larry S. Davis. ActionFlowNet: Learning Motion Representation for Action Recognition. *arXiv preprint arXiv:1612.03052*, 2016. 2.1

[89] Juan Carlos Niebles, Chih-Wei Chen, and Li Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *ECCV*. 2010. 3.2.2

[90] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 2001. 9.3.1

[91] Dan Oneata, Jakob Verbeek, Cordelia Schmid, et al. Action and event recognition with fisher vectors on a compact feature set. In *ICCV*, 2013. 2.2, 7.2

[92] Sangho Park and Jake K Aggarwal. A hierarchical bayesian network for event recognition of human actions and interactions. *Multimedia systems*, 10(2):164–179, 2004. 4.2

[93] Xiaojiang Peng, Limin Wang, Xingxing Wang, and Yu Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *arXiv preprint arXiv:1405.4506*, 2014. 2.1, 4.2, 5.2, 6.2

[94] Xiaojiang Peng, Changqing Zou, Yu Qiao, and Qiang Peng. Action recognition with stacked fisher vectors. In *Computer Vision–ECCV 2014*, pages 581–595. Springer, 2014. 6.1

[95] Florent Perronnin and Christopher Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007. 2.2, 8.1, 8.2

[96] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*. 2010. 2.1, 2.2, 4.2, 7.2, 8.1, 8.1, 8.1, 8.2, 8.3.1

[97] Zhaofan Qiu, Ting Yao, and Tao Mei. Deep quantization: Encoding convolutional activations with deep generative model. *arXiv preprint arXiv:1611.09502*, 2016. 2.1, 5.2, 5.5.1, 5.5.4

[98] Kishore K Reddy and Mubarak Shah. Recognizing 50 human action categories of web videos. *Machine Vision and Applications*, 24(5):971–981, 2013. 3.2.2

[99] Alexander Richard and Juergen Gall. A bow-equivalent recurrent neural network for action recognition. In *BMVC*, 2015. 6.2, 6.4.1, 6.4.2

[100] Jorge Sánchez, Florent Perronnin, and TeóFilo De Campos. Modeling the spatial layout of images beyond spatial pyramids. *Pattern Recognition Letters*, 33(16):2216–2223, 2012. 2.2, 7.2

[101] Michael Sapienza, Fabio Cuzzolin, and Philip HS Torr. Feature sampling and partitioning for visual vocabulary generation on large action classification datasets. *arXiv preprint arXiv:1405.7545*, 2014. 5.2, 6.2

[102] Bernhard Schölkopf and Christopher JC Burges. *Advances in kernel methods: support vector learning*. MIT press, 1999. 9.3.4

[103] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013. 4.1

[104] Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Learnability, stability and uniform convergence. *JMLR*, 11(Oct):2635–2670, 2010. 4.1

[105] Ling Shao, Xiantong Zhen, Dacheng Tao, and Xuelong Li. Spatio-temporal laplacian pyramid coding for action recognition. *IEEE Transactions on Cybernetics*, pages 2168–2267, 2013. 2.1, 4.2

[106] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *arXiv preprint arXiv:1406.2199*, 2014. 2.1, 4.2, 5.1, 5.2, 5.4.3, 5.5.4, 6.1, 6.2, 6.5, 6.6.4, 1

[107] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2.1, 4.2

[108] Cees GM Snoek and Marcel Worring. Concept-based video retrieval. *Foundations and Trends in Information Retrieval*, 2(4):215–322, 2008. 7.2

[109] Cees GM Snoek, Marcel Worring, and Arnold WM Smeulders. Early versus late fusion in semantic video analysis. In *ACMMM*, 2005. 1.1, 1.2, 2.4, 9.1, 9.2

[110] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 2.1, 3.2.2

[111] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. *arXiv preprint arXiv:1502.04681*, 2015. 6.2

[112] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*, 2014. 2.3

[113] Ju Sun, Xiao Wu, Shuicheng Yan, Loong-Fah Cheong, T-S Chua, and Jintao Li. Hierarchical spatio-temporal context modeling for action recognition. In *CVPR*, 2009. 2.1,

4.2

[114] Vladyslav Sydorov, Mayu Sakurada, and Christoph H Lampert. Deep fisher kernels–end to end learning of the fisher kernel gmm parameters. In *CVPR*, 2014. 6.2, 6.4.1, 6.4.2

[115] Cisco systems. White paper: Cisco vni forecast and methodology, 2015-2020. *Cisco*, 2016. 1.1

[116] Amir Tamrakar, Saad Ali, Qian Yu, Jingen Liu, Omar Javed, Ajay Divakaran, Hui Cheng, Harpreet Sawhney, and S R I International Sarnoff. Evaluation of low-level leatures and their combinations for complex event detection in open source videos. *CVPR*, pages 3681–3688, 2012. 2.1, 11.2

[117] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. C3d: generic features for video analysis. *arXiv preprint arXiv:1412.0767*, 2014. 5.2

[118] Joel A Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12(4):389–434, 2012. 4.4.1

[119] Koen EA Van De Sande, Theo Gevers, and Cees GM Snoek. Evaluating color descriptors for object and scene recognition. *PAMI*, 2010. 9.3.1, 11.3.1

[120] Merijn Van Erp, Louis Vuurpijl, and Lambert Schomaker. An overview and comparison of voting methods for pattern recognition. In *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on*, pages 195–200. IEEE, 2002. 2.4, 9.1

[121] Balakrishnan Varadarajan, George Toderici, Sudheendra Vijayanarasimhan, and Apostol Natsev. Efficient large scale video classification. 2015. 2.1, 5.2, 6.2

[122] GÃŒl Varol, Ivan Laptev, and Cordelia Schmid. Long-term Temporal Convolutions for Action Recognition. *arXiv preprint arXiv:1604.04494*, 2016. 2.1

[123] Andrea Vedaldi and Andrew Zisserman. Efficient additive kernels via explicit feature maps. *IEEE transactions on pattern analysis and machine intelligence*, 34(3):480–492, 2012. 5.4.3

[124] Heng Wang, Muhammad Muneeb Ullah, Alexander Klaser, Ivan Laptev, and Cordelia Schmid. Evaluation of local spatio-temporal features for action recognition. In *BMVC*, 2009. 6.1

[125] Heng Wang, Alexander Klaser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *CVPR*, 2011. 2.1, 4.1, 4.2, 4.3, 5.2, 6.1, 6.2, 6.3, 6.5, 7.3.1, 8.2

[126] Heng Wang, Cordelia Schmid, et al. Action recognition with improved trajectories. In *ICCV*, 2013. 2.1, 2.2, 4.2, 4.5, 4.5.1, 4.5.1, 4.5.3, 5.1, 5.2, 6.1, 6.2, 6.6.1, 7.4.1, 8.1, 8.2, 8.3.4, 10.3, 11.3.1

[127] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. 2015. 2.1, 5.2, 6.2

[128] Limin Wang, Yuanjun Xiong, Zhe Wang, and Yu Qiao. Towards Good Practices for Very Deep Two-Stream ConvNets. *arXiv preprint arXiv:1507.02159*, 2015. 5.1, 5.4.2, 5.4.3

[129] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal Segment Networks: Towards Good Practices for Deep Action Recogni-

tion. In *ECCV*, 2016. (document), 2.1, 5.1, 5.2, 5.4, 5.4.1, 5.4.2, 5.4.3, 5.5.1, 5.5.4

[130] Zuxuan Wu, Xi Wang, Yu-Gang Jiang, Hao Ye, and Xiangyang Xue. Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. *arXiv preprint arXiv:1504.01561*, 2015. 2.1, 5.2, 6.2

[131] Dan Xu, Elisa Ricci, Yan Yan, Jingkuan Song, and Nicu Sebe. Learning deep representations of appearance and motion for anomalous event detection. *arXiv preprint arXiv:1510.01553*, 2015. 12

[132] Zhongwen Xu, Yi Yang, and Alexander G Hauptmann. A discriminative cnn video representation for event detection. *arXiv preprint arXiv:1411.4006*, 2014. 2.2, 5.1, 5.5.1, 5.5.2, 8.1, 8.1, 8.1, 8.2

[133] Junji Yamato, Jun Ohya, and Kenichiro Ishii. Recognizing human action in time-sequential images using hidden markov model. In *CVPR*, 1992. 4.2

[134] Jun Yang, Yu-Gang Jiang, Alexander G Hauptmann, and Chong-Wah Ngo. Evaluating bag-of-visual-words representations in scene classification. In *Workshop on ICMR*, pages 197–206. ACM, 2007. 2.1, 11.2

[135] Yang Yang, De-Chuan Zhan, Ying Fan, Yuan Jiang, and Zhi-Hua Zhou. Deep learning for fixed model reuse. 2017. 12

[136] Yi Yang, Yue-Ting Zhuang, Fei Wu, and Yun-He Pan. Harmonizing hierarchical manifolds for multimedia document semantics understanding and cross-media retrieval. *TMM*, 2008. 2.4, 9.1

[137] Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. Describing videos by exploiting temporal structure. In *CVPR*, 2015. 5.5.3

[138] Ehsan Younessian and Deepu Rajan. Multi-modal fusion for associated news story retrieval. *Multimedia Tools and Applications*, 74(8):2563–2585, 2015. 12

[139] C. Zach, T. Pock, and H. Bischof. A Duality Based Approach for Realtime TV-L1 Optical Flow. In *29th DAGM conference on Pattern recognition*, 2014. 2.1

[140] Shengxin Zha, Florian Luisier, Walter Andrews, Nitish Srivastava, and Ruslan Salakhutdinov. Exploiting image-trained cnn architectures for unconstrained video classification. *arXiv preprint arXiv:1503.04144*, 2015. 2.1, 5.2, 5.5.1, 6.2

[141] Bowen Zhang, Limin Wang, Zhe Wang, Yu Qiao, and Hanli Wang. Real-time Action Recognition with Enhanced Motion Vector CNNs. In *CVPR*, 2016. 2.1

[142] Yi Zhu and Shawn Newsam. Depth2Action: Exploring Embedded Depth for Large-Scale Action Recognition. In *ECCVW*, 2016. 5.2