2018

Simple and Complex Human Action Recognition in Constrained and Unconstrained Videos

Eman Mohammadi Nejad
University of Windsor

Simple and Complex Human Action Recognition in Constrained and Unconstrained Videos

By

Eman Mohammadi Nejad

A Dissertation
Submitted to the Faculty of Graduate Studies
through the Department of Electrical and Computer Engineering
in Partial Fulfilment of the Requirements for
the Degree of Doctor of Philosophy
at the University of Windsor

Windsor, Ontario, Canada

2018

Simple and Complex Human Action Recognition in Constrained and Unconstrained Videos

by

Eman Mohammadi Nejad

APPROVED BY:

G. Fortino, External Examiner
University of Calabria

J. Lu
School of Computer Science

K. Tepe
Department of Electrical and Computer Engineering

B. Shahrrava
Department of Electrical and Computer Engineering

M. Saif, Co-advisor
Department of Electrical and Computer Engineering

J. Wu, Advisor
Department of Electrical and Computer Engineering

February 2, 2018

Declaration of Co-Authorship and Previous Publication

I. Co-Authorship

I hereby declare that this thesis incorporates material that is result of joint research, as follows: This dissertation incorporates the outcome of a joint research undertaken in collaboration with Dr. Yimin Yang under the supervision of professors Jonathan Wu and Mehrdad Saif. The collaboration is covered in Chapters 3, 4, 5 and 6 of the dissertation. In all chapters, the key and primary contributions, experimental designs, data analysis and interpretation, were implemented by the author, and the contributions of the co-authors were primarily through the provision of proof reading and reviewing the research papers regarding the technical and vocabulary contents.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis, and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis.

I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

II. Previous Publication

This thesis includes 6 original papers that have been previously published/submitted for publication in peer reviewed journals and conferences, as follows:

Dissertation Chapter	Publication title/full citation	Publication status
Chapter 3	E. Mohammadi, Q.M. Jonathan Wu, M. Saif, "Human Activity Recognition Using an Ensemble of Support Vector Machines," Proceedings of International Conference on High Performance Computing and Simulation (HPCS), Innsbruck-Austria, 2016.	Published
	E. Mohammadi, Q.M. Jonathan Wu, M. Saif, "Human Action Recognition by Fusing the Outputs of Individual Classifiers," Proceedings of Computer and Robot Vision (CRV), Victoria-BC, 2016.	Published
Chapter 4	E. Mohammadi, Q.M. Jonathan Wu, M. Saif, Y. Yang, "Effect of Wavelet and Hybrid Classification On Motion Based Features," Proceedings of International Conference on Image Processing (ICIP), Beijing-China, 2017.	Published
Chapter 5	E. Mohammadi, Q.M. Jonathan Wu, M. Saif, "Improved Rank Pooling Strategy for Complex Action Recognition," IEEE International Conference on Systems, Man, and Cybernetics (SMC), Banff-AB, 2017.	Published
	E. Mohammadi, Q.M. Jonathan Wu, Yimin Yang, M. Saif, "Hierarchical Feature Representation for Unconstrained Video Analysis," Neurocomputing Journal.	Accepted with Revision
Chapter 6	E. Mohammadi, Q.M. Jonathan Wu, M. Saif, Yimin Yang, "Key Action Perception for Constrained and Unconstrained Video Analysis,"IEEE Transactions on Circuits and Systems for Video Technology.	Under Review

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as a graduate student at the University of Windsor.

III. General Declaration

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis. I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

Human action recognition plays a crucial role in visual learning applications such as video understanding and surveillance, video retrieval, human-computer interactions, and autonomous driving systems. A variety of methodologies have been proposed for human action recognition via developing of low-level features along with the bag-of-visual-word models. However, much less research has been performed on the compound of pre-processing, encoding and classification stages. This dissertation focuses on enhancing the action recognition performances via ensemble learning, hybrid classifier, hierarchical feature representation, and key action perception methodologies.

Action variation is one of the crucial challenges in video analysis and action recognition. We address this problem by proposing the hybrid classifier (HC) to discriminate actions which contain similar forms of motion features such as walking, running, and jogging. Aside from that, we show and proof that the fusion of various appearance-based and motion features can boost the simple and complex action recognition performance.

The next part of the dissertation introduces pooled-feature representation (PFR) which is derived from a double phase encoding framework (DPE). Considering that a given unconstrained video is composed of a sequence of simple frames, the first phase of DPE generates temporal sub-volumes from the video and represents them individually by employing the proposed improved rank pooling (IRP) method. The second phase constructs the pool of features by fusing the represented vectors from the first phase. The pool is compressed and then encoded to provide video-parts vector (VPV). The DPE framework allows distilling the video representation and hierarchically extracting new information. Compared with recent video encoding approaches, VPV can preserve the higher-level information through standard encoding of low-level features in two phases. Furthermore, the encoded vectors from both phases of DPE are fused along with a compression stage to develop PFR.

The real-world long-shot video streams contain complicated contents and editing artifacts. However, the conventional action recognition frameworks are only capable of analyz-

ing the pre-segmented short-shot videos. The last chapter of this dissertation focuses on key action perception (KAP) along with a robust video action clustering for unconstrained and constrained video analysis. The KAP includes two classifiers: the former detects the key action among multiple temporal clusters, and the latter recognizes the key action which is obtained by the former classifier. The video action clustering is the essential pre-processing step for KAP implementation. The sequential relationship of the video frames and complexity of motion representations provide challenges in video action clustering. We propose two novel multi-layer subspace video action clustering (ML-VAC) techniques to encode the sequential relationships of constrained and unconstrained video frames without having any prior knowledge about the number of temporal clusters in a given video.

We evaluate the proposed techniques on simple and complex datasets, such as UCF50, HMDB51, Hollywood2, KTH, Weizmann, URADL, UCF101, Olympic Sports, and Keck Gestures. The employed datasets contain constrained and unconstrained video samples to test the proposed strategies in different conditions.

Dedication

to my

loving wife and daughter

Acknowledgements

I would like to express my special appreciation and thanks to my advisor, Dr. Q.M. Jonathan Wu and co-advisor, Dr. Mehrdad Saif for giving me the opportunity to work under their supervision as well as for their guidance and support in my research. Additionally, I like to thank the committee members, Dr. Behnam Shahrrava, Dr. Jianguo Lu, and Dr. Kemal Tepe for taking time out of their busy schedule to come over and help me with their insightful comments and encouragement. I like to convey my sincere gratitude to Dr. Louise Rueda, who helped me to learn the fundamental items of the machine learning domain through his course on pattern recognition. Furthermore, I sincerely appreciate the department graduate secretary Ms. Andria Ballo for all her support and guidance. I like to convey my sincere thanks to Dr. Yimin Yang who has continuously spent time and energy in reviewing my papers. I sincerely thank my beloved wife, Atefeh, who continuously motivated me and stayed beside me throughout my Ph.D. program. I thank my fellow labmates for the stimulating discussions and for all the fun we have had in the last few years. Words cannot express how grateful I am to my mother, father, my mother-in-law, and father-in-law for all of the sacrifices that they have made on my behalf. Their prayer for me was what sustained me thus far. Finally, I convey my sincere thanks to Wikipedia, Google, and the researchers around the world, helping me to grow my knowledge and figure out the right way to come out of the darkness of ignorance and false beliefs.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

HC: Hybrid Classifier

PFR-EF: Pooled Feature Representation with Early Fusion

PFR-LF: Pooled Feature Representation with Late Fusion

DPE: Double Phase Encoding

IRP: Improved Rank Pooling

VPV: Video Parts Vector

KAP: Key Action Perception

ML-VAC: Multi-layer Subspace Video Action Clustering

UCAV: Unconstrained Video Action Clustering

CVAC: Constrained Video Action Recognition

HOG: Histogram of Oriented Gradients

HOF: Histogram of Oriented Flows

MBH: Motion Boundary Histograms

IRV: Individual Represented Vectors

LSN: Learning Strategy with Sub-Network Nodes

SVM: Support Vector Machine

ELM: Extreme Learning Machine

NN: Neural Networks

CNN: Convolutional Neural Networks

IDT: Improved Dense Trajectory

DS: Dempster Shafer

PCA: Principal Component Analysis

GMM: Gaussian Mixture Model

FVE: Fisher Vector Encoding

RSS: Random Subset Selection

DL-SNN: Double Layer Net with Sub-Network Nodes

DWT: Discrete Wavelet Transform

CHAPTER 1

Introduction

In this chapter, we introduce the topic of this Ph.D. dissertation, simple and complex action recognition in constrained and unconstrained videos in Section 1.1. Next, we present the motivation of enhancing the action recognition performances in Section 1.2. Then, we clarify the research challenges related to simple and complex action recognition in Section 1.3. The objective and major contributions of our work are presented in Section 1.4. Finally, we explain the structure of this dissertation in Section 1.5.

1.1 Video Action Recognition

Nowadays, video streams and video cameras are considered as crucial parts of human life. The video cameras are utilized in different public and private places such as homes, banks, schools, hospitals, and many other locations. The required hardware for capturing and storing the videos is becoming more and more available among societies. Mobile phones, tablets, computers, laptops, and on-the-shelf cameras can capture, save, and distribute video streams. The captured video streams are considered as valid forms of transferring information in different real-world scenarios such as video surveillance and healthcare applications. However, monitoring of captured videos is a tough task due to the lack of human resources and privacy issues. Instead of human resources, intelligent systems can be employed to monitor and analyse video streams. The affordable, powerful, and rapidly evolving systems can analyse video streams much faster than human resources.

Human action recognition plays a prominent role in video analysis and visual learning frameworks. The goal of human action recognition is to automatically analyse ongoing video streams from unknown video frames to detect and recognize human actions. In a

more extreme and continuous scenario, the starting and ending frames of all activities from an input video must be detected and followed by an action recognition framework. Simple actions have the property of simplicity and periodicity such as walking, boxing, and running. Therefore, they can be easily recognized due to their simplicity and limited viewpoints. However, complex action recognition is considered as a challenging task due to viewpoint changes, cluttered backgrounds, and motion speed variations. Furthermore, the same class of a complex action may contain a huge intra-class appearance and motion variations. Consequently, complex actions contain longer and sophisticated temporal structures compare to the simple actions. The constrained videos are captured with the stable cameras and backgrounds. In other words, the camera is not moving while capturing the constrained videos. However, unconstrained videos are captured with the dynamic cameras and backgrounds. A given unconstrained video may contain complex contents including numerous human actions. Thus, analysis of an unconstrained video is a very challenging task and requires sophisticated algorithms compare to the constrained video analysis.

We believe that different types of frameworks with a variety of complexities can be applied to recognize human actions in constrained and unconstrained videos. These frameworks are applied to different applications with a prior knowledge about the video types. For instance, the video surveillance and entertainment applications usually deliver constrained videos since the cameras and backgrounds are stable. In this dissertation, we enhance the simple and complex action recognition in constrained and unconstrained videos via supervised learning. Our ultimate goal is to boost the action recognition performances in constrained and unconstrained videos by learning actions from training video samples and recognizing them in unseen and realistic video settings. It is worth pointing out that we learn action models using training video samples and their labels. Then, we apply the unseen videos into the trained models to recognize actions.

1.2 Motivation

Many of the video analysis approaches are dependent on the action recognition frameworks. Visual content understanding and video action recognition are the fundamental

building blocks in diverse applications such as video retrieval, healthcare monitoring, natural human-machine interaction, video surveillance, autonomous driving platforms, tele-immersion systems, and digital entertainment [6, 7, 8, 9]. We believe that boosting the action recognition performances aims to design and develop powerful video analysis systems in different industrial domains. This section describes the most important applications which can benefit from action recognition algorithms (see Figure 1.2.1).

1.2.1 Applications

Video Retrieval. Nowadays, people pay more attention to record their daily activities using digital cameras and upload the video streams on the internet. The rapid growing of the video streams leads to having problems of categorizing the available videos according to their contents and existing human actions in videos. Furthermore, the developments of social and semantic web applications have inspired the advancement of video retrieval and annotation tools using action recognition frameworks. Manual classification of uploaded videos into a variety of groups is a tremendously time-consuming and impossible task. The field of video action recognition has attracted considerable attention to addressing the issue of video retrieval in an automatic format. Previous video retrieval approaches aimed to solve this problem via a pattern recognition framework which is trained using the features from training videos and enables the intelligent system to categorize the video actions in new unseen samples. Altogether, action recognition systems and video understandings are highly essential and unavoidable for applicable video retrieval systems.

Video Surveillance. Recently, video surveillance cameras have widely accepted by societies and employed in a variety of public and private locations such as airports, train stations, shopping malls, banks, casinos, swimming pools, cinemas, and parking lots. The ultimate goal of video surveillance systems is to detect intrusion and analysis of human behavior. There are numerous real-world applications which require the benefits from video surveillance systems. The intelligent systems can watch for kids and monitor their behaviors when the parents are away. The intelligent cameras can be employed to properly take care of elderly parents or neighbors throughout a day. The recent threats around the

world encourage the usage of intelligent cameras to monitor people in public places. All in all, video surveillance cameras are becoming more often to detect and even prevent security violations. The rapid growing of the installed video surveillance cameras in different locations shows the increased demands for reliable action recognition systems.

Tele-Immersion Systems. Tele-immersion is an advanced form of virtual reality which aims people who are widely separated from each other to work and share experiences together as if they were located in the same environment. There are numerous potential applications for tele-immersions such as design and virtual prototyping, maintenance and repair, and surgical activities. In a tele-immersive environment, intelligent systems localize and recognize human actions along with the physical and virtual objects. Then, the people and objects are projected in realistic and geographically distributed immersive environments. Human action detection and recognition play a crucial factor in tele-immersion applications where people and their actions must be recognized and projected to an immersive space.

Driver Assistance Systems. The advanced generation of driver assistance systems is capable of considering driver preferences, driver intentions, and the overall traffic conditions. Human action prediction is necessary to monitor people around a car and control the car's movements automatically. From the other side, action recognition is an important factor to monitor the driver and alarm about the consciousness based on the movements of the hands and body of the driver.

Health Care Monitoring. Nowadays, the population of old people is rapidly increased around the world. Thus, there is a fast increasing demand for intelligent systems which aim to detect and emerge about upcoming and existing physical and mental health problems of old people and patients. For the purpose of healthcare monitoring of elderly people and patients, the human action and behavior recognition is becoming more and more important and widely utilized in communities. Detecting, recognizing, and recording the time and location of daily human activities such as housekeeping, walking, food preparation, sleeping, and exercise, allows medical scientists to offer appropriate strategies to improve diet and medication adherence.

Digital Entertainments and Human-Computer Interactions. The human-computer

interaction is a communication format between users and digital devices. The flow of data between the user and device is described as the loop of interaction. Nowadays, video cameras and visual learning frameworks are considered as the main loop of interaction to provide a natural and intuitive way of human interaction with a digital equipment. The Microsoft Kinect sensor (see Figure 1.2.1(e)) is one of the digital interacting equipments, available in millions of homes around the world. Kinect includes efficient motion sensors which allow users to control and interact with it through a natural user interface. Consequently, human gesture and action recognition is the key factor for the Kinect and other available interacting devices.

1.3 Research Challenges

Video action recognition is a challenging task despite significant efforts by the image processing and computer vision scientists. Figure 1.3.1 depicts the frame sequences from a complex video to justify the challenges in the video analysis and action recognition domain. In this section, we present several research challenges in the field of video action recognition.

Occlusion. The general action recognition algorithms assume to clearly see the performed action in video streams. In the real-world scenario, we always encounter extreme cases where people are occluded while performing an action. The occlusions can be happened by the objects or people which are located in the field of view of the camera. Since some of the body parts are not visible during occlusions, analyzing and recognizing an action is very challenging in extreme cases. Figure 1.3.1 depicts some moments with occlusions from a given video.

Volumetric analysis of actions via local features aims to address this challenge by analysis entire spatiotemporal volumes of a given video. In this case, local features are extracted when they are not occluded and this is considered as enough informative data for action recognition. The local appearance and motion features can categorize actions via analysis of tiny patches. Additionally, the local representations aim to deal with illumination, pose and shape changes more efficiently compare to the global representations. It is worth

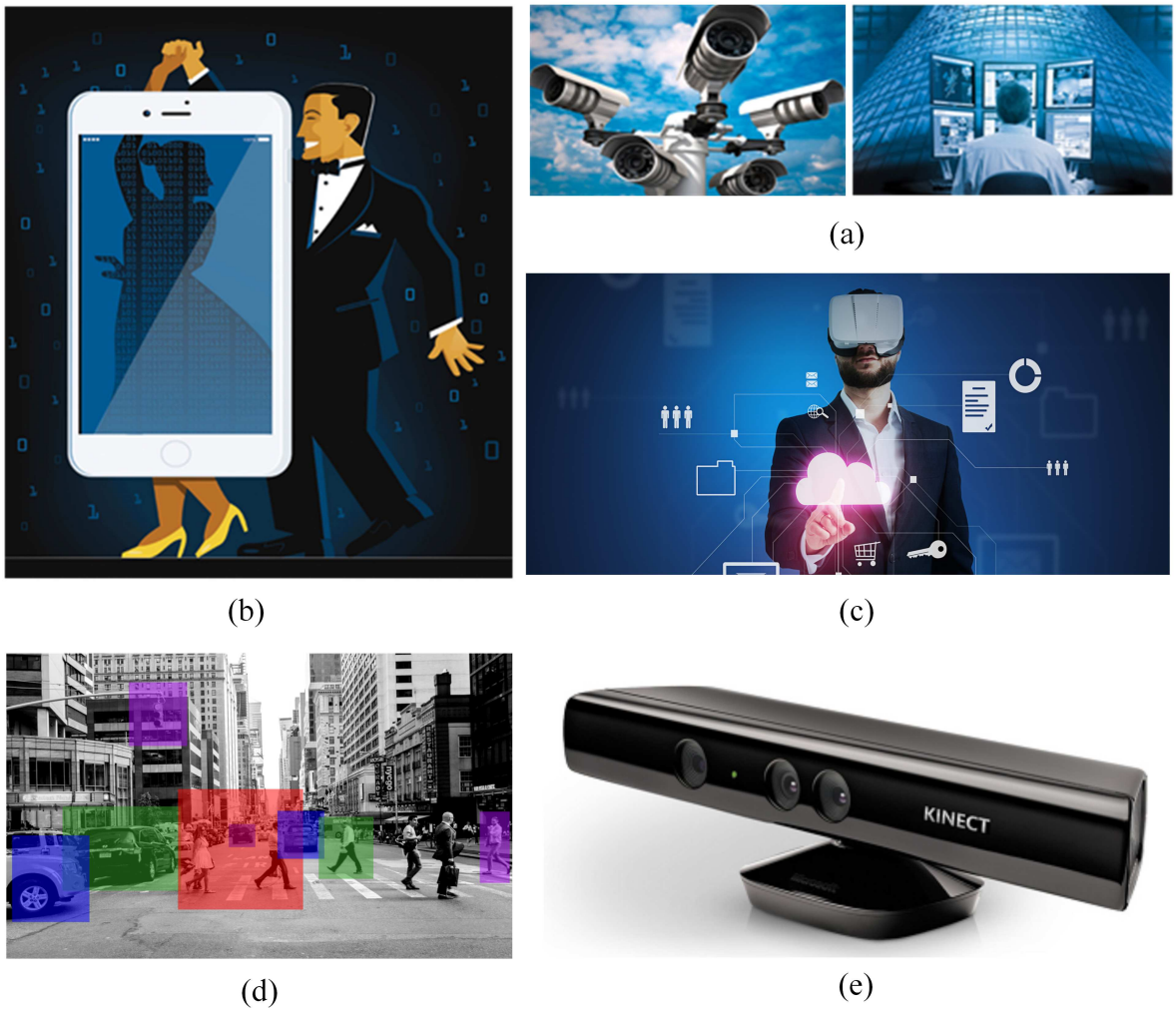


Figure 1.2.1: The important samples of applications based on human action recognition frameworks. The (a) refers to video surveillance systems; (b) refers to tele-immersion application; (c) shows the human-computer interaction; (d) refers to the advanced deriving systems; and (e) shows the Kinect sensor.

pointing out that selection of the informative local features and patches is an open problem in the computer vision domain.

Viewpoint Variations. The angle of the camera to a subject directly affects the action recognition performances. Most of the conventional visual learning frameworks assume to see an action from a fixed viewpoint. However, in the real-world cases, the videos can be captured using different points of views. In different positions of a camera, the posture and form of a body are changed considerably. Consequently, the appearance and motion-based features are changed based on a variety of body poses derived from different camera angles.

Increasing the number of training samples from different viewpoints of cameras is the most effective solution for this challenge. Thus, the features from the same action and different viewpoints are utilized to train a single classifier. Alternatively, we can capture several videos from different points of views on a subject. Then, the features from a variety of viewpoints can be synthesized to create a more robust feature vector for action recognition. We can classify actions using different feature sets in two formats: first, the single classifier can be trained using the fused feature vector inspired by concatenation of individual features. Second, several single classifiers can be trained using individual feature sets. Then the scores of single classifiers are fused to make the final decision about a given video. Chapter 3 of this dissertation evaluates different strategies to fuse individual feature sets and enhance the action recognition performances in constrained and unconstrained videos.

Camera Motion. The dynamic of cameras highly affects action recognition performances since redundant and unnecessary motion features are extracted along with the informative features in unconstrained conditions. Figure 1.3.1 shows several moments with camera motion while capturing the video. In this case, the irrelevant features are extracted from the background and included in the feature sets of walking action. Recently, the motion boundary histograms (MBH) have been proposed by [10] to address this issue. MBH is obtained by computing the derivatives of motion patterns and aims to use the trajectories from humans and objects of interests. The MBH enhances the action recognition performance using the motion features derived from optical flow.

Interclass Similarity and Anthropometric Variations. The same action may vary

from person to person, which leads to the intraclass variations. Furthermore, the speed and strength of action performance highly affect the interclass gaps. For instance, walking action can be performed slowly or rapidly based on the age, shape, and other physical properties of different people. Additionally, there is no guarantee that a person will iterate the action at the same speed and strength every time. From the other side, any video dataset may contain similar classes such as running, walking and jogging. The similar activities may express identical shapes and provide comparable probabilities while classifying a video based on the actions. This issue is called as interclass similarity which is a common problem in video action recognition.

The robust video representation algorithms aim to deal with interclass similarity and anthropometric variations. Appearance-based features such as the histogram of oriented gradients (HOG) are the simplest means of capturing pose and shape variations. HOG aims to represent the shape of the body performing an action. The motion-based features such as the histogram of oriented flows (HOF) and MBH are robust to address the issue of anthropometric variations. These local motion features neglect the pose and appearance variations. We hypothesize that enhancing the classification module using motion features can improve the action recognition performances by solving the interclass similarity issue in constrained videos. Chapter 4 of this dissertation thoroughly describes the novel hybrid classification module to address the issue of interclass similarity.

Cluttered Background. The dynamic or cluttered background provides ambiguous information in the feature sets from a given video. Motion-based features are highly affected by a dynamic background since irrelevant data is attached to the informative motion features of a video. To address this issue, most of the algorithms assume to see a static background or handle background segmentation from the videos prior to employing the action recognition platforms. The advanced local feature extraction methods, such as improved dense trajectories, are less affected by cluttered backgrounds. The HOF and MBH local features in the volumetric analysis of a video are robust with noises, camera motion, and dynamic backgrounds. Chapter 5 of this dissertation provides a method to extract the most informative features of a video and addresses the issue of dynamic backgrounds in action recognition frameworks.

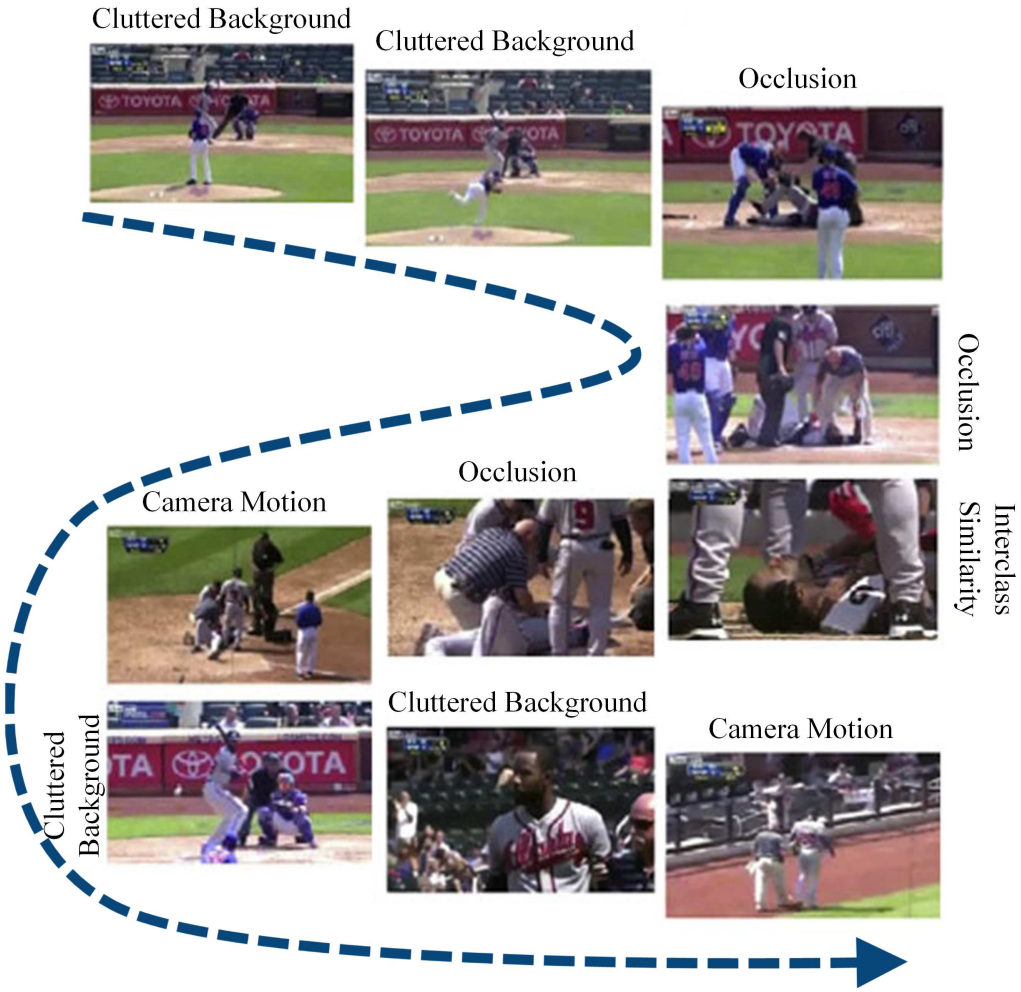


Figure 1.3.1: The frame samples of a video stream to demonstrate the overall challenges in the action recognition domain.

Real-World Unconstrained Conditions. In real-world scenarios, the dynamic camera and background may be involved in unconstrained videos. These problems can be solved by enhancing the local features and video representation methodologies. The bigger problems come up when a video consists of several human actions. Most of the common action recognition algorithms assume to see a sole action in a short shot video stream. Thus, the extracted video features contain information about that single action. However, in case of having several actions, the video features will contain information from several actions since the conventional algorithms analyse the entire video for feature extraction. In order to address this issue, a powerful video action clustering must be applied to temporally segment a given video into several actions. Next, the action recognition frameworks can be applied to individual temporal segments for analyzing of individual actions. Chapter 6 of this dissertation proposes multi-layer video clustering approaches for constrained and unconstrained video analysis. Furthermore, the Chapter 6 detects the key action among multiple clusters and recognizes the action of the key cluster using a double label learning framework.

1.4 Objective and Contributions

The major objective of this Ph.D. dissertation is to improve the performance of video action recognition by proposing a number of enhancements to the bag of visual world model (BoVW). The enhancements are performed by introducing a video clustering approach before extraction of features in the BoVW model. The video clustering approach is capable of segmenting a given video into plausible actions without having any prior knowledge about the number of clusters. Additionally, the proposed hybrid classifier and hierarchical feature representation approaches along with the improved rank pooling method aim to boost the simple and complex action recognition in constrained and unconstrained benchmark video datasets. This section enlists the major contributions of this dissertation as follows:

1) As an initial step, we have studied and tried the cooperative learning methods to enhance the action recognition algorithms by combining the appearance and motion-based features of a given video. The obtained experiments demonstrate that fusion

of different features can enhance the action recognition performances in benchmark datasets. However, the early fusion of features generates the huge vectors which make the classification task very challenging and time-consuming. To address this issue, we have trained single classifiers using individual features and combined the scores of classifiers for making the final decision about a given video. This strategy could enhance the action recognition performances compare to the early fusion method.

2) As the second contribution, we propose a hybrid classifier (HC) to confidently discriminate the similar actions such as walking, running, and jogging. It is worth pointing out that similar classes provide analogous probabilities in the action classification step. The proposed HC is capable to check the validity of produced probabilities before making the final decision about a given sample. Furthermore, we analyse the effect of motion saliency map and proposed hybrid classifier on simple and complex action recognition performance in benchmark datasets.

3) As the third contribution, we propose the hierarchical pooled-feature representation (PFR) methodology to leverage the motion and appearance-based information from a given video. The PFR is derived from the proposed double phase encoding (DPE) framework which represents the individual temporal blocks of a given video in two phases. The low-level information is obtained by representing the individual blocks at the first phase of DPE. At the second phase of DPE, The higher-level information is achieved by synthesizing the represented features of the first phase. The PFR has been experimented on six benchmark datasets and obtained reliable results compare to the state-of-the-art methods.

4) The representation of individual blocks of DPE framework is performed by the proposed improved rank pooling (IRP) strategy. The IRP enhances the regular rank pooling method by removing the similar frames of a given video. The cleaned set of features aims to generate reliable rank-pooled vectors to represent individual temporal blocks in the DPE framework. Additionally, the double thresholding scheme has been introduced to reliably clean redundant data from short and long shot videos.

5) The conventional action recognition algorithms have been developed and tested on pre-segmented video datasets which contain a sole action in few seconds. However, the videos contain long and complicated temporal contents in real-world scenarios. We hypothesize that clustering a captured video into plausible actions is a crucial pre-processing step to action recognition task in real-world applications such as video surveillance and video retrieval. As the fourth contribution, we propose the two video action clustering approaches for constrained and unconstrained videos. The proposed clustering approaches aim to segment a given video into plausible actions without having any prior knowledge about the number of clusters in a given video. The introduced video clustering approaches obtained comparable results in benchmark datasets.

6) As the last contribution, we propose the key action perception (KAP) framework to analyse the temporal clusters of a video. The KAP aims to detect and recognize the key action among multiple clusters in a video stream. Based on different applications, we may not be able to recognize all the actions of individual clusters due to lack of video samples in training phase. The KAP includes two classifiers to address this issue: the first classifier detects the key and noise temporal clusters in a video and the second classifier recognizes the action of the key clusters. We have manually labeled the key and noise clusters of Hollywood2 and URADL datasets to train the first KAP classifier. We utilize the labels of the key clusters to train the second KAP classifier. The KAP approach obtained competitive performance with the state-of-the-art techniques in benchmark datasets.

1.5 Organization of Thesis

The rest of the dissertation is structured as follows: Chapter 2 consists of an extensive literature review of video analysis algorithms including feature extraction and encoding, classification, and time-series video clustering approaches. The ensemble learning to leverage different video descriptors for action recognition is discussed in Chapter 3. The effect of motion saliency map and hybrid classification system on action recognition is introduced

in Chapter 4. Chapter 5 proposes the hierarchical feature representation along with the improved rank pooling methodology. Finally, the key action perception methodology along with the video action clustering approaches are introduced in Chapter 6 followed by drawing a conclusion and presenting some scopes for future work in Chapter 7.

CHAPTER 2

Related Work and Datasets

2.1 Overview

Human actions and activities contain the simple movements of human body which are called gestures. For instance, playing violin contains a mix of different simple movements in hands. Variety of researchers define the term of an action based on their own intuitions. We follow the proposed terminology in [11] to describe a human action. As mentioned in [11], ”action is the most elementary human-surrounding interaction with a meaning”. The category of each individual action is obtained by analysis of the interaction between the human and its surroundings.

The real-world videos include long and complex temporal contents with various editing artifacts, variable length, and different camera motion conditions [9]. Variety of recent methodologies pay attention to simple and complex human activity recognition for video analysis approaches [12, 13, 14, 15, 16, 17, 18]. Complex actions are composed of a sequence of simple actions, such as long jump, vault, and discus throw [19]. The same class of an action contains significant intra-class appearance and motion variations due to the complicated temporal structure, cluttered backgrounds, viewpoint changes and motion speed variations [19]. Consequently, the representation and classification of simple and complex human actions are still considered as challenging problems in constrained and unconstrained videos. During the past three decades, action recognition researchers first focused on simple actions in constrained conditions, but more attention is now paid to complex actions in realistic videos [19].

Activity recognition is considered as a multi-class classification problem where each target class is assigned with a specific action type. An action recognition system consists

of two major steps: first, the features are selected, described, and encoded. Then, a classification algorithm is employed to categorize actions. In such a system, an efficient feature set is able to reduce the burden of the classification algorithm. On the other hand, a powerful classification algorithm is able to accurately classify actions even with low discriminative feature sets. Computer vision scientists have been working to compare a variety of visual descriptions and representations which are well-suited for video action recognition problems. The performances of these representations are typically evaluated on benchmark datasets [11, 20].

To justify our major contributions and bold the context of our research, this chapter reviews and presents the existing literature on video action recognition. We separate the existing methodologies related to the field of action recognition into four major groups. First, we describe the holistic representations which are derived from the human body architecture, movements and shapes as global information. Second, we review the local representations inspired by edges, colors, trajectories, and interest points from video streams. Third, we review the classification modules to categorize a given video using local or global representations. Finally, we review the available literature to address the video action clustering problems. Altogether, this chapter presents the state-of-the-art methods and describe their advantages and limitations to recognize human activities in videos.

2.2 Holistic Representations

Holistic representations are one of the early approaches used for video action recognition. The methodology of silhouette frames and features has been proposed by [21], and extracts the human shape mask for each individual frame. Then, the ratio of background to foreground pixels is calculated based on the cells of the grid over silhouette. Next, the codebook is generated using the vector quantization algorithm using the grid of the video frames. For categorizing a given video, they utilize the Hidden Markov Models (HMM) to figure out the best model which matches the obtained symbol sequence.

The idea of temporal templates has been proposed by [1] to extract shape masks from individual video frames and then accumulate the differences between frame sequences. The

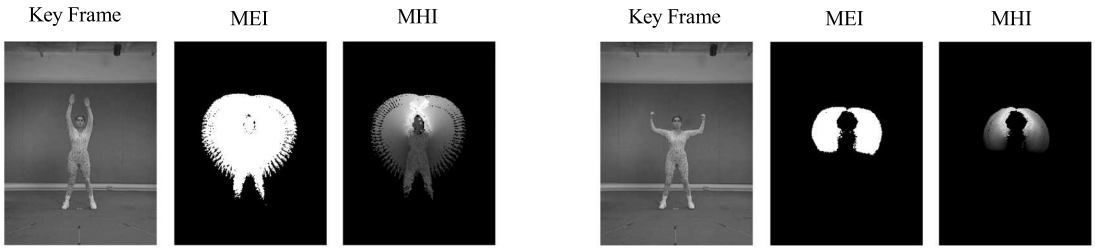


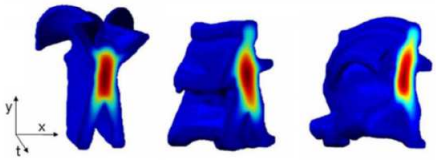
Figure 2.2.1: The motion energy image and motion history image for temporal template matching [1].

achieved information using differences of frame sequences is used to generate the binary motion-energy image (MEI) and scalar-valued motion history image (MHI) as depicted in Figure 2.2.1. The MEI represents the existence of a motion in video frames, and MHI presents the pixel intensities as a temporal history of the motion. Finally, the action of a given video is recognized by temporal template matching against obtained examples of training samples.

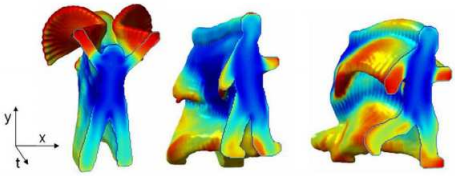
The extraction of three-dimensional shapes using silhouettes has been proposed by [2]. This methodology has five major steps: first, the background subtraction is performed to calculate the silhouette information; second, the silhouettes are concatenated over ten frames to generate a spatio-temporal shape as shown in Figure 2.2.2 (a) ; third, the action dynamics, shape structure, and saliency of that blocks are calculated using Poisson's equation as depicted in Figure 2.2.2; fourth, the blocks of ten frames are matched using the sliding window approach; finally, the actions are recognized using the nearest neighbour classifier with Euclidean distance.

It is worth pointing out that the accuracy of holistic approaches is highly dependent on the segmentation of a video. In the real-world scenario, the segmentation task is very difficult where videos contain long shots including dynamic backgrounds and motion of cameras while capturing frames. From the other side, holistic algorithms are not strong to deal with action variations. For instance, the holistic approaches definitely fail to properly recognize actions if the videos of the same action are captured from different points of views with occlusions. In order to address the limitations of silhouette approaches, the advanced video action recognition frameworks have been shifted to extract local features from a given video. Next section presents the summary of the video local representation

(a)



(b)



(c)

Figure 2.2.2: (a) demonstrates the three-dimensional shapes using silhouettes [2], (b) shows the solution to the Poisson equation on space-time shapes [2], (c) depics the local space-time saliency features [2].

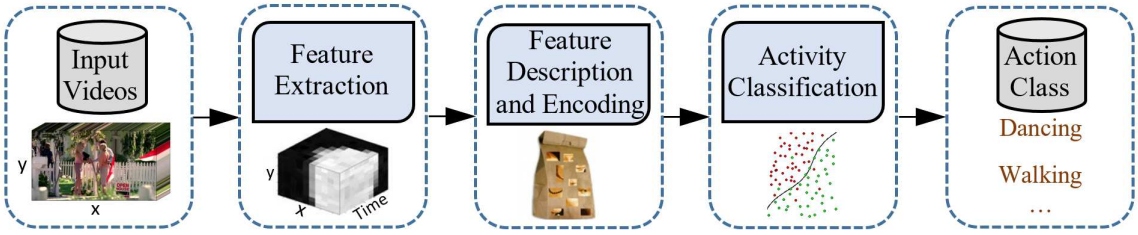


Figure 2.3.1: The general framework of bag of visual words model for video action recognition.

algorithms.

2.3 Local Representations

Local features aim to represent a given video by analysis of spatiotemporal volumes. Analysis of spatiotemporal windows helps to solve the issues of occlusions in videos due to the following reason: local representation approaches usually analyse the entire video for feature selection and description. Consequently, the extracted features from the normal moments with no occlusions are enough to represent a video. Many of the successful action recognition frameworks employ low-level features with bag of visual word (BoVW) framework [22]. The pipeline of BoVW contains four major steps: feature detection, feature description, feature encoding, and classification as depicted in Figure 2.3.1. This section thoroughly describes the state-of-the-art methods to enhance the individual blocks of the BoVW framework.

2.3.1 Feature Detectors

One of the early successful local feature detectors is Harris3D [3] which is considered as the extension of regular Harris detector [23]. The Harris3D is derived from the significant eigenvalues of a spatiotemporal second-moment matrix at each pixel of video frames. The local positive spatiotemporal maxima is employed to define the final corners as interest points in video frames as depicted in Figure 2.3.2. Additionally, Harris3D brings the novelty to video analysis domain by employing separate spatial and temporal scale values [3]. In some extreme cases, the number of detected corners are too rare for video action

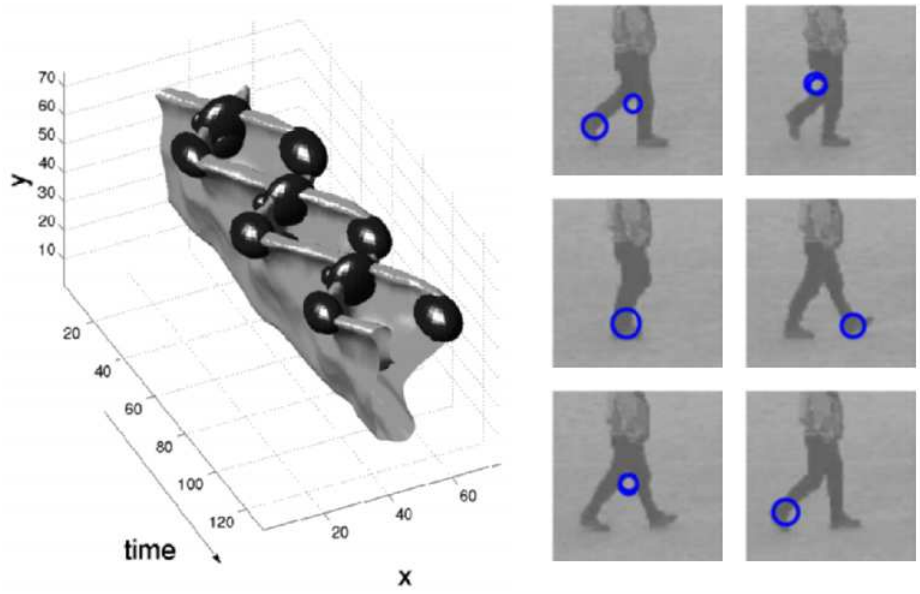


Figure 2.3.2: Detection of spatio-temporal interest points from the movements of the legs while performing walking action [3].

recognition. To address this problem, the Gabor detector [24] has been proposed to obtain denser interest points from video frames. The Gabor detector employs Gaussian kernels on the spatial domain and Gabor filters on the temporal domain of video frames. Finally, the interest points are identified using local maxima of the employed response functions.

The Gabor detector extracts a huge number of interest points for action recognition and video analysis frameworks. However, many of the selected points are irrelevant to human actions and deliver noises to the feature sets. To address this problem, the Hessian3D detector [4] has been proposed to select the most reliable interest points as shown in Figure 2.3.3. The Hessian matrix is computed for each individual interest points and its determinant is used for scale selection and point localization. The obtained interest points are scale invariant and denser than points in Harris3D detector. Additionally, the Hessian3D is much faster compared to the previous detectors due to the usage of integral video by calculating derivatives with the box-filter algorithm.

The dense sampling strategy has been proposed by [25] to analyse the spatiotemporal domains of a video for extracting of reliable interest points. The dense sampling detector extracts a huge number of interest points and then selects the most reliable points for action recognition. The comparison of Gabor detector, Hessian3D, Harris3D, and dense sampling

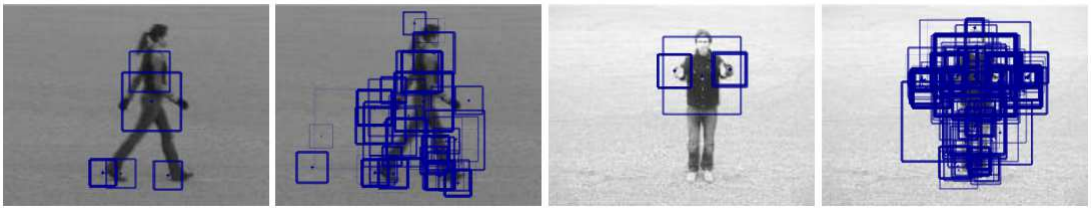


Figure 2.3.3: The results of Hessian3D detector with different thresholding values to extract sparse and dense sets of interest points [4].

have been performed in [25]. As described in [25], it is shown that Harris3D outperforms other feature detectors for simple action recognition. However, dense sampling obtained impressive results for complex action recognition. The major limitation of dense sampling is to extract a huge number of interest points which is computationally very expensive. Aside from that, the moving subjects are ignored and motion features are not considered in dense sampling detector.

To address these limitations, the dense trajectories has been introduced by [5] as shown in Figure 2.3.4. The interest points are extracted and tracked using an optical flow algorithm. The tracked interest points are described by powerful descriptors such as the histogram of oriented gradients (HOG) [26], the histogram of oriented flows (HOF) [27], and motion boundary histograms (MBH) [10] which deliver minor spatiotemporal details of a video. The most widely used local feature extractions are evaluated and analysed in [28, 22]. Based on the mentioned literature, the best current frameworks for human action recognition rely on dense trajectory features [5] that are then encoded by Fisher vector (FV) method. The extracted interest points are supposed to be described for video action classification. Next section presents the advanced feature descriptors for human action recognition in videos.

2.3.2 Feature Descriptors

The appearance and motion characteristics of each detected interest point must be described for video action recognition. One of the early descriptors has been introduced by [29] based on the histogram of spatiotemporal gradients. They employed the higher-order derivatives along with the optical flow methodologies to describe the motion information from indi-

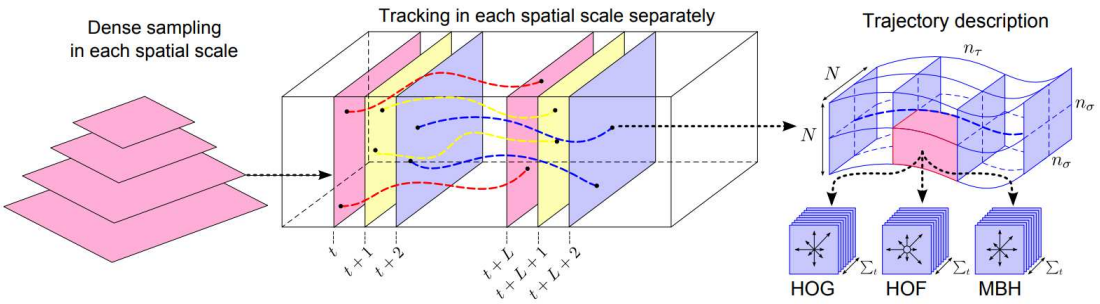


Figure 2.3.4: The dense trajectory structure including HOF, HOG, and MBH feature descriptors [5].

vidual interest points.

The more advanced appearance and motion-based descriptors have been proposed in [27]. These local descriptors are obtained in local cuboids followed by spatial-temporal interesting points (STIP) detectors [30] or dense sampling strategies [5]. Each interest point is divided into a spatiotemporal grid to extract the most informative data from individual local features. The histogram of oriented gradients (HOG) has been proposed by [27] to describe shape information and visual appearance. The histogram bins are quantized by calculating the orientations of edges in video frames. The motion features are described by the histogram of oriented flows (HOF) [27]. The histogram bins are quantized by calculating the optical flows over each interest point. the HOG and HOF are calculated for each individual cell of the grid. Finally, the obtained HOG and HOF descriptors are normalized and merged into the final descriptor. The HOF describes all the moving interest points from video frames. Thus, it cannot address the issue of moving points which are obtained based on moving cameras. To address this limitation, the motion boundary histogram (MBH) has been proposed by [31].

The MBH describes a given interest point by separating the optical flow into vertical and horizontal components as depicted in Figure 2.3.5. The histogram is quantized based on the vertical and horizontal orientation information. In other words, the spatial derivatives are calculated based on vertical and horizontal perspectives. Finally, the obtained histograms are merged to generate the final MBH descriptor. Additionally, the trajectory descriptor has been proposed to extract informative data from dense trajectories [5]. The trajectory descriptors analyse the shape of a trajectory using a set of displacement vectors. It is worth

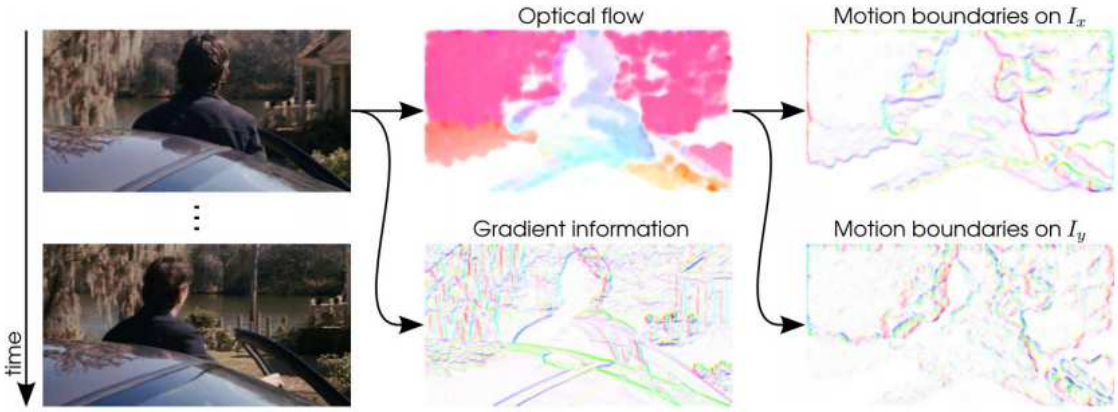


Figure 2.3.5: The results of the MBH descriptor for ignoring the irrelevant moving objects from cluttered background [5].

pointing out that the displacement vectors are normalized by the sum of their magnitudes before generation of trajectory descriptors.

Each individual descriptor represents a given video from different perspectives. Thus, the concatenation of different descriptors as early fusion aims to boost the action recognition performance [31]. However, the early fusion of different descriptors provides huge vectors for the classification stage. We hypothesize that late fusion strategies can address this limitation since separate classifiers are trained over individual descriptors. Then, the score level fusion of single classifiers can enhance the action recognition performance as presented in chapter 3 of this dissertation. The video samples of a dataset are

2.3.3 Feature Encoding

Selection of appropriate encoding method is a significant step to action recognition performance in the BoVW framework [22]. Advanced feature encoding methods have been presented for action recognition, such as vector of locally aggregated descriptors (VLAD) [32], soft-assignment [33], and Fisher encoding [34]. The effective encoding methods for action recognition are evaluated in [22] and it is shown that Fisher encoding outperforms the state-of-the-art strategies. As discussed in [10], the Fisher vector (FV) encoding method is successfully adopted with improved dense trajectory features and obtained promising results on challenging action datasets. The combination of dense trajectories and Fisher vector encoding was first proposed in [35] and obtained state-of-the-art results on several

action datasets. The approach was further modified and enhanced in [36] by employing the stacked FVs. Moreover, Jain et al. [37] showed that the accuracy of this approach could be improved by modeling the context of an action. We adopt FV encoding in this dissertation due to its impressive performance in the field of video action recognition.

The adopted FV algorithm encodes the difference among video features and the vocabulary by applying derivative operations on the likelihood concerning the distribution parameters, (mean (μ), weights (ϖ), and covariance (σ)), of the vocabulary. The Gaussian Mixture Model (GMM) is adapted to shape the vocabulary parameters. The GMM, including K components $\{k_\varsigma, \varsigma = 1...K\}$ is trained to learn the parameters over a subset of training features. However, training a huge number of video descriptors from a dataset is computationally expensive and requires enormous amounts of memory. Thus, it is recommended to randomly select 1000 features from each video descriptor to train the GMM vocabularies. The random selection of features should not affect the learning procedure of GMM vocabularies. Before building vocabulary, principal component analysis (PCA) is applied to reducing the size of the local feature dimension. PCA decorrelates features to support the diagonal covariance assumption since the covariance matrices of the GMM vocabulary are considered to be diagonal. The dimension of the local features is reduced to D using the PCA algorithm before the creation of GMM vocabularies. Finally, the first and second order derivatives are calculated based on the local video features and vocabularies as follows,

$$\bar{u}_{k_\varsigma} = \frac{1}{N\sqrt{\varpi_{k_\varsigma}}} \sum_{i=1}^{N} \gamma_{k_\varsigma i} (\frac{x_i - \mu_{k_\varsigma}}{\sigma_{k_\varsigma}}) \tag{2.3.1}$$

$$\bar{v}_{k_\varsigma} = \frac{1}{N\sqrt{2\varpi_{k_\varsigma}}} \sum_{i=1}^{N} \gamma_{k_\varsigma i} [(\frac{x_i - \mu_{k_\varsigma}}{\sigma_{k_\varsigma}})^2 - 1] \tag{2.3.2}$$

where $x_i \in \Re^{N \times D}$ is the set of features from a given video, N is the number of local features, D is the dimension of features, $\bar{u}_{k_\varsigma} \in \Re^D$, $\bar{v}_{k_\varsigma} \in \Re^D$and $\gamma_{k_\varsigma i}$ is the weight of local features for k_{th} Gaussian. The $\gamma_{k_\varsigma i}$ is obtained by,

$$\gamma_{k_\varsigma i} = \frac{e^{[-\frac{1}{2}(x_i - \mu_{k_\varsigma})^T \sigma_{k_\varsigma}^{-1} (x_i - \mu_{k_\varsigma})]}}{\sum_{\varsigma=1}^{K} e^{[-\frac{1}{2}(x_i - \mu_{k_\varsigma})^T \sigma_{k_\varsigma}^{-1} (x_i - \mu_{k_\varsigma})]}} \tag{2.3.3}$$

where the Gaussian Mixture Model (GMM) links each vector of x_i to a component k in the mixture with a strength calculated by the posterior probability. The final Fisher vector is obtained by concatenation of the first and second order derivatives as $\bar{f} = \{u_{k_\varsigma}, v_{k_\varsigma} | \varsigma = 1...K\}$. In a given Gaussian component, the distribution of features inclines to small values around zero when the number of components increases [34]. In this case, the power and L_2 normalization approaches aim to unsparsify the Fisher vectors [34]. As the power normalization, we apply the signed squared rooting by applying the function $f_{(\bar{f})} = \mathrm{sgn}(\bar{f})\sqrt{|\bar{f}|}$ to each dimension of $\bar{f}$ where $\mathrm{sgn}(.)$ is the $signum$ function [38]. As the L_2 normalization, the obtained vector is further normalized by the L_2 norm [38]. The final dimension of the Fisher vector is $2DK$, where D is the dimension of the set of local features and K is the number of components in GMM training.

2.4 Higher-level Representations

In the computer vision domain, the ultimate goal of action recognition frameworks is to provide informative higher-level and semantic video representations on top of the low-level features. The low-level features deliver local intensity variations such as key points, edges, and gradients from the video frames. However, the higher-level features contain more realistic and structured information based on the whole shape and motion of an action. Consequently, an enormous gap exists between low-level video features and high-level action parts. This gap includes the differences between lower-level features (tiny spatiotemporal details of the video frames) and higher-level representations (more structured information from the whole motion in video frames). Minimizing the gap between low-level and high-level representations aims to boost the action recognition performance in unconstrained and constrained videos.

It should be noted that the standard FV (SFV) efficiently encodes the local features into a high-dimensional space, and aggregates the codes into a huge encoded vector. However, FV is incompetent to directly extract more structured data from a given video. To address this problem, a variety of higher-level representations have been proposed for complex action recognition to mine discriminative action parts [39, 40, 41, 42, 43, 44, 45]. These

methods take the advantages of discriminative spatiotemporal components while having limited representative ability or requiring a high computational cost. Most of the current frameworks train a classifier for each spatiotemporal segment and finally present the outputs of these classifiers as video representations by max-pooling. Consequently, each part produces a single value for the final representation. This procedure limits the power of the higher-level representations. For example, discriminative cuboids by Exemplar-SVM and multiple instances learning frameworks which are adapted to mine discriminative action parts are introduced in [44]. Furthermore, the clustering and ranking algorithms are presented in [41, 40] to extract the discriminative 3D features for video representation. The mined 3D parts are large sub-volumes and contain rich semantic information which is related to action categories. Along the line of these ideas, Peng *et al.* [36] presented a multi-layer hierarchical mid-level methodology to densely encode sampled sub-volumes via Fisher encoding. However, this method is highly expensive to implement due to the encoding each feature sample in the first layer and to consider a massive number of spatiotemporal sub-volumes in the second layer. The mid-level representation based on atoms and phrases is proposed in [46]. Even though the [46] framework obtains promising results, it is highly expensive to implement due to the clustering stage as the first step of the framework.

Furthermore, the deep networks demonstrate an efficient power to produce higher level video representations [47, 12, 48, 49, 50]. Video processing with deep networks is inspired by the success of image representation and classification using deep networks [12, 50]. It is very expensive to train an effective deep multi-layer network for video analysis even though the deep networks can represent videos from the low level to mid and high-level features. Aside from that, deep features include the foreground and background information from image sequences whereas many of the video sections are irrelevant to actions. For instance, zooming, tilting, and rotation of a camera may transfer irrelevant features through deep networks. Thus, the diversity and irregular distribution of features make it difficult to employ deep networks for action recognition directly. In other words, the interest region detection is supposed to be adopted as a preprocessing step in the deep learning approaches to analyse the unconstrained videos efficiently.

We hypothesize that combining the temporal sub-volumes of low-level motion features is capable of providing significant clues to video analysis since an elaborate video always includes multiple finer-grained parts in the temporal domain. Chapter 5 of this dissertation proposes a novel hierarchical strategy, called double phase encoding (DPE), to provide higher-level representations from an unconstrained video. The principal motivation behind this framework is to represent the entire video with higher level information derived from temporal sub-volumes of low-level motion features. Unlike the standard feature encoding where all the features of a given video are encoded in one stage, the DPE methodology encodes a video using two hierarchical phases. At the first phase, the motion features are extracted and then the temporal sub-volumes of low-level features are represented individually. Each represented vector presents specific information related to its temporal sub-volume. At the second phase, a pool of the individual represented vectors (IRVs) is generated by concatenating the IRVs from the first phase. The pool may contain redundant features since the temporal sub-volumes are occluded in different scales. The principal motivation behind compression of encoded features is to extract the most informative data in the second phase of DPE. The compressed pool transfers the most prominent information for higher-level encoding at the second phase of DPE. We target to encode the compressed pool for extraction of the video-parts vector (VPV). The VPV allows distilling the representation and extracting new higher level information from temporal sub-volumes of local features.

2.5 Classification Modules

Based on the nature of action recognition frameworks, we employ the supervised learning algorithms in this dissertation. In the supervised frameworks, the training samples are available along with their annotated labels. The learning models are trained based on the available training samples. Then, the models are validated and tested using the rest of data. The classification is the last stage in a general action recognition framework. Action classification is tremendously challenging for computers due to the complexity of video data and the subtlety of human actions [51, 52].

Nowadays, a variety of classification strategies have been proposed for different applications [53]. For the classification of video-based features, the recent action recognition frameworks employ typical support vector machine (SVM) [54], extreme learning machine (ELM) [55], or neural networks (NN) [5, 10, 20, 28, 22, 56]. Based on the Jaegers estimation [57], most of the classification strategies mainly focus on single-hidden-layer feed-forward networks (SLFN) due to their universal approximation capability. The SLFN strategies include an input layer which receives the data from environments, a single hidden layer where the parameters and weights of the network are calculated, and an output layer which prepares the network outputs. According to conventional NN theories, all the weights and parameters can be adjusted in the SLFN frameworks [58].

The SVMs maximize the distance between a hyperplane that separates two classes of data [54]. Linear and non-linear separations can be performed using a kernel function in the SVM classifier. It is worth pointing out that SVM classifiers avoid ending in a local minimum and reach the global minimum. According to the authors of [54], SVMs are reliable in the speed of classification and robust to deal with irrelevant information in the feature sets. In chapter 5 of this dissertation, we adopt the learning strategy with sub-network nodes (LSN) [59], inspired by [58, 55], to classify the encoded video-based features. Furthermore, we evaluate and compare the LSN performance with standard SVM and ELM classifiers in the video analysis domain.

Employing of a single classifier to classify patterns has been recently challenged by multiple classifier approaches, where the classification system is derived from an ensemble of single classifiers whose outputs are pooled in a way to attain a more accurate final decision. In an ensemble classification approach, every single classifier will focus on diverse aspects of the data and will err under different situations. It should be noted that single classifiers errors have to be uncorrelated in an ensemble classification system [60]. Consequently, the total errors can be reduced by an applicable combination of the single classifiers if different errors are obtained from individual classifiers.

Generally, two methods are used to employ the power of several representations or description techniques. The first method is to merge the extracted feature sets, which is called early fusion, and then to feed this higher dimensional feature set to an individual classi-

fier. The second method is to train several single classifiers using separated feature sets and efficiently fuse the outputs of the single classifiers in a late fusion model. It should be noted that the discriminative power of encoded information cannot be completely utilized while employing individual recognition techniques. In other words, while dealing with difficult action recognition problems, mainly when working with many action types and/ or having resemblance between actions, the single recognition classifiers are not able to accurately categorize actions. Therefore, an ensemble classification framework can be employed to improve the recognition performance, where each combination of a feature set and a classifier is a human action learner. The strategic combination of the learners can significantly enhance the classification accuracy. The joint efficiency of the ensemble of multiple classifiers can compensate for a deficiency in one learner while employing the late fusion techniques over several single classifiers.

Action classification can be very challenging for similar classes in a dataset. For example, equivalent probabilities may be provided for running, jogging and walking classes while classifying the samples of KTH dataset. The classifier is not capable of making the final decision indubitably when equivalent probabilities are generated for different classes. To this end, in chapter 4 of this dissertation, we propose a hybrid classifier to automatically compress the encoded features and select the best SVM kernel for classification.

2.6 Video Action Clustering

The recent action recognition methodologies rely on the bag of visual words (BoVW) framework which contains feature extraction, feature encoding, and classification stages. Most of the BoVW models adopt Fisher vectors of improved dense trajectories [10, 46, 61] or CNN features [49, 7, 62, 63, 64, 65] with a classifier such as support vector machine (SVM), and achieve reliable results on pre-segmented video datasets, such as UCF-101 [66] and HMDB51 [67]. The available action recognition methods work well on fine-grained videos where only one action is available in entire video [10, 49, 68, 7, 61, 46]. Therefore, the action detection can be ignored since most of the video datasets contain single actions in short-shot videos. However, the available action recognition frameworks fail to catego-

rize a long-shot video which consists of multiple actions. To address this problem, the key action cluster must be detected as the pre-processing step to the action recognition frameworks. Chapter 6 of this dissertation proposes key action perception (KAP) framework as a multi-label learning system including two classifiers. Multi-label learning frameworks aim to solve problems where a single sample is represented by a single feature vector and multiple class labels [69]. The first KAP classifier categorizes the obtained temporal clusters into noise and key actions. Then, the second KAP classifier recognizes the content of the key temporal cluster. The video action clustering is the challenging pre-processing task to the KAP framework.

Previous action detection frameworks focus on selection of appropriate action candidates among multiple spatiotemporal sliding windows [70, 71, 72, 73]. These strategies are capable of locating a single action in the short videos. However, in case of having long-shot videos including multiple actions, the mentioned methods fail to segment the video into plausible actions and detect the key action among clusters. Recent approaches utilize deep models for detection of appropriate actions in unconstrained videos [74, 75]. The work [75] proposes a method to predict the staring and finishing frames of an action by employing of the Recurrent Neural Network. The deep networks work well for the short-shot video analysis, but they are very time-consuming for long-shot video analysis due to the dense snippet sampling. The relation between temporal structures can deliver important information for video action clustering and detection. Thus, the segment-convolutional neural network (SCNN) [76] method has been proposed to employ 3D convolution for modeling the temporal structures of a given video. Even though SCNN considers the order of temporal structures, it is limited to detect actions in short-shot videos. From the other side, defining the key action among the available temporal segments is still an open problem. All in all, deep network clustering models are computationally costly and require plenty of data, which is not directly available for some particular applications. Thus, deploying these models on video action clustering and detection is very challenging. We need to make more straightforward models without sacrificing accuracy.

Subspace clustering is a reliable methodology to cluster the videos into plausible actions automatically [49]. Subspace clustering aims to produce coding information for affin-

ity matrix generation. Enforcing different constraints on the coefficients [77], or developing scalable implementations [78, 79] have been proposed to improve the performance of subspace clustering. However, with the notable exception of [80, 81], most of the proposed approaches ignore the specific properties of time-series data. The available mentioned methodologies assume that the data points are independently drawn from multiple subspaces. They independently model the data points [82] or implicitly consider the global structural information in data [83]. However, they neglect the sequential relationships that possibly reside in data. In this dissertation, we aim to move beyond these limitations and develop effective action clustering methods for constrained and unconstrained videos. The proposed methods achieve superior performance on constrained and unconstrained datasets regardless of the complexity of videos.

2.7 Video Datasets

We present experiments in a broad range of datasets with the same training and testing splits as reported by the state-of-the-art methods. Table 2.7.1 enlists the specifications of the employed datasets in this dissertation. Figure 2.7.1 shows some frame samples from the six benchmark datasets including simple and complex actions. The brief description of the datasets is presented below.

Hollywood2 [89]. This dataset contains 12 action categories, 1707 video samples, and approximately 20.1 hours of videos in total. The samples are captured from 69 Hollywood movies and include long and complex temporal contents. We employ the proposed training and testing sets by [89] for evaluation of our methods. The recognition accuracy is measured by mean average precision (mAP) over all classes as described in [89]. The samples of the video frames from this dataset are depicted in Figure 2.7.1(e).

URADL [84]. This is a high-resolution dataset of 15 complex actions. It includes 150 video samples which are captured with a stable camera and background. Thus, this dataset consists of constrained video samples. However, the videos of this dataset contain the longest lengths compared to the other employed datasets. The 10-fold cross validation strategy is used to validate the classification performance of the URADL dataset.

Table 2.7.1: Specification of the employed benchmark datasets to evaluate the proposed methods in this dissertation

Datasets	Year	Number of Videos	Number of Classes	Camera Motion	Background	Video Duration (Ave.)
Hollywood2 [27]	2008	1707	12	Yes	Dynamic	13.73 seconds
URADL [84]	2009	150	10	No	Static	16.16 seconds
Olympic Sports [85]	2010	783	16	Yes	Dynamic	7.69 seconds
HMDB51 [67]	2011	6849	51	Yes	Dynamic	3.12 seconds
UCF-50 [86]	2012	6676	50	Yes	Dynamic	7.00 seconds
UCF-101 [66]	2012	13320	101	Yes	Dynamic	7.21 seconds
KTH [87]	2004	600	6	No	Static	19.34 seconds
Weizmann [2]	2007	90	10	No	Static	2.44 seconds
Keck Gesture [88]	2012	98	14	Yes	Static	17.84 seconds

Olympic Sports [85]. This dataset consists of athletes practicing different sports, which are collected from YouTube and annotated using Amazon Mechanical Turk. The dataset contains 16 classes, represented by a total of 783 video clips. We use standard splits with 649 training clips and 134 testing clips and report mean average precision (mAP) over all classes as recommended in [85].

HMDB51 [67]. This dataset consists of roughly 7,000 videos and 51 action classes. The samples of this dataset are subject to different camera motions, viewpoints, video quality and occlusions. The samples are collected from different sources such as movies and YouTube videos. The samples of the video frames from HMDB51 are shown in Figure 2.7.1(b). We apply the leave-one-split-out cross-validation over three standard splits, as presented in [67]. For every class, there are 70 samples for training and 30 samples for testing. We present the average accuracy over the three train-test splits as the performance measure for the proposed action recognition frameworks.

UCF50 [86]. This dataset includes 6,676 video clips and 50 action classes. The video samples are split into 25 groups and each group contains at least 4 action clips. The same group may share some common features, such as similar background, same person, or similar viewpoint. We apply the leave-one-group-out cross-validation as recommended by the authors in [86] and report average accuracy over all classes.

UCF101 [66]. This dataset is the extension of the UCF50 including 13320 videos and 101 classes, consisting of realistic videos from YouTube. The video samples are grouped into 25 sections while each group consists of 4 to 7 videos of an action. The videos from the same groups may share some common features, such as similar background and viewpoint. We follow the validation protocol, proposed by the authors of UCF101 to evaluate our methods. As presented in [66], three groups of train-test video samples are defined for evaluation of action recognition algorithms. In each split, videos from 18 of the 25 sections are used as training samples, and the rest for testing. The average accuracy of three splits demonstrates the overall performance measure on this dataset.

KTH [87]. KTH is one of the popular video action datasets including six action categories (boxing, hand clapping, hand waving, jogging, running and walking). Each individual action is performed by 25 people. Every single action has been implemented in four

different conditions including indoor, outdoor, variations in scale, and changes in clothing. We follow the leave-one-person-out protocol to evaluate our methods. It means that we train the model using the video samples of 24 people and test the model using the videos of the remained person.

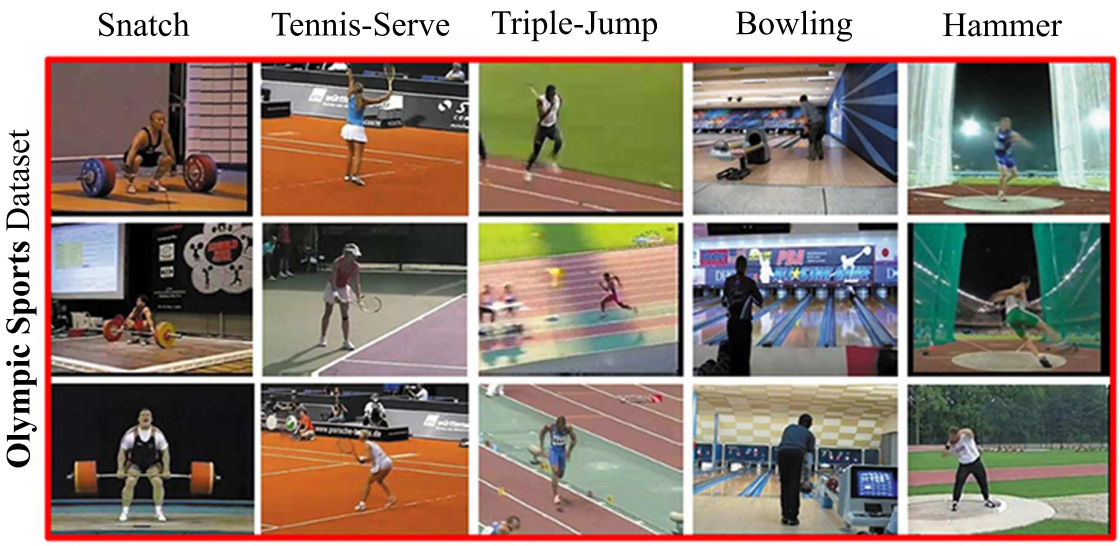
Weizmann [2]. This dataset contains 90 videos and 10 classes of simple actions, captured from 9 subjects. The camera and background are stable while capturing the videos in Weizmann dataset. Thus, we evaluate the proposed video action clustering algorithm to temporal segment the concatenated videos of this dataset.

Keck Gesture [88]. This dataset includes 14 gesture classes in 294 video samples which are derived from a group of military signals. In each video sample, one gesture is iterated for three times. We consider this dataset as a set of unconstrained videos since the camera is moving while capturing some of the samples. We test our proposed video action clustering algorithm to cluster the concatenated unconstrained videos of this dataset.

2.8 Summary

This chapter presented the most relevant and prominent methodologies for human action recognition in videos. As the early approaches, holistic representations have been summarized along with their limitations. The holistic approaches assume to see the videos with stable backgrounds. In other words, the camera must be stable and videos are supposed to be captured in a constrained condition for holistic representations. Otherwise, holistic approaches fail to recognize human actions in unconstrained videos where the background is sophisticated and the camera is moving while capturing video sequences.

Next, the local features along with the robust descriptors and encoders have been reviewed in this chapter. It is shown that local features provide state-of-the-art results compared to other approaches. Aside from that, we need less computational powers to recognize human actions using local features compared to the deep and more sophisticated models. All in all, the local representations based on bag-of-visual-words frameworks are robust and fast to video action recognition and even higher-level information can be extracted on the top of local features. However, in some extreme cases, the local representations fail to


Snatch
Tennis-Serve
Triple-Jump
Bowling
Hammer
Olympic Sports Dataset

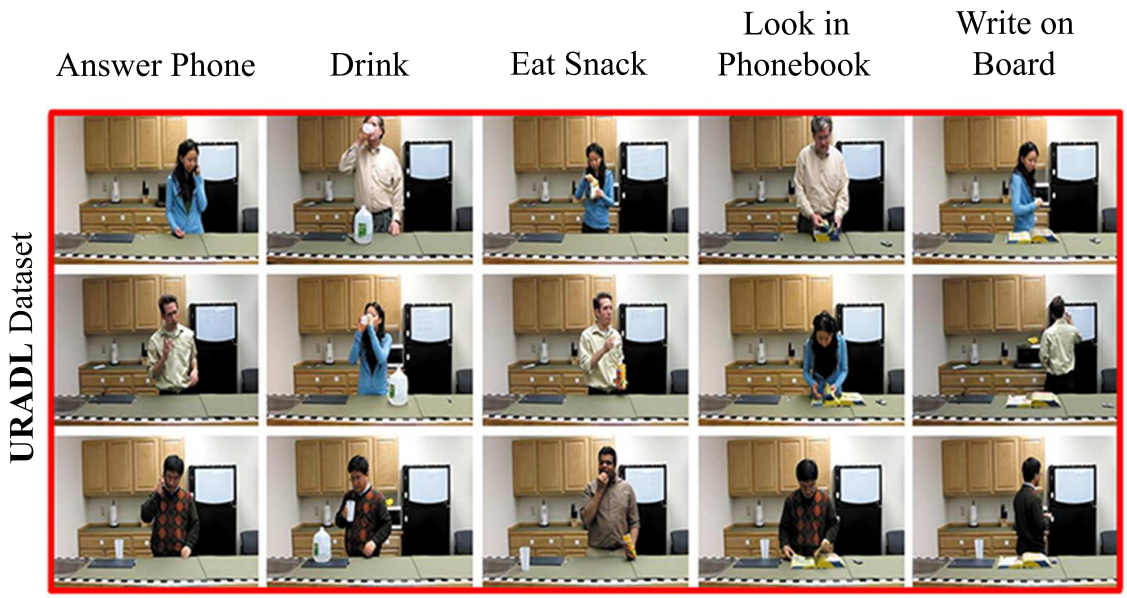
(a)


Dive
Eat
Golf
Hug
Laugh
HMDB51 Dataset

(b)


Basketball
Biking
Horse Riding
Parade
Salsa
UCF50 Dataset

(c)


Answer Phone
Drink
Eat Snack
Look in Phonebook
Write on Board
URADL Dataset

(d)

Figure 2.7.1: The samples of video frames from six benchmark datasets to evaluate the action recognition frameworks.

properly recognize human actions. For instance, it is difficult to address action variations using general local representations. In more extreme cases when the videos contain several actions, the local representations cannot provide enough clues for discriminating actions. We target to enhance the action recognition performances using local representations in constrained and unconstrained conditions.

We tackle the video analysis and action recognition problems in four chapters as follows. Chapter 3 of this dissertation evaluates different fusion techniques to leverage motion and appearance-based features for action recognition in benchmark datasets. Chapter 4 presents the novel hybrid classifier to deal with the action variation problems in constrained videos. Chapter 5 presents the hierarchical feature representation which carries the low-level and higher-level information for action recognition. Finally, chapter 6 presents the key action perception along with the robust video action clustering frameworks for constrained and unconstrained video action recognition.

CHAPTER 3

Effect of Ensemble Learning on Simple and Complex Action Recognition

3.1 Overview

The fusion of different features and descriptors aims to enhance the action recognition performance since each individual descriptor provides specific motion or appearance-based information from a given video. There are two general approaches to leverage the power of different video features and descriptors. As the first approach, we merge the obtained video features and then feed this higher dimensional feature set to a single classifier. The concatenation of individual features provides the huge set of data which makes the classification very challenging and time-consuming. In order to address this problem, we hypothesize that combining of different features can be performed as the late fusion format where the single classifiers are trained using the individual feature sets and then the scores of classifiers are fused to make the final decision. This chapter focuses on the late fusion strategy and evaluates its effect on action recognition datasets.

The performance of pattern classification using a single classifier has been recently challenged by employing of multiple classifier paradigms [90]. The underlying idea of the ensemble of classifiers is that instead of employing a very sophisticated representation or learning technique, we can learn action categories using a set of relatively simple and diverse classifiers, each trained using an individual feature set. In an ensemble classification system, it is hoped that each base classifier will focus on different aspects of the data and will err under different situations. Thus, the total errors can be reduced by fusion of scores from single classifiers.

In this work, as shown in Figure 3.2.1, the encoded data from the individual descriptors (HOG, HOF, MBH, and Trajectory) are fed to the single classifiers separately. Then, the outputs of single classifiers are fused to classify actions. A combination function is employed to merge the outputs of single classifiers. The most common combination functions for fusing the single classifiers have been presented in [60]. This chapter adopts the Dempster-Shafer (DS) fusion method and the algebraic combiners to synthesize the outputs of single classifiers. The stated fusion methods make an ensemble of classifiers by observing the output of single classifiers as a measure of evidence. The following sections thoroughly describe the ensemble learning framework and extensive experiments on benchmark action recognition datasets.

3.2 Ensemble Learning Framework for Action Recognition

As the first step of the ensemble learning framework, we extract the video features using the improved dense trajectory (IDT) method [10]. In the IDT, each frame of the video is analysed to sample the points densely from a multi scale pyramid. Then, the sampled points are tracked for a given time window. It should be noted that the employed tracking is based on dense optical flow field computation [91] and is applied on each spatial scale separately. For each trajectory, three descriptors (HOF, MBH, and HOG) are computed in the space-time volume with exactly the same parameters as stated in [10]. HOF and MBH are based on optical flow and capture motion information. The orientations of flow vectors are quantized using the HOF descriptor. However, MBH divides the optical flow into horizontal and vertical components. Then, the derivatives of each component are computed by MBH descriptor. HOG is based on the orientation of image gradients, and describes the information from static appearance. The feature dimensions of the HOG, HOF, and MBH for x and y axis are 96, 108, 96 and 96 respectively as stated in [10].

The ensemble learning of the proposed activity recognition framework consists of the following blocks as shown in Figure 3.2.1: random subset selection (RSS), principal com-

ponent analysis (PCA), Gaussian mixture model (GMM), and Fisher vector encoding (FVE) where τ stands for the four separated descriptors (HOG, HOF, MBH, and Trajectory); γ stands for the PCA projection coefficients; δ stands for the GMM vocabularies for four descriptors; ψ and φ respectively refer to the training and testing feature sets, encoded by the fisher vector. Due to the huge dimension of video features, it is impossible to employ all the features for the training of GMM. As the first step, we randomly select a set of features from the descriptors and then reduce their dimension using PCA to train the GMM for Fisher vector encoding. The random selection of features is not affecting the results of GMM training. The single SVM classifiers are trained using the feature sets that are encoded by the Fisher vector representation. Then, the outputs of single SVM classifiers are fused to make the final decision about the test samples. As presented in the following sections, we use the DS and algebraic fusion methods to synthesize the scores of single classifiers.

3.2.1 Dempster-Shafer Fusion

Many fusion strategies are inspired from Dempster-Shafer (DS) theory of evidence which adopts belief functions instead of probabilities to quantify the evidence from each sample. The belief functions are synthesized by the Dempster's rule of combination to make the final decision about a testing sample. In the DS, the beliefs in a hypothesis are calculated as the sum of the subjective probabilities of all classes it encloses. In other words, the DS allows combining evidences from different sources and arriving at a degree of belief (represented by a belief function) that takes into account all the available evidences.

As shown in Figure 3.2.2, a decision profile is employed to discover the overall support for each class and then to label the test sample in the class with the largest support. Considering $X \in \Re^n$ to be a feature vector and $\Omega = \{\omega_1, \omega_2, \omega_3, ..., \omega_c\}$ to be the set of class labels, each classifier D_i in the ensemble $\mathrm{E} = \{D_1, D_2, ..., D_L\}$ results c degrees of support. All defined c degrees of support are considered to be in the interval [0, 1]. Classifier D_i defines that X comes from class ω_j by showing the $d_{i,j}(X)$ support. The class label ω_j is assigned to an instance, if the support of that class is the largest one compare to other

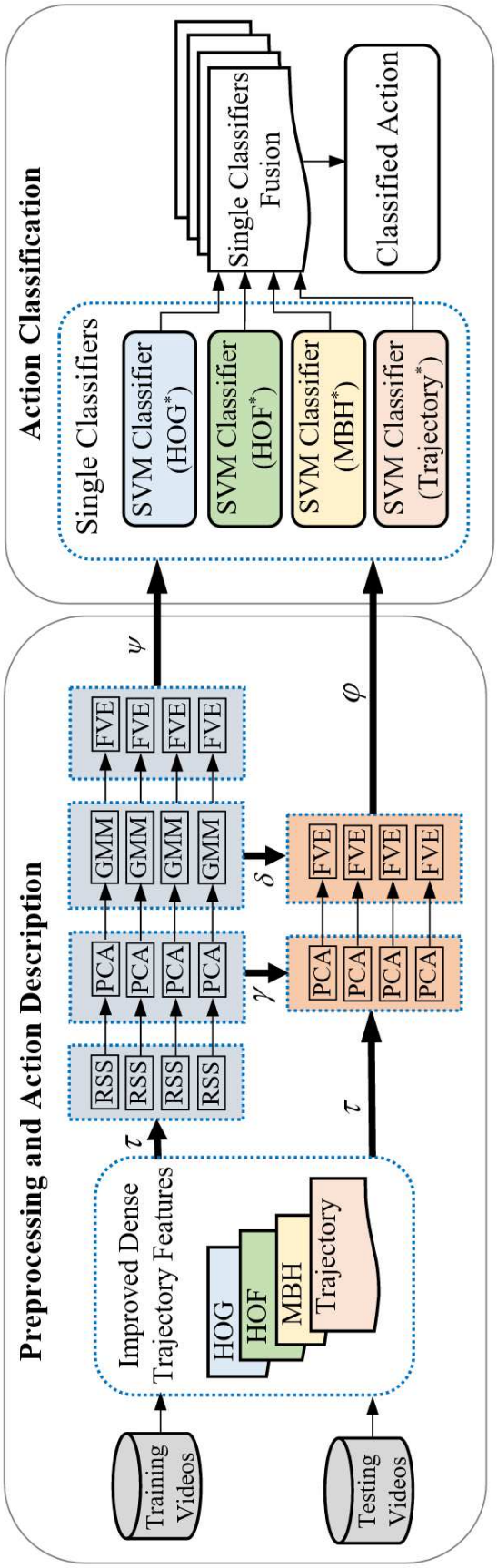


Figure 3.2.1: The ensemble learning framework to leverage the features of different perspectives.

supports. Figure 3.2.2 presents the decision profile $(DP(X))$ from the L classifier outputs for a particular instance X where α refers to the available classes and β refers to the supports from L trained classifiers $D_1, ..., D_L$ for the class ω_j. The following four stages are performed to predict the target class of each test sample using the DS fusion strategy.

As the first step, the decision templates are made by calculating the means of the decision profiles for all training samples belonging to ω_j. We obtain the decision templates DT_j for $j = 1, ..., c$ as:

$$DT_j = \frac{1}{N_j} \sum_{k=1}^{N_j} d_{i,j}(X_k) \tag{3.2.1}$$

where N_j refers to the number of training samples belonging to the class of ω_j.

As the second step, we calculated the proximity between the decision templates and the output of classifiers:

$$\phi_{j,i}(X) = \frac{(1 + ||DT_j^i - D_i(X)||)^{-1}}{\sum_{j=1}^{c} (1 + ||DT_j^i - D_i(X)||)^{-1}} \tag{3.2.2}$$

where DT_j^i denotes the i_{th} row of the decision template DT_j, and D_i refers to the output of the i_{th} classifier (the i_{th} row of the decision profile DP(x)). The $||.||$ is the Frobenius matrix norm in Equation 3.2.2.

As the third step, the belief degrees are computed for each class $j = 1, ..., c$ and classifier $i = 1, ..., L$ to show how a test sample is correctly assigned into a given class ω_j by a given classifier. The following equation calculates the belief degrees of a given testing sample:

$$B_{j,i}(X) = \frac{\phi_{j,i}(X) \Pi_{k \neq j}(1 - \phi_{k,i}(X))}{1 - \phi_{j,i}(X)[1 - \Pi_{k \neq j}(1 - \phi_{k,i}(X))]} \tag{3.2.3}$$

Finally, the Dempster rule is applied to combine the belief degrees which are derived from each of the single classifiers. Based on the Dempster rule, the belief degrees from L classifiers must be multiplied to achieve the final support for each class:

$$\mu_j(X) = \prod_{i=1}^{L} B_{j,i}(X), j = 1, ..., c \tag{3.2.4}$$

where the final class is assigned to the testing sample by considering the largest $\mu_j(X)$.

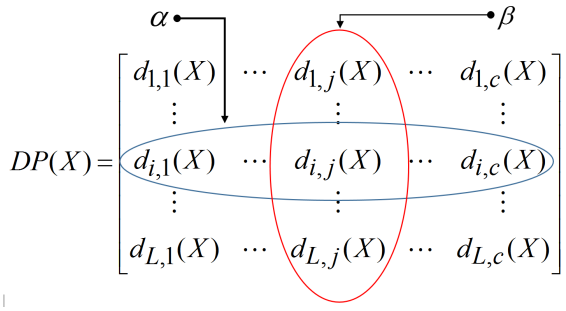


Figure 3.2.2: The form of the decision profile in DS fusion method.

3.2.2 Algebraic Combiners

We utilize the mean, maximum, and product rules as the algebraic combiners to fuse the outputs of single classifiers. In the product rule, the total support for each class is calculated using a simple algebraic function over the scores of single classifiers. As stated in Figure 3.2.2, in the decision profile, each column represents the supports from separated classifiers and each row shows the outputs of each single classifier for classes $j = 1, .., c$. The total obtained support for class ω_j is obtained from the column j of the decision profile as follows:

$$\mu_j(X) = F[d_{1,j}(X), ..., d_{i,j}(X), ..., d_{L,j}(X)] \tag{3.2.5}$$

where F is the following function called product combination rule:

$$\mu_j(X) = \frac{1}{L} \prod_{i=1}^{L} d_{i,j}(X) \tag{3.2.6}$$

The product rule selects the class whose product of supports from each classifier is the highest. It should be noted that the product rule decimates any class that obtains at least one zero or very small support due to the nulling nature of multiplying by zero.

For the mean rule, the support for a given class is the average of all classifiers' outputs for that given class. Thus, the F function is considered as an averaging function for the mean rule fusion. The final decision is the class ω_j for which the total support $\mu_j(X)$ is the highest. The maximum rule simply takes the maximum among the classifiers' individual outputs where the ensemble decision is chosen as the class for which total support is largest.

3.2.3 Classification

A set of visual feature sets are extracted using the most popular state-of-the-art descriptors: Histogram of Oriented Gradients (HOG), Histogram of Optical Flow (HOF), and Motion Boundary Histograms (MBH) for x and y axis. The extracted feature sets are encoded using the Fisher Vector (FV) encoding method. The employed codebook size for the Gaussian mixture model in the FV encoding approach is 128. Then, the action learning models are trained by feeding the individual encoded feature sets to single classifiers. For the classification, the support vector machines with a cost of 100 and the linear kernel have been used as the single classifiers. Next, the outputs of single classifiers are fused using the Dempster-Shafer, product rule, mean rule, and maximum rule combiners. It should be noted that the individual classifiers are trained using different feature sets with a variety of dimensions in both feature space and sample length. As a result, it is shown that the derived predictions from single classifiers can be combined to boost the recognition performance.

Five models of single classifiers are trained using different feature sets to create the ensemble of classifiers. The models utilize different combination of individual feature sets to train single classifiers. The form of combination is thoroughly presented in the next section.

3.3 Experiments

In this chapter, the issue of automatic recognition is addressed for human action recognition via supervised learning. It means that for every training video we know which action or actions it contains. The action models are learnt using training videos, and then these actions are recognized in new, unseen videos, i.e. videos for which we do not have annotations. This section describes the experimental setup and results for the proposed action recognition framework. The UCF 101, UCF Sports, URADL, and Weizmann datasets are used to evaluate the effect of ensemble learning on action recognition.

3.3.1 Experimental Setup

The action recognition is performed using the following two models: first, as the early fusion method, the individual feature sets are merged to create a higher dimensional feature set. Then, the higher dimensional feature set is fed to a single SVM classifier to recognize actions. Second, as the late fusion method, the ensemble of classifiers are trained using the five extracted feature sets (HOF, HOG, MBH on x and y axis, and trajectories). The five ensemble models are presented as follows.

As the first model, five individual feature sets are trained using five SVMs and the scores of single classifies are fused for action recognition.

As the second model, four separated feature sets (MBH on x and y axis, HOG, and HOF) are trained by four single SVMs. Then, the scores of single classifiers are synthesized using the DS, product, mean, and maximum fusion methods. It should be noted that we ignore the trajectory descriptor due to its inconsistency in ensemble approaches. In other words, trajectory descriptor cannot provide any useful information to the ensemble models.

As the third model, the MBH features on x and y axis are merged to yield a single MBH feature set. Then, the HOG, HOF, and MBH feature sets are trained using three single SVMs. The scores of three single SVM are fused to assign a label to a given video.

As the fourth model, the HOG, MBH-x, and MBH-y feature sets are merged and with single HOF feature set have been used to train two single classifiers. The score fusion of two classifiers makes the final decision about a given video.

Finally, the HOF and HOG are merged to generate a single feature set. Next, the single SVMs are trained by the new HOF-HOG feature set and MBH. Then, the outputs of the stated single classifiers are fused to categorize a video into a proper class.

3.3.2 Experimental Results

The Weizmann, UCF Sports, and URADL datasets are evaluated using the Leave-One-Out cross-validation scheme. This scheme takes out one sample video for testing, and trains using all of the remaining videos of an action class. This is implemented for all the sample videos in a cyclic manner, and the overall accuracy is calculated by averaging the accuracy

Table 3.3.1: Accuracies of Action Recognition using Single Classifiers

Method	URADL	UCF101	UCF Sports	Weizmann
Trajectory and Single SVM	78.00%	41.55%	67.8%	76.5%
HOG and Single SVM	83.33%	74.84	76.00%	84.95%
HOF and Single SVM	90.00%	78.88	80.29%	**92.47%**
MBH-x and Single SVM	87.33%	77.82%	75.29%	**92.47%**
MBH-y and Single SVM	88.67%	78.15%	70.29%	78.49%
Early Fusion and Single SVM	**94.00%**	**85.06%**	**83.86%**	**92.47%**

of all iterations. However, for the UCF 101 dataset, the proposed evaluation method in [14] has been employed to calculate the action recognition accuracy.

Table 3.3.1 shows the results of action recognition performances over UCF101, URADL, UCF Sports, and Weizmann datasets using individual feature sets and early fusion method. It is shown that the early fusion can enhance the performance of the video features for the task of action recognition. However, training and testing of the single classifiers using the early fused vectors are time-consuming and challenging due to the huge dimension of fused vectors. To address this problem, the ensemble learning framework is proposed to deal with different motion and appearance-based features.

Figure 3.3.1 shows the accuracies of the five ensemble models for the URADL and UCF101 datasets. As shown in Figure 3.3.1, the highest accuracies for the ensemble approaches are attained by training two single SVM classifiers based on HOF and merged of HOG-MBH descriptors as the fourth ensemble model. Then, the scores of these single classifiers are fused using the product rule and DS fusion. As shown in Figure 3.3.1, the product rule of the fourth ensemble model outperforms the DS fusion and other ensemble models.

The attained recognition accuracies using Dempster Shafer, mean, maximum, and product fusion methods on the UCF101, UCF Sports, Weizmann, and URADL datasets are presented in Table 3.3.2. In addition to the fusion methods, the accuracies of individual classifiers, each trained on separated feature sets, are shown in Table 3.3.1. For each dataset, the highest achieved accuracy is highlighted in both tables. The highest accuracy for the ensemble approaches are attained by training two single SVM classifiers based on HOF and HOG-MBH descriptors, and fusion the outputs of these single classifiers using

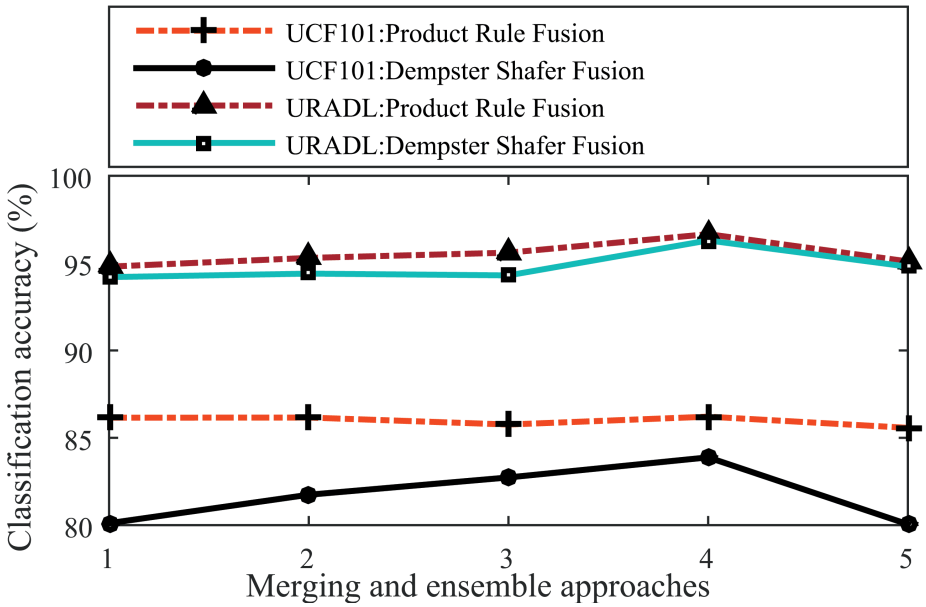


Figure 3.3.1: The attained accuracies using five types of merging and ensemble approaches for UCF101 and URADL datasets

the product rule approach. It must be noted that the score level fusion based on the product rule combination has remarkably improved the results for the benchmark datasets due to the decimation of any class that obtains at least one zero or very small support respect to the nulling nature of multiplying by zero. Furthermore, the mean rule fusion over four individual descriptors obtains comparable results to product rule fusion. However, we conclude that the product rule fusion outperforms other combination strategies since we only train two classifiers based on the fourth ensemble model and product rule fusion. Thus, it is more efficient to employ product rule fusion and the fourth ensemble model to enhance the processing time, required power, and action recognition accuracies.

3.4 Summary

In this chapter, the issue of automatic action recognition is addressed via supervised ensemble learning. The performance of human action recognition is enhanced by utilizing the Fisher vector representation and improving the classification module. Each of the single classifiers are trained over individual feature descriptors that are encoded by the Fisher vector approach. The outputs of single classifiers are fused using the DS and algebraic

Table 3.3.2: Accuracies of Action Recognition using Ensemble of Classifiers

Ensemble Model	Feature Sets	Fusion Method	UCF101	UCF Sports	Weizmann	URADL
1	HOG, HOF, MBH-x, MBH-y, Trajectories	DS	78.15%	80.29%	92.47%	90.0%
2	HOG, HOF, MBH-x, and MBH-y	DS	84.42%	83.14%	**96.77%**	94.0%
3	HOF, HOG, MBH	DS	81.74%	83.14%	**96.77%**	**96.66%**
4	HOF and Merged HOG-MBH	DS	83.88%	81.71%	95.70%	96.30%
5	MBH and Merged HOF-HOG	DS	82.73%	81.71%	93.55%	96.30%
1	HOG, HOF, MBH-x, MBH-y, Trajectories	Mean Rule	78.15%	81.00%	91.40%	90.0%
2	HOG, HOF, MBH-x, and MBH-y	Mean Rule	**86.21%**	**84.57%**	**96.77%**	96.30%
3	HOF, HOG, MBH	Mean Rule	85.06%	81.71%	96.77%	94.0%
4	HOF and Merged HOG-MBH	Mean Rule	85.72%	82.43%	95.70%	94.0%
5	MBH and Merged HOF-HOG	Mean Rule	84.92%	82.43%	91.40%	94.0%
1	HOG, HOF, MBH-x, MBH-y, Trajectories	Maximum Rule	78.15%	80.29%	92.47%	90.0%
2	HOG, HOF, MBH-x, and MBH-y	Maximum Rule	84.92%	81.00%	93.55%	96.30%
3	HOF, HOG, MBH	Maximum Rule	84.02%	81.71%	94.62%	94.0%
4	HOF and Merged HOG-MBH	Maximum Rule	85.16%	82.43%	95.70%	96.30%
5	MBH and Merged HOF-HOG	Maximum Rule	84.42%	81.71%	92.47%	96.30%
1	HOG, HOF, MBH-x, MBH-y, Trajectories	Product Rule	81.74%	81.71%	92.47%	90.0%
2	HOG, HOF, MBH-x, and MBH-y	Product Rule	83.88%	83.86%	**96.77%**	96.30%
3	HOF, HOG, MBH	Product Rule	86.16%	84.71%	95.70%	94.0%
4	HOF and Merged HOG-MBH	Product Rule	**86.21%**	**84.57%**	**96.77%**	**96.66%**
5	MBH and Merged HOF-HOG	Product Rule	85.76%	81.71%	91.40%	90.0%

combiners to improve the performance of the action recognition. The performances of early and late fusion approaches are compared in the extensive experiments. Following the multiple classifier philosophy, the experiments demonstrate that the fourth proposed ensemble approach based on the product rule fusion outperforms standard non-ensemble strategies and other fusion methods for action recognition. It is worth pointing out that the fourth learning ensemble includes two single classifiers which are trained over HOF and HOG-MBH descriptors.

CHAPTER 4

Effect of Hybrid Classifier on Action Recognition Performance

4.1 Overview

Any action dataset may contain similar classes such as running, walking and jogging. Therefore, equivalent probabilities may be provided for different classes upon action classification. In this case, the classifier cannot indubitably assign a class to a given sample. To address this problem, we propose a new hybrid classifier to automatically compress the features and classify them using SVM with polynomial or sigmoid kernels. Furthermore, we hypothesize that motion saliency detection can strength the power of motion feature extraction in the bag of visual words framework (BoVW). To this end, we evaluate the effect of 3D-discrete wavelet transform (3D-DWT), as the preprocessing step, on motion feature extraction. The experimental results show that the proposed framework achieves promising results on KTH, Weizmann, and URADL datasets, and outperforms recent state-of-the-art approaches.

As depicted in Figure 4.2.1, the proposed hybrid classifier is composed of three layers. We hypothesis that huge vector of encoded features may transfer redundant info and outliers to classifier. Thus, in case of providing equivalent probabilities in the first layer, we compress the encoded features to d dimension in the second layer, and then pass the compressed features to the third layer. The major motivation behind data compression is to extract the most useful and prominent information while reducing the dimension of data. In the third layer, the system chooses the best kernel among polynomial and sigmoid functions for SVM classifier. The experimental results of the proposed framework

show a significant improvement over traditional SVM classifier for action recognition. In summary, this chapter makes the following contributions.

1) Applying 3D-DWT on videos to extract motion saliency maps. Different thresholding values are evaluated to extract the best motion saliency map for local feature extraction. The effect of 3D-DWT on motion-based features is evaluated in this chapter.

2) Proposing a hybrid classifier to automatically compress the extracted features and select the best SVM kernel for action classification.

The remainder of this chapter is organized as follows. Section 4.2 thoroughly presents the architecture, formulation, and implementation of our framework for action recognition. The experiments and results are described in Section 4.3. Finally, Section 4.4 summarizes the proposed method and experimental results.

4.2 Proposed Framework

In this chapter, we employ the 3D-discrete wavelet transform (3D-DWT), as a preprocessing step, in the BoVW model. Moreover, we propose a hybrid classification system to confidently classify human actions. The proposed framework is depicted in Figures 4.1.1 and 4.2.1, and discussed in the following subsections.

4.2.1 Preprocessing and Feature Extraction

The three dimensional discrete wavelet transform (3D-DWT) can be considered as a combination of three 1D-DWT in the x, y and t directions [92]. It is composed of high-pass and low-pass filters that perform a convolution of filter coefficients and input pixels. After applying of 3D-DWT, the volume of image sequences is decomposed into 8 sub-signals. We employ the sub-band which is generated with high-pass filters in three directions. We first resize the image sequences to 500×500 and then apply 3D-DWT on resized videos. The extracted sub-signal, which is composed of high-pass filters to each direction, is converted to video with 10 frames per second. Then, the motion saliency map is generated by

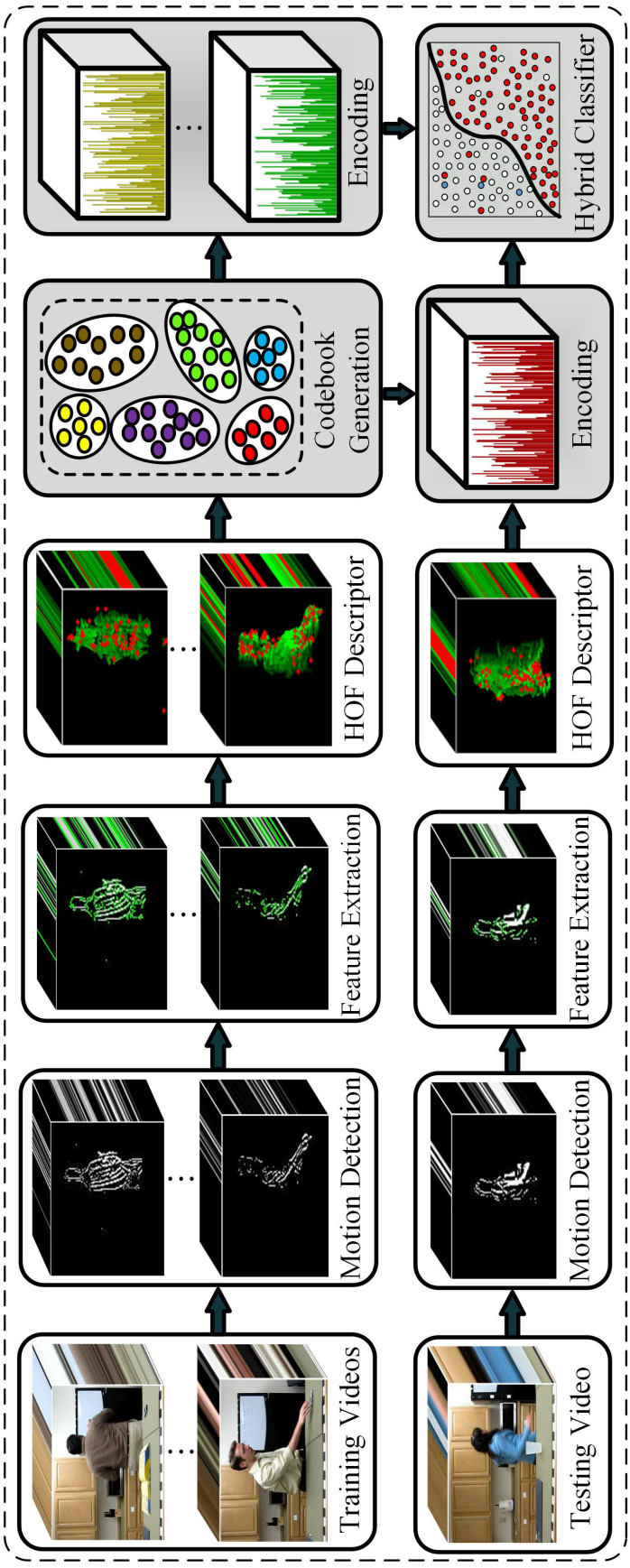


Figure 4.1.1: The enhanced action recognition framework by adding the pre-processing and hybrid classifier stages to the BoVW model.

applying a threshold of (θ) on created pixels. We evaluate different thresholding values to provide the best motion saliency map.

We hypothesize that only the motion features can provide enough information to recognize actions from the videos which are captured with static cameras. To this end, the dense trajectory features [5] are extracted from preprocessed videos and then described by histogram of optical flow (HOF). The HOF describes the local motion by defining a grid around the encompassing space-time area and computing a histogram of optical flow for each cell of the grid [27]. The HOF description is performed faster on motion saliency maps compare to the raw videos. The described features are encoded by Fisher vector [38], and then fed to the proposed hybrid classifier. The dimension of the encoded features is $2DK$ where D is the dimension of the initial features and K is the codebook size while encoding the features. We further call the extracted motion features as W-HOF since the wavelet is employed to extract the motion saliency maps before HOF extraction.

4.2.2 Hybrid Classification

Equivalent probabilities may be provided for similar action categories while classifying a given sample. In this case, the classifier cannot confidently categorize the actions. In order to address this problem, we propose a novel hybrid classifier to use the appropriate SVM kernel when equivalent probabilities are generated by linear SVM for multiple classes. Our proposed hybrid classifier is composed of the following three layers as depicted in Figure 4.2.1.

First layer. The encoded features are fed to the SVM classifier with linear kernel. The one-vs-all strategy is followed and the linear SVM is trained for multiple classes. For a given sample, the set of generated probabilities, $P = [p_1, \ldots, p_m]$, is checked and evaluated to figure out whether the maximum probability is confidently assigned to its class. The thresholding value τ is obtained as

$$\tau = \max_1(P) - \max_2(P) + \tfrac{1}{m} \tag{4.2.1}$$

where m is the number of classes and P is the set of probabilities, generated in one-vs-

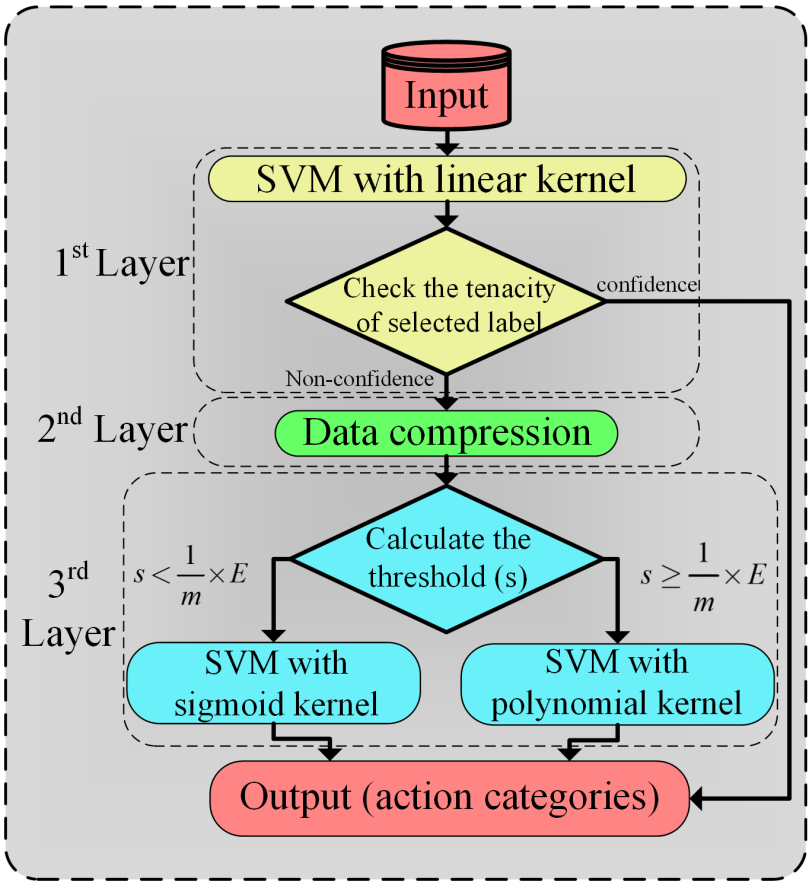


Figure 4.2.1: The proposed hybrid classifier for action classification.

all mode. The $\max_1(P)$ and $\max_2(P)$ are the first and second maximum values in set of probabilities which are generated by linear SVM. In case of providing $\tau \leq \max_1(P)$, the result of linear SVM is considered as non-confident and the features are passed to the second layer. Otherwise, the final decision is made based on the $\max_1(P)$ which is generated by linear SVM.

Second layer. The double-layer net with sub-network nodes (DL-SNN) [93] is employed to generate the compressed version of encoded features. The major motivation behind the usage of DL-SNN is to extract the most useful features and remove the redundant info from data. The features are compressed to d dimension and then transferred to the third layer.

Third layer. The experiments demonstrate that SVM with sigmoid and polynomial kernels obtain different recognition performances based on the compressed features. Therefore, in the third layer, the SVM classifier with polynomial or sigmoid kernels is adopted to classify the compressed features which are inherited from the second layer. The sigmoid and polynomial kernels are selected based on the following conditions:

$$\begin{aligned} Sigmoid: &\quad \max_1(P) - \max_2(P) < \tfrac{1}{m} \times E \\ Polynomial: &\quad \max_1(P) - \max_2(P) \geq \tfrac{1}{m} \times E \end{aligned} \tag{4.2.2}$$

where E denotes the constant threshold in classifying the samples of three employed datasets. It is worth pointing out that we train three models during the training stage. the first model is created by SVM with linear kernel over the original encoded features. The second and third models are created by SVM with polynomial and sigmoid kernels over the compressed encoded features. The three models are trained and evaluated by libsvm library [94]. During training of three models the cost is set to 100 and the rest of the parameters remain as defaults in libsvm package [94].

Compression Stage

The architecture of DL-SNN for compressing the encoded features at the second layer of hybrid classifier is shown in Figure 4.2.2. The section (a) of Figure 4.2.2 demonstrates the

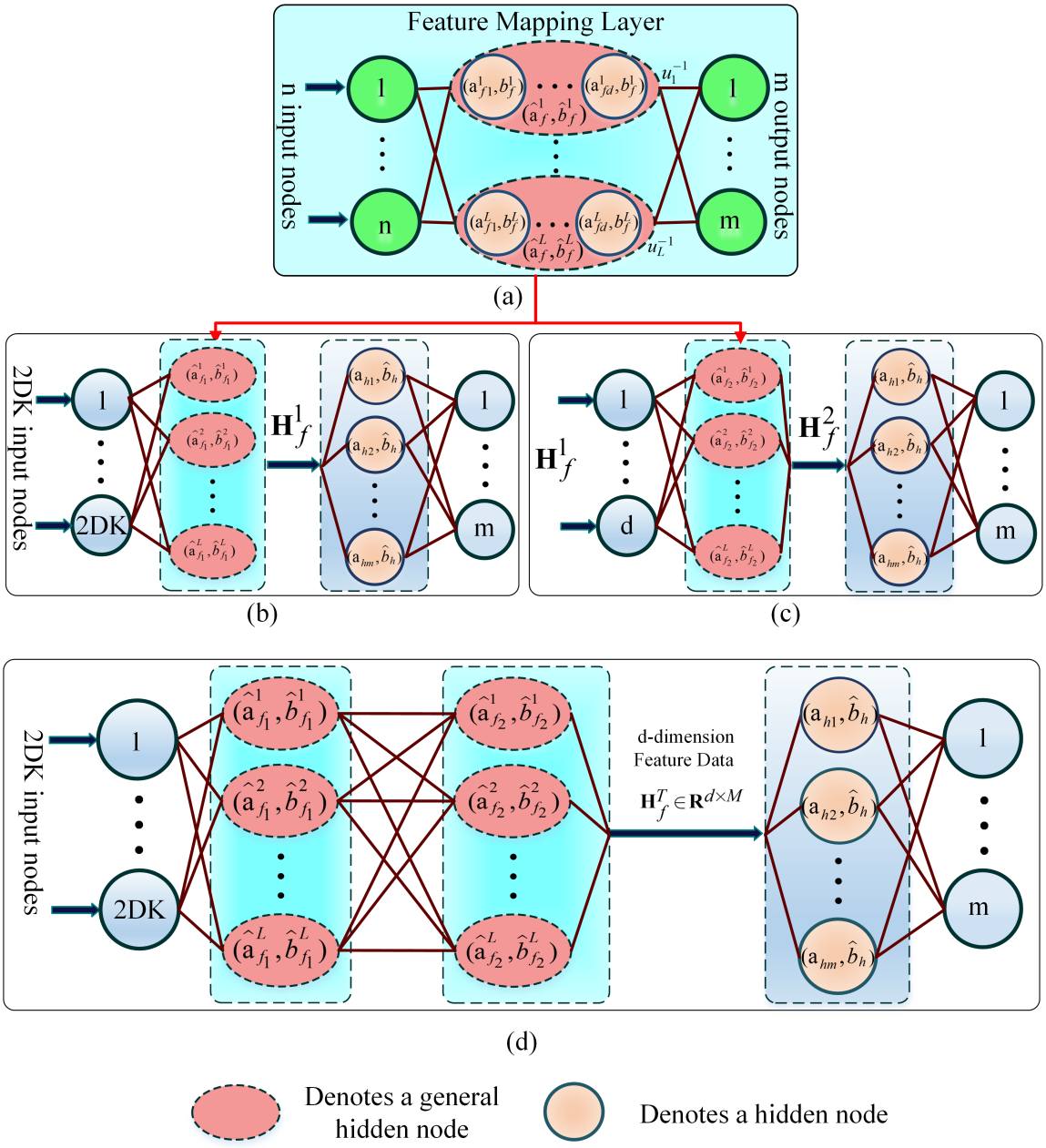


Figure 4.2.2: Structure of DL-SNN for compressing the encoded features at the second layer of hybrid classifier. The (a) is the feature mapping layer; (b) and (c) refer to the networks with feature mapping and learning layers; (d) shows the double layer network with sub-network nodes including two feature mapping and single learning layer.

feature mapping layer; (b) and (c) show the first and second networks for compressing the original data in two stages; and (d) shows the combination of the first and second stages in the multi-layer network including two feature mapping layers.

The DL-SNN is composed of general nodes formed by several hidden nodes to compressing features (see Figure 4.2.2). The number of general nodes and output dimension are independent while the number of hidden nodes in each general neuron must be equal to the dimension of outputs (m). The optimal general parameters are generated in feature mapping layer using the inverse of the activation function. The following five steps are performed to provide the optimal feature set in the DL-SNN framework.

As the first step, we randomly generate the initial general node of the feature mapping layer , by setting $j = 1$, as

$$\mathbf{H}_f^j = \mathbf{g}\left(\hat{\mathbf{a}}_f^j.\,\mathbf{x} + \hat{\mathrm{b}}_f^j\right), \left(\hat{\mathbf{a}}_f^j\right)^T.\hat{\mathbf{a}}_f^j = \mathbf{I}, \left(\hat{\mathrm{b}}_f^j\right)^T.\,\hat{\mathrm{b}}_f^j = 1 \tag{4.2.3}$$

where $\mathbf{H}_f^j$ is the current feature data, and $\hat{\mathbf{a}}_f^j \in \mathbf{R}^{d\times 2DK}$, $\hat{\mathrm{b}}_f^j \in \mathbf{R}$ are the orthogonal random weight and bias of feature mapping layer.

As the second step, we calculate the parameters in the learning layer based on the sigmoid activation function ($\mathbf{g}$) for any continuous desired outputs ($\mathbf{y}$),

$$\begin{aligned}
&\hat{\mathbf{a}}_h = \mathbf{g}^{-1}(u_{2DK}(\mathbf{y}))\cdot\left(\mathbf{H}_f^j\right)^{-1} \quad, \hat{\mathbf{a}}_h^j \in \mathbf{R}^{d\times m}\\
&\hat{b}_h = \sqrt{\text{mse}\left(\hat{\mathbf{a}}_h^j\cdot\mathbf{H}_f^j - \mathbf{g}^{-1}(u_{2DK}(\mathbf{y}))\right)}\ , \hat{b}_{2DK}^j \in \mathbf{R}\\
&\mathbf{g}^{-1}(\cdot) = -\log(\frac{1}{(\cdot)} - 1) \quad \text{if } \mathbf{g}(\cdot) = 1/(1+e^{-(\cdot)})
\end{aligned} \tag{4.2.4}$$

where $\mathbf{H}^{-1} = \mathbf{H}^T(CI + \mathbf{H}\mathbf{H}^T)^{-1}$ while C is a positive value, u_{2DK} is a normalized function $u_{2DK}(\mathbf{y}) : \mathbf{R} \to (0, 1]$, and $\mathbf{g}^{-1}$ and u_{2DK}^{-1} represent reverse functions.

As the third step, we update the output error as $\mathbf{e}_j = \mathbf{y} - u_{2DK}^{-1}\mathbf{g}(\mathbf{H}_f^j, \hat{\mathbf{a}}_h, \hat{b}_h)$, and obtain the error feedback data as

$$\mathbf{P}_j = \mathbf{g}^{-1}(u_{2DK}(\mathbf{e}_j))\cdot(\hat{\mathbf{a}}_h)^{-1} \tag{4.2.5}$$

As the fourth step, we update the feature data as $\mathbf{H}_f^j = \sum_{l=1}^{j} u_l^{-1}\mathbf{g}(\mathbf{x}, \hat{\mathbf{a}}_f^l, \hat{b}_f^l)$ by setting $j = j + 1$ and adding a new general node $\hat{\mathbf{a}}_f^j, \hat{b}_f^j$ in the feature mapping layer by

$$\begin{aligned} \hat{\mathbf{a}}_f^j &= \mathbf{g}^{-1}(u_j(\mathbf{P}_{j-1})) \cdot \mathbf{x}^{-1} \ , \hat{\mathbf{a}}_f^j \in \mathbf{R}^{d\times 2DK} \\ \hat{b}_f^j &= \sqrt{\text{mse}(\hat{\mathbf{a}}_f^j \cdot \mathbf{x} - \mathbf{P}_{j-1})} \ , \hat{b}_f^j \in \mathbf{R} \end{aligned} \tag{4.2.6}$$

Finally, we iterate the steps 2 to 4 for $L - 1$ times. It is worth to mention that a new general node is added to the existing network when repeating steps 2 to 4 once. The parameters $\{\hat{\mathbf{a}}_f^j, \hat{b}_f^j\}_{j=1}^{L}$ are *optimal projecting parameters* and the feature data $\mathbf{H}_f^L = \sum_{j=1}^{L} u_j^{-1}\mathbf{g}(\mathbf{x}, \hat{\mathbf{a}}_f^j, \hat{b}_f^j) = \mathbf{H}_f^*$ are the *optimal feature data*.

The DL-SNN can be used as a multi-layer network. The multi-layer network provides a better general performance than double-layer structure. In the multi-layer strategy, the input data is transformed into multi-layers, and the input raw data is converted into d-dimensional space using multitude feature mapping layers. As depicted in Figure 4.2.2(d), given a training set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{M} \subset \mathbf{R}^{2DK} \times \mathbf{R}^m$, the compressed features are represented as $\mathbf{H}_f^T = \sum_{i=1}^{L} \mathbf{g}(\mathbf{H}_f^T \cdot \hat{\mathbf{a}}_f^i + \hat{b}_f^i)$ where $\mathbf{H}_f^T$ is the output of the second layer in the multi-layer network.

4.3 Experiments and Results

This section describes the employed datasets, effect of the 3D-DWT and hybrid classification on action recognition, and the experimental results of our proposed approach compared with the recent state-of-the-are methods.

4.3.1 Datasets

We evaluate the effect of preprocessing method and hybrid classifier over Weizmann [2], URADL [84], and KTH [87] datasets. All the videos of employed datasets are captured with static cameras and homogeneous backgrounds.

The KTH and Weizmann datasets contain simple actions such as running and walking. However, the URADL dataset contains more complex actions such as writing on board and

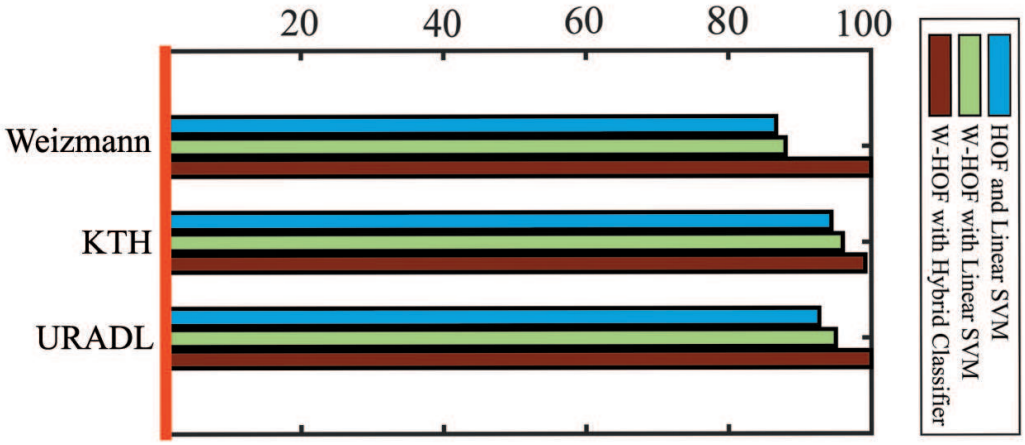


Figure 4.3.1: The recognition performance of HOF and W-HOF based on the traditional linear SVM and proposed hybrid classifier on three employed datasets. The horizontal axis refers to the percentages of action recognition accuracy over three datasets.

drinking water. The same experimental settings are kept for training and testing stages of the hybrid classifier over each dataset.

4.3.2 Effect of motion saliency map on action recognition

As depicted in Figure 4.3.1, the W-HOF provides a good performance on three employed datasets. We evaluate the W-HOF performance using the traditional linear SVM and proposed hybrid classifier. In both cases, the results are boosted compare to the HOF features which are trained by traditional linear SVM. It shows that the described features by W-HOF delivers advanced information to the classification stage. Thus, 3D-DWT can be considered as a powerful option to extract motion saliency maps before HOF description.

We evaluate a set of thresholding values to extract motion saliency maps from the transformed data. Based on the experiments, 200 is considered as the best thresholding value to provide optimal motion saliency maps for HOF description.

4.3.3 Evaluation of hybrid classifier

The encoded features are compressed to d dimension at the second layer of hybrid classifier. We evaluate different compression dimensions to compress the data. As shown in Figure

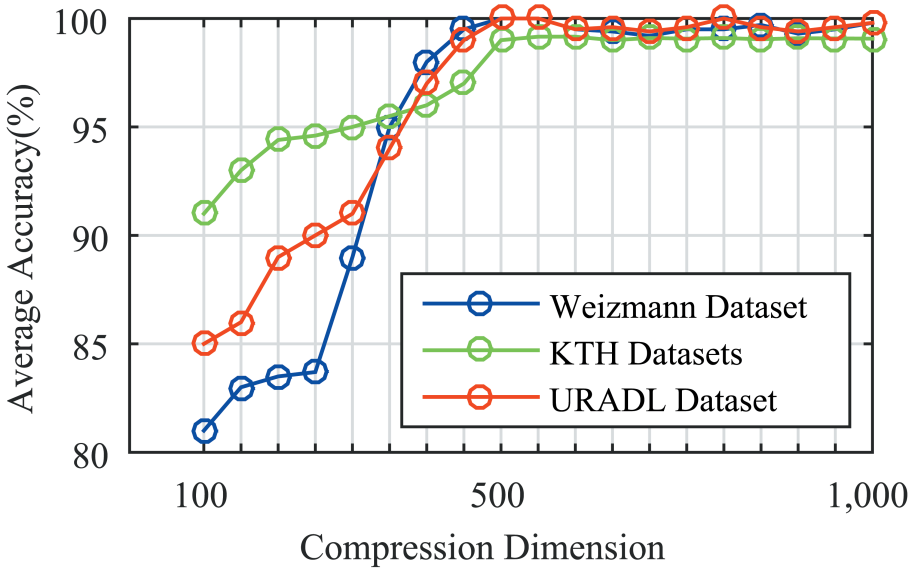


Figure 4.3.2: Evaluation of a set of dimensions for compressing the features at the second layer of hybrid classifier.

4.3.2, compressing the features to 500 is considered as the best option for three datasets. It should be noted that the double-layer net with sub-network nodes (DL-SNN) is not sensitive to the parameters of the networks. Thus, we can select the parameters randomly without affecting the generalization performance in the learning process.

The optimized thresholding constant (E) in Equation 4.2.2, is considered as 2.5 for three employed datasets. As shown in Figure 4.3.1, the proposed hybrid classifier outperforms the traditional SVM while classifying the W-HOF features. For the URADL and Weizmann datasets, all the samples are automatically fed to the SVM with sigmoid kernel in the third layer of hybrid classifier. However, for the KTH dataset, some of the samples which provide equivalent probabilities are automatically classified by sigmoid or polynomial kernels in the third layer. Moreover, the classification of boxing and hand-waving samples is performed confidently with linear SVM at the first layer of hybrid classifier.

4.3.4 Results

In Table 4.3.1, we further compare our results with several state-of-the-art approaches. The proposed framework achieves 100%, 98%, and 100% accuracy for Weizmann, KTH, and URADL datasets, and outperforms the state-of-the-art methods. The obtained results

Table 4.3.1: Comparison of our results based on hybrid classifier to the state-of-the-arts

Dataset	Method	Recognition Rate
Weizmann	Cao et al. [95]	99.6%
	Lei et al. [96]	89.2%
	Samanta et al. [97]	90.0%
	Sushma et al. [98]	95.55
	Proposed Framework	**100.00%**
KTH	Cao et al. [95]	92.0%
	Lei et al. [96]	93.97%
	Samanta et al. [97]	94.7%
	Barrett et al. [99]	94.9%
	Proposed Framework	**98.00%**
URADL	Prest et al. [100]	92%
	Bilibski et al. [101]	94.7%
	Wang et al. [5]	96%
	Eman et al. [102]	96.6%
	Proposed Framework	**100.00%**

demonstrate that the compression of encoded features can enhance the recognition performance in the hybrid classifier. This is due to the automatical usage of polynomial or sigmoid kernels in the third layer of hybrid classifier. The sigmoid and polynimial kernels are mainly very effective while training a data with moderate feature dimenssion.

We conclude that the encoded features may contain outliers and redundant info which make the classification more challenging and time-consuming. And the employed DL-SNN provides optimal compressed features for action recognition in our hybrid classifier.

4.4 Summary

This chapter evaluates the effect of 3D-DWT on motion features and proposes a hybrid classifier for action recognition. The experimental results show that motion saliency maps, which are obtained by 3D-DWT, are capable of maturing the motion feature extraction for action recognition. Furthermore, it is shown that the proposed hybrid classifier is capable of leveraging the linear, sigmoid and polynomial kernels in SVM classifier. The results show that the compression of encoded features can enhance the recognition performance

in hybrid classifier. The experimental results demonstrate that the proposed framework achieves promising results compared with the state-of-the-art approaches.

CHAPTER 5

Hierarchical Feature Representation for Complex Action Recognition

5.1 Overview

Complex video analysis is a challenging problem due to the long and sophisticated temporal structure of unconstrained videos. This chapter introduces pooled-feature representation (PFR) which is derived from a double phase encoding framework (DPE) to address this problem. Considering that a given unconstrained video is composed of a sequence of simple frames, the first phase of DPE generates temporal sub-volumes from the video and represents them individually. The second phase constructs the pool of features by fusing the represented vectors from the first phase. The pool is compressed and then encoded to provide video-parts vector (VPV). This framework allows distilling the video representation and hierarchically extracting new information. Compared with recent video encoding approaches, VPV can preserve the higher-level information through standard encoding of low-level features in two phases. Furthermore, the encoded vectors from both phases of DPE are fused along with a compression stage to develop PFR. The early and late fusion steps are adopted based on the priority of compression stage over the concatenation of represented vectors.

Feature ranking from video-wide temporal evolution brings reliable information for complex action recognition. However, a video may contain similar features in the sequence of frames which deliver unnecessary information to the ranking function. In addition to the PFR method, this chapter proposes a method to enhance the general rank-pooling strategy which captures the optimized latent structure of the video sequence data. The optimiza-

tion is followed by removing the redundant features from the sequence data. The cosine and correlation distance metrics are employed to detect the identical features and extract the most efficient information from the video frames. Then, the ranked features are generated from the optimized and clean sequence data. The proposed improvement is easy to implement, fast to compute and effective in recognizing complex actions.

We adopt the improved rank pooling methodology to represent the temporal blocks at the first phase of DPE framework. To validate the proposed IRP and PFR methods, we conduct extensive experiments on six complex action datasets: UCF50, HMDB51, URADL, Olympic, Hollywood2 and UCF101. Experimental results demonstrate that PFR with early fusion achieves the state-of-the-art performance by capturing the most prominent features with minimum dimension compared to the typical video representation frameworks.

In particular, this chapter makes the following contributions:

1) Enhancing the general rank pooling strategy by removing the redundant features from identical frames of a given video.

2) Employing the cosine and correlation distance metrics for detecting and removing the redundant features using the proposed consequent and protracted checking methods. Furthermore, we define the optimized threshold for removing the identical features.

3) Proposing the DPE framework to provide useful information for complex video analysis based on the assumption that a complex video is composed of a sequence of simple actions. The early fusion of individual represented vectors (IRVs), at the first phase of DPE, outperforms the traditional encoded vectors obtained from features of entire video.

4) Introducing VPV at the second phase of DPE to represent higher-level information for unconstrained video analysis.

5) Proposing the PFR with early and late fusion strategies to leverage the motion and appearance-based information from a given video. It is worth pointing out that PFR with early fusion provides the most significant features with least dimension and highest efficiency for complex action recognition.

6) Adopting the learning strategy with sub-network nodes (LSN) classifier in the action recognition domain and compare its training speed and accuracy with traditional extreme learning machine (ELM) and support vector machine (SVM) over six challenging datasets.

The rest of the chapter is organized as follows: Section 5.2 explains the proposed framework w.r.t. existing works. This is followed by the evaluation and testing of our methods in Section 5.3. Finally, Section 5.4 summarizes the proposed methods and experimental results.

5.2 Proposed Framework

We hypothesize that hierarchically encoding the temporal sub-volumes of local video-based features is a crucial step to producing higher-level features for action recognition. This chapter introduces the pooled-feature representation (PFR), inherited from double phase encoding (DPE), to present the most informative and prominent features of an unconstrained video. The PFRs are employed as the input to the learning strategy with sub-network nodes (LSN) [59] to classify actions. It should be noted that we adopt the proposed improved rank pooling (IRP) strategy to represent the temporal blocks in the DPE framework. This section presents the proposed action recognition framework by describing the IRP, DPE, PFR, and LSN strategies.

5.2.1 Improved Rank Pooling

The relative ordering of the video frames plays a crucial factor in video representation frameworks. The general rank pooling strategy [61] has been proposed to represent a given video by analysis the orders of the available frames in that video. However, the similar encoded features from the identical frames may affect the ranking machines and produce inefficient video representation. To address this problem, we detect the identical features and remove them from the $\mathbf{F}=\{\bar{f}_1,\cdots,\bar{f}_t\}$ where $\mathbf{F}$ is the set of features from a given video and t is the length of that video. Consequently, we obtain a set of $\mathbf{F}=\{\bar{f}_1,\cdots,\bar{f}_\eta\}$ where

all the η features are unique. In this case, the video representation is generated using the unique set of features without any similarity among data. The similarity between a pair of vectors is detected using the distance metrics. The cosine and correlation distance metrics are employed to measure the similarity among features. We evaluate the IRP framework based on the stated similarity metrics in the experimental section of this chapter. The brief description of theses metrics is stated as follows:

- Cosine distance metric [103]: To assign a numeric score to pair of vectors, the model measures the similarity between the query vectors as

$$d^{\cos}_{(f_n, f_{n+1})} = \frac{\sum_{i=1}^{\mathbf{D}} (f_n^{(i)} f_{n+1}^{(i)})}{\sqrt{\sum_{i=1}^{\mathbf{D}} (f_n^{(i)})^2} \sqrt{\sum_{i=1}^{\mathbf{D}} (f_{n+1}^{(i)})^2}} \tag{5.2.1}$$

  The angle between two vectors is used as a measure of divergence between the vectors, and cosine of the angle is used as the numeric similarity. Based on the cosine's property, we obtain 1.0 for identical vectors and 0.0 for orthogonal vectors. Therefore, if the numeric similarity is above the threshold, the later vector is removed from the feature set.

- Correlation distance metric: The correlation similarity score between two vectors is calculated as

$$d^{corr}_{(f_n, f_{n+1})} = \frac{\frac{1}{\mathbf{D}} \sum_{i=1}^{\mathbf{D}} (f_n^{(i)} f_{n+1}^{(i)}) - (\mu_{f_n} \mu_{f_{n+1}})}{\sigma_{f_n} \sigma_{f_{n+1}}} \tag{5.2.2}$$

  where μ_{f_n} and $\mu_{f_{n+1}}$ are the means and σ_{f_n} and $\sigma_{f_{n+1}}$ are the standard deviations of f_n and f_{n+1} respectively. The numerator of the equation is called the covariance of f_n and f_{n+1}, and is the product of f_n and f_{n+1} subtracted from the product of their means.

The similar frames are removed based on the protracted and consequent methods where the subsequent feature vectors are analysed and compared. As the protracted method, in

case of having similarity among f_n and f_{n+1}, the later vector is removed and then comparison is performed between the f_n and f_{n+2} vectors. If the similarity score is below the threshold, both the vectors are kept and the next comparison is performed between f_{n+2} and f_{n+3}. As the consequent method, the couple checking is performed between any couple vectors. If the similarity score is over the threshold, the later vector is removed. Figures 5.2.1(a) and 5.2.1(c), show the feature vectors from the original video frames. As depicted in Figure 5.2.1(b) and 5.2.1(d), the similar vectors are removed from the original set and the sets of cleaned vectors are obtained based on the protracted and consequent methods respectively. The protracted and consequent removing schemes are performed from the starting frame to the last frame of a particular video.

Additionally, we propose the double thresholding scheme (DTS) as the complementary step for removing the identical features from temporal sub-volumes. The underlying motivation behind the DTS is to behave with short and long shot videos efficiently. More identical feature vectors may exist in the long shot sub-volumes. Therefore, the higher threshold ε_h is used to remove more vectors from a long shot sub-volume. However, we may achieve just a few feature vectors from a short video while using the same threshold which is used for long shot videos. In this case, we consider the lower ε_l to remove the similar feature vectors less strictly. If the number of frames of the processed sub-volume is above 80 frames, the higher value is selected to remove the identical features. Otherwise, the lower threshold is adopted for eliminating similar features. Based on the adopted cosine and correlation distance metrics, the DTS evaluation for finding the most reliable thresholding values is explained in Section 5.3.1.

The relative ordering of the feature vectors from a given video frames is preserved regardless of the significant variability in intra-class action performance speeds. For instance, the walking action can be performed faster or slower. However, regardless of the speed of walking, the ordering of the frames remains the same for all walking actions. We employ the rank pooling method as described in [61] to represent the cleaned ordering of the frames from a particular video. The rank pooling is performed based on the assumption that a given video is composed of t frames while the frame feature vector at a discrete time step n is denoted by $\bar{f}_n$. The rank pooling represents the temporal evolution

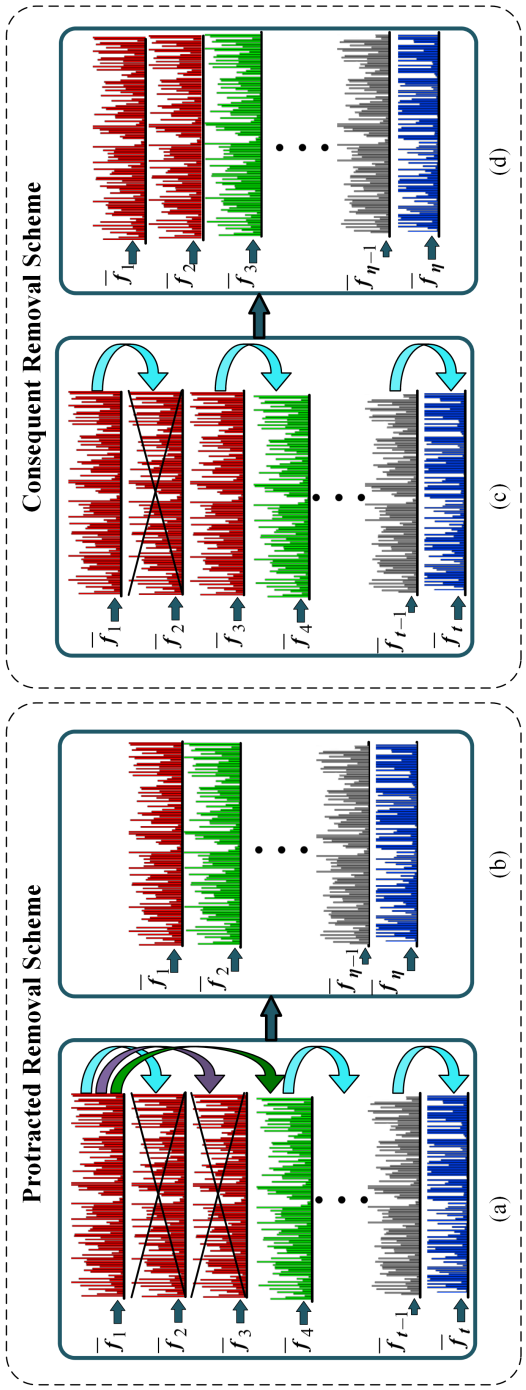


Figure 5.2.1: Protracted and consequent removal schemes over an example. (a) and (b) show the original and clean feature sets based on the protracted scheme. (c) and (d) show the original and clean feature sets based on the consequent scheme.

of $\mathbf{F}$=$\{\bar{f}_1,\cdots,\bar{f}_n,\cdots,\bar{f}_t\}$ from a given sub-volume. The $\bar{V}$ indicates the represented information which reflects the change of frame features from time n to $n+1$ for unique frames in a given sub-volume. As in [61], a pairwise linear ranking machine learns parameters p from the data $\mathbf{F}$ where p lies in the same space as the cleaned data in $\mathbf{F}$. In other words, p represents the information from the content of $\mathbf{F}$ by capturing the data about ordering in $\mathbf{F}$. In other words, rank pooling is a function-based temporal pooling approach to capture the latent structure of the video sequence data, e.g., how frame-level features evolve in a video [61]. The optimization objective for the appearance evolution is expressed based on RankSVM which is considered as a pairwise learning to rank framework [104]. The dynamics of $\mathbf{F}$=$\{\bar{f}_1,\cdots,\bar{f}_\eta\}$, denoted by Δ, are obtained using a linear function φ_p=$\varphi(\mathbf{F};\, p)$ parametrized by p, where φ approximates Δ by

$$\arg\min_{p} \|\Delta - \varphi_p\| . \tag{5.2.3}$$

where dynamics Δ are driving force for ordering the frame feature vectors in the correct sort [61]. The improved rank pooling framework is evaluated over benchmark datasets and the experimental results are presented in Section 5.3.1. In the rest of the chapter, we employ the IRP framework to develop the double phase encoding strategy for complex action recognition.

5.2.2 Double Phase Encoding

We introduce DPE as a hierarchical approach to provide rich semantic clues for complex action recognition through low-level features. As depicted in Figure 5.2.2, the proposed DPE includes two phases where τ denotes the number of temporal sub-volumes at the first phase; the $\bar{V}$ and $\bar{C}$ denote the represented vectors from temporal sub-volumes and compressed feature sets respectively; the d denotes the dimension of features after compression at the second phase; the D and K denote the original feature dimension and number of components for GMM training.

The first phase consists of the following three steps: feature extraction, temporal sub-volume generation, and sub-volume representation. At the first step, we extract features

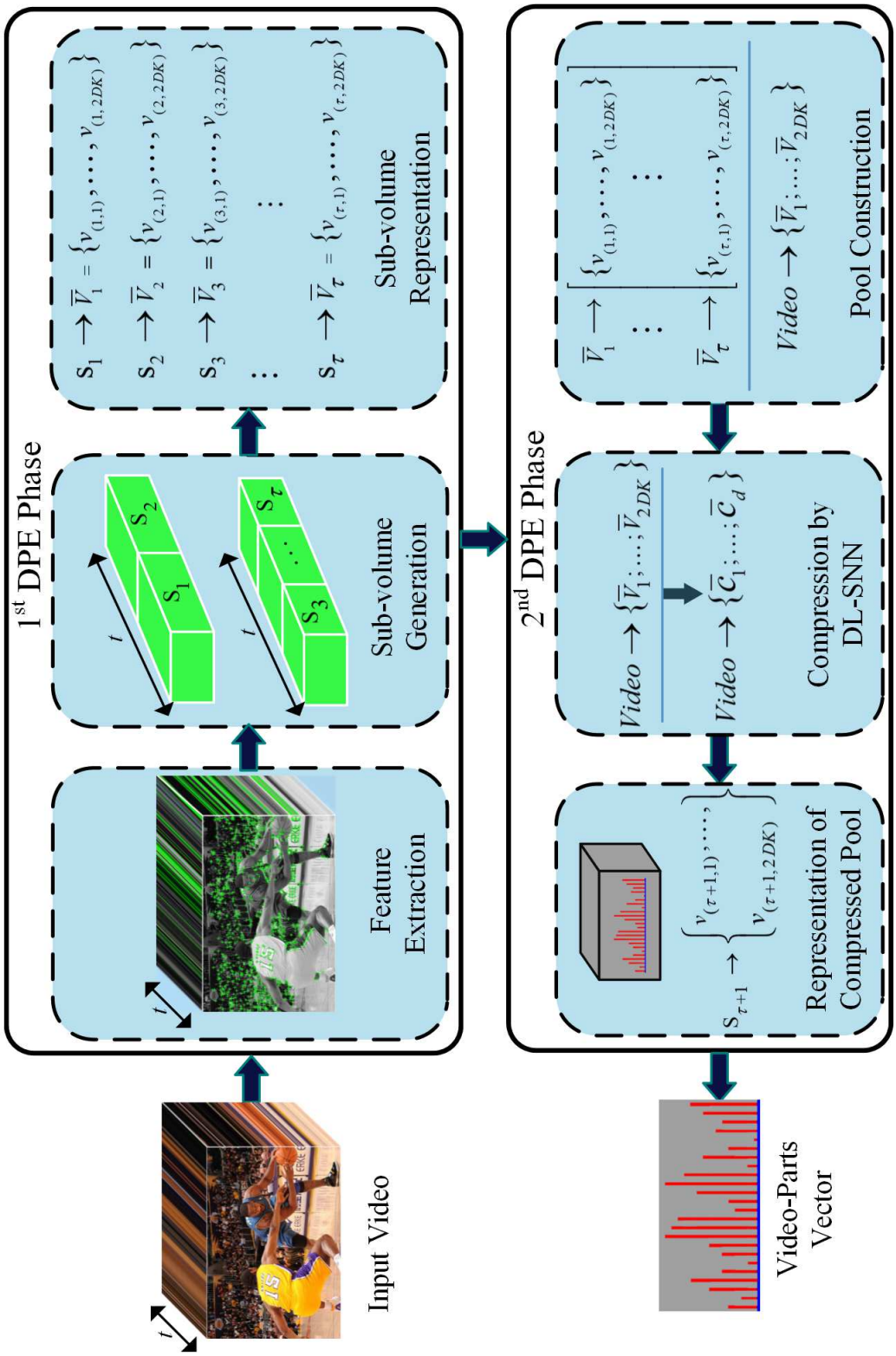


Figure 5.2.2: The proposed double phase encoding (DPE) framework for unconstrained video analysis.

with $\{N, t\}$ dimensions from a given video. The N and t respectively present the number of feature samples and the temporal length of a given video. Even though our proposed methodology is independent of the choice of features, we use the improved dense trajectory (IDT) feature extraction method along with HOG, HOF, and MBH descriptors [31] as the initial step in our framework (see Figure 5.2.3). The IDT features [31] provide state-of-the-art results on benchmark action datasets. The length of the trajectories, number of scales, sampling stride, neighborhood size, and number of spatial and temporal cells are the adjustable parameters for IDT feature extraction [31]. We follow the same parameters from [31] since the feature extraction parameters are precisely optimized in [31].

At the second step, we generate the temporal sub-volumes based on $\left\{\frac{t}{2}, \frac{t}{3}, \frac{t}{4}, \ldots, \frac{t}{w}\right\}$ scales where t is the temporal length of a given video and w is the maximum number of equal sized sub-volumes in the last division scale. Therefore, the video features are split into τ sub-volumes as $\mathbf{S} = \{\mathrm{s}_1, \ldots, \mathrm{s}_\tau\}$ where $\tau = \left(\frac{w}{2}(1+w)\right) - 1$ (see 1^{st} DPE phase at Figure 5.2.2). It is worth pointing out that we generate equal size temporal sub-volumes based on each scale, e.g., $\frac{t}{2}$ produces 2 equal sized temporal sub-volumes and so on for the rest scales.

At the third step, we represent the individual temporal sub-volumes $\mathbf{S} = \{\mathrm{s}_1, \ldots, \mathrm{s}_\tau\}$ of local features as $\mathbf{V} = \{\bar{V}_1, \ldots, \bar{V}_\tau\}$. We employ our IRP method to represent the individual sub-volumes. The variety of features from τ temporal sub-volumes aim to represent a given video considering its temporal details. We encode the features from individual frames in each temporal sub-volume using the Fisher vector (FV) algorithm. The adopted FV algorithm encodes the difference among video features and the vocabulary by applying derivative operations on the likelihood concerning the distribution parameters, (mean (μ), weights (ϖ), and covariance (σ)), of the vocabulary. The Gaussian mixture model (GMM) is adapted to shape the vocabulary parameters. Even though HOF, HOG, and MBH (MBHx, and MBHy) are all histogram based descriptors, they carry different information from different perspectives. Consequently, the individual GMMs are learnt for each descriptor and four different represented vectors are produced for each sub-volume of $\mathbf{S} = \{\mathrm{s}_1, \ldots, \mathrm{s}_\tau\}$. Figure 5.2.3 shows the employment of four descriptors through our representation methodology for complex action recognition. As depicted in Figure 5.2.3, the

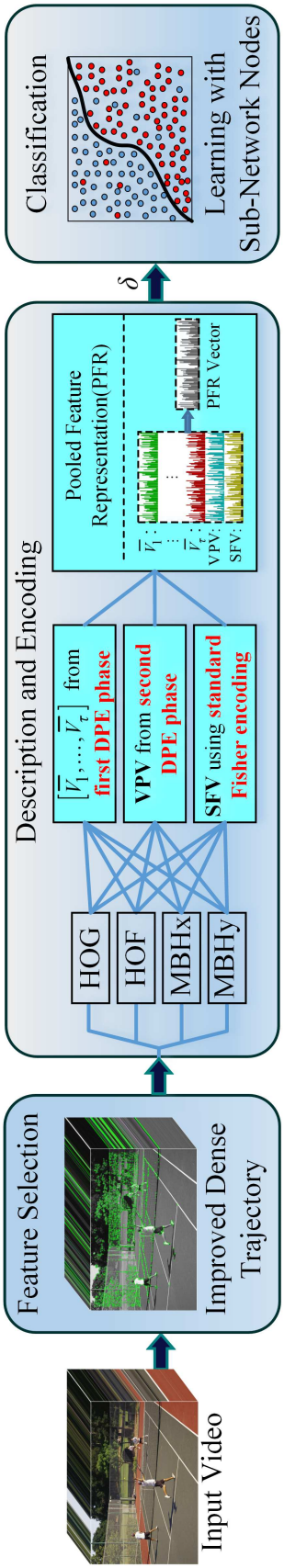


Figure 5.2.3: Video action recognition framework based on the first and second DPE phases and standard Fisher encoding.

improved rank pooling (IRP) representations from four descriptors (HOG, HOF, MBHx, and MBHy) are combined where δ contains the pooled feature representations (PFR) inherited from different descriptors. Section 5.2.3 thoroughly presents the PFR strategy.

In this chapter, we further call the set of τ represented vectors, $\{\bar{V}_1,\cdots,\bar{V}_\tau\}$ as individual represented vectors (IRVs) obtained at the third step of the first phase of DPE framework.

The Second Phase

The ultimate goal of the second phase of DPE is to extract the higher-level video information by encoding the individual feature vectors which are generated at the first phase. The second phase consists of the following three steps: pool construction, pool compression, and pool representation (see 2^{nd} DPE phase in Figure 5.2.2) .

As the first step, we construct the pool of features by fusing the individual feature vectors $\{\bar{V}_1,\cdots,\bar{V}_\tau\}$. As depicted in Figure 5.2.2, the pool includes $2DK$ vectors while each vector contains τ elements. The D, K and τ are the dimension of features, number of components in GMM training, and number of temporal sub-volumes respectively. The pool of features contain the low-level feature vectors from τ temporal sub-volumes of local features. We hypothesize that an encoding strategy aims to extract higher-level features from the pool. However, the size of the pool is too high-dimensional to be represented by the Fisher vector encoding at the second phase. Aside from that, Fisher vectors may contain minimal values, close to zero, which cannot provide extended information for higher-level feature extraction. We hypothesize that compression of the pool of features aims to extract the most reliable and prominent features for the encoding step at the second phase. Thus, as the second step in the second phase of DPE, the pool of features is mapped to a lower dimensionality space, impressively. The pool is compressed from $2DK$ to d dimension using the double-layer net with sub-network nodes (DL-SNN) framework.

As the third step, the compressed pool is decorrelated using PCA and then employed as the input to the Fisher vector encoding in the second phase of DPE. The GMM with the same codebook size, as in the first phase, is learned and the higher level information is obtained as shown in Figure 5.2.2. The VPV contains the higher level information which is derived from the encoding of temporal sub-volume features. Next section describes the

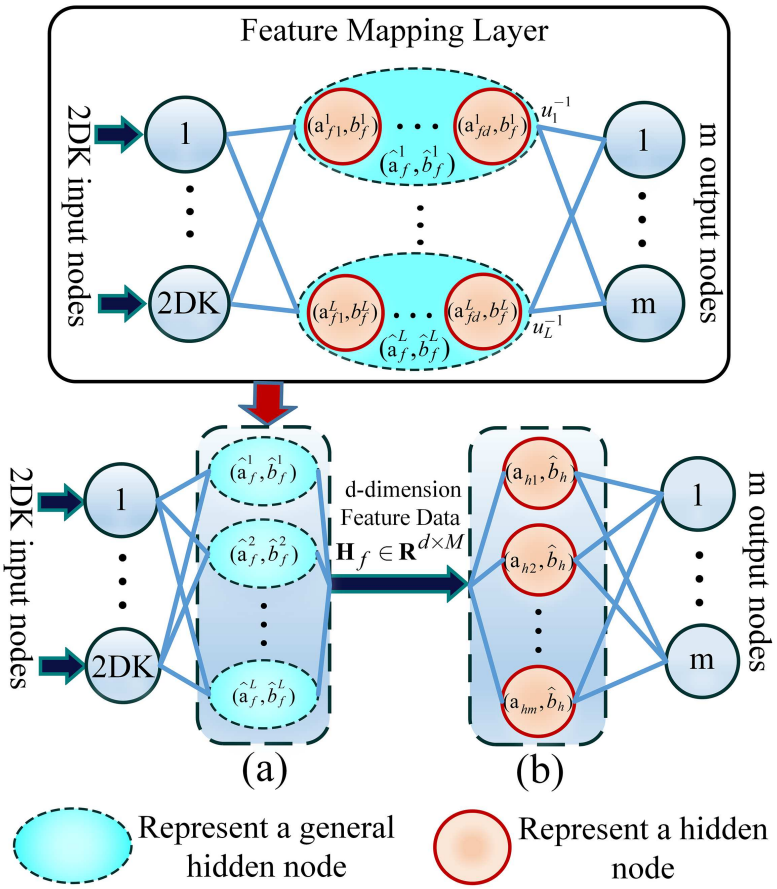


Figure 5.2.4: Different layers of the Double-Layer Net with Sub-Network Nodes. (a) demonstrates the feature mapping layer. (b) shows the learning layer

DL-SNN framework for compression of the pool of features as the second step of second DPE phase.

Dimensionality Reduction

The generated pool at the second phase of DPE includes a high dimension of features. The high dimensionality significantly increases the required time and memory for data processing [105]. Thus, we map the pool to a lower dimensional space to extract the most useful and prominent information while reducing the dimension of the pool. We adopt the double layer net with sub-network nodes (DL-SNN) [93] to map the pool to a lower dimensional space. The learning speed of the employed DL-SNN is much faster than deep networks such as stacked autoencoders (SAE) and deep belief networks (DBNs) [93]. Furthermore,

Table 5.2.1: Notations to be used in the proposed video analysis framework

Notation	Meaning
M	Number of training samples.
D	Local features dimension after reducing their size by PCA.
K	Number of components for Gaussian mixture training.
$2DK$	Dimension of the encoded features using Fisher vector method.
w	Maximum number of temporal sub-volumes in DPE framework.
m	Output data dimension (number of classes).
L	The numbers of general hidden nodes.
c	Regularization parameter.
$(\boldsymbol{\alpha}, \flat)$	A hidden node.
$(\mathbf{a}, b)$	A general hidden node (or subnetwork node).
$\hat{\mathbf{a}}_f^j$	Input weight of the jth general hidden node in feature mapping layer. $\hat{\mathbf{a}}_f^j \in \mathbf{R}^{d\times n}$.
$\hat{b_f^j}$	Bias of the jth general hidden node in feature mapping layer $\hat{b_f^j} \in \mathbf{R}$.
$(\boldsymbol{\alpha}_{fi}^j, \flat_f^j)$	The ith hidden node in the jth general hidden node.
$(\hat{\mathbf{a}_h}, \hat{b}_h)$	Hidden nodes in learning layer and $\hat{\mathbf{a}_h} \in \mathbf{R}^{m\times d}$.
u_j	Normalized function in the jth general node, $u_j(\cdot) : \mathbf{R} \to (0, 1]$, u_j^{-1} represent its reverse function.
$\mathbf{H}_f^j$	Feature data generated by j general nodes in a feature mapping layer.
$\mathbf{e}_L$	The residual error of current two-layer network (L general nodes in the first layer and $(\mathbf{a}_h, b_h)$ in the second layer).

the DL-SNN can provide a better generalization performance than other dimension reduction approaches such as isomap and linear discriminant analysis [93]. Additionally, we show that DL-SNN is not sensitive to the parameters of the networks. Therefore, we can select the parameters randomly without affecting the generalization performance in the learning process.

The summary of the notations, used in DL-SNN, is described in Table. 5.2.1. The DL-SNN platform consists of feature mapping and learning layers. The feature mapping layer includes general nodes formed by several hidden nodes which naturally forms the natural learning procedure (see Figure 5.2.4). The numbers of general nodes and output dimension are independent while the number of hidden nodes in each general neuron must be equal to the size of outputs (m). To achieve the optimal general parameters in the feature mapping layer, the inverse of the activation function is adopted as the based on the learning steps. The following five steps are performed to ensure that the represented features, derived from DL-SNN, are the optimal feature data at the second phase of DPE.

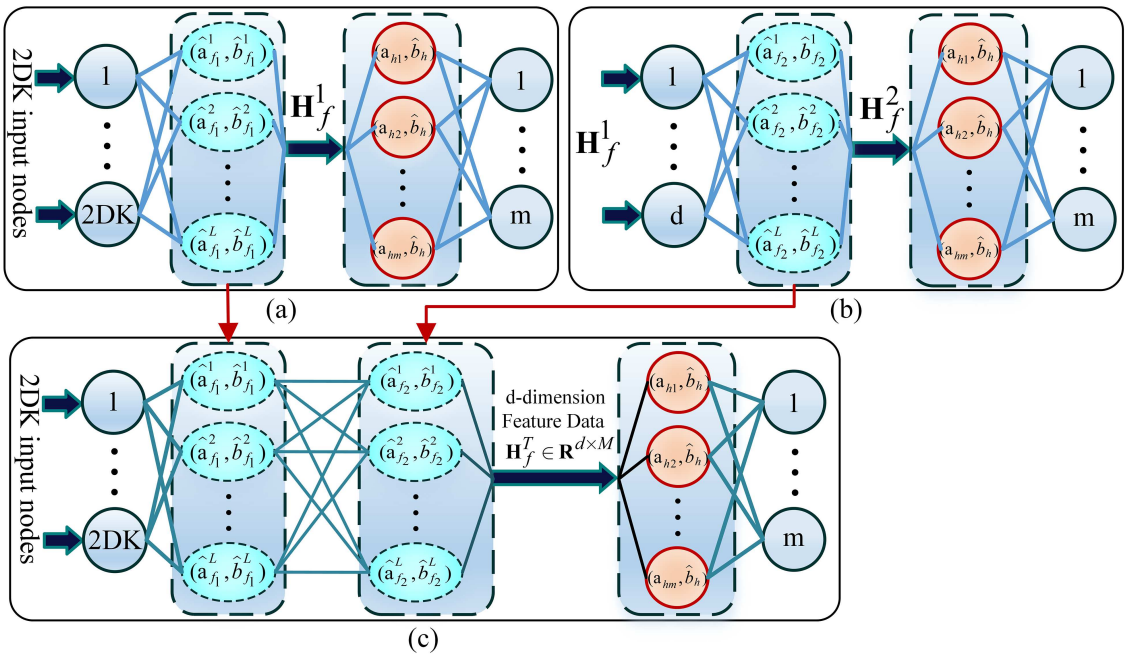


Figure 5.2.5: Double-layer net with sub-network nodes for mapping the feature data.

Step 1: We set $j = 1$ for M distinct training samples, $\{(\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{M}, \mathbf{x}_k \in \mathbf{R}^{2DK}, \mathbf{y} \in \mathbf{R}^m$. Then, the initial general node of the feature mapping layer is generated randomly as

$$\mathbf{H}_f^j = \mathbf{g}\left(\hat{\mathbf{a}}_f^j . \mathbf{x} + \hat{\mathrm{b}}_f^j\right), \left(\hat{\mathbf{a}}_f^j\right)^T . \hat{\mathbf{a}}_f^j = \mathbf{I}, \left(\hat{\mathrm{b}}_f^j\right)^T . \hat{\mathrm{b}}_f^j = 1 \tag{5.2.4}$$

where $\mathbf{H}_f^j$ is the current feature data, $\mathbf{g}$ is the activation function, and $\hat{\mathbf{a}}_f^j \in \mathbf{R}^{d\times 2DK}$, $\hat{\mathrm{b}}_f^j \in \mathbf{R}$ are the orthogonal random weight and bias of feature mapping layer.

Step 2: Calculate the parameters in the learning layer based on a sigmoid or sine activation function $\mathbf{g}$ for the continuous desired outputs $\mathbf{y}$,

$$\begin{aligned}
&\hat{\mathbf{a}}_h = \mathbf{g}^{-1}(u_{2DK}(\mathbf{y})) \cdot \left(\mathbf{H}_f^j\right)^{-1} \quad , \hat{\mathbf{a}}_h^j \in \mathbf{R}^{d\times m} \\
&\hat{b}_h = \sqrt{\mathrm{mse}\left(\hat{\mathbf{a}}_h^j \cdot \mathbf{H}_f^j - \mathbf{g}^{-1}(u_{2DK}(\mathbf{y}))\right)} \quad , \hat{b}_{2DK}^j \in \mathbf{R} \\
&\mathbf{g}^{-1}(\cdot) \begin{cases} \arcsin(\cdot) & \text{if } \mathbf{g}(\cdot) = \sin(\cdot) \\ -\log(\dfrac{1}{(\cdot)} - 1) & \text{if } \mathbf{g}(\cdot) = 1/(1+e^{-(\cdot)}) \end{cases}
\end{aligned} \tag{5.2.5}$$

where $\mathbf{H}^{-1} = \mathbf{H}^T(CI + \mathbf{H}\mathbf{H}^T)^{-1}$ while C is a positive value, u_{2DK} is a normalized function

$u_{2DK}(\mathbf{y}) : \mathbf{R} \rightarrow (0, 1]$, and $\mathbf{g}^{-1}$ represents reverse activation function.

Step 3: Update the output error $\mathbf{e}_j$ as

$$\mathbf{e}_j = \mathbf{y} - u_{2DK}^{-1}\mathbf{g}(\mathbf{H}_f^j, \hat{\mathbf{a}}_h, \hat{b}_h), \tag{5.2.6}$$

and obtain the error feedback data as

$$\mathbf{P}_j = \mathbf{g}^{-1}(u_{2DK}(\mathbf{e}_j)) \cdot (\hat{\mathbf{a}}_h)^{-1} \tag{5.2.7}$$

Step 4: Update the feature data as

$$\mathbf{H}_f^j = \sum_{l=1}^{j} u_l^{-1}\mathbf{g}(\mathbf{x}, \hat{\mathbf{a}}_f^l, \hat{b}_f^l) \tag{5.2.8}$$

by setting $j = j + 1$ and adding a new general node $\hat{\mathbf{a}}_f^j, \hat{b}_f^j$ in the feature mapping layer by

$$\begin{aligned} \hat{\mathbf{a}}_f^j &= \mathbf{g}^{-1}(u_j(\mathbf{P}_{j-1})) \cdot \mathbf{x}^{-1} \ \ , \hat{\mathbf{a}}_f^j \in \mathbf{R}^{d \times 2DK} \\ \hat{b}_f^j &= \sqrt{\text{mse}(\hat{\mathbf{a}}_f^j \cdot \mathbf{x} - \mathbf{P}_{j-1})} \ \ , \hat{b}_f^j \in \mathbf{R} \end{aligned} \tag{5.2.9}$$

Step 5: Repeat steps 2 to 4 for $L - 1$ times. It is worth to mention that a new general node is added to the existing network in the feature mapping layer when repeating steps 2 to 4 once. The parameters $\{\hat{\mathbf{a}}_f^j, \hat{b}_f^j\}_{j=1}^L$ are *optimal projecting parameters* and the feature data $\mathbf{H}_f^L = \sum_{j=1}^L u_j^{-1}\mathbf{g}(\mathbf{x}, \hat{\mathbf{a}}_f^j, \hat{b}_f^j) = \mathbf{H}_f^*$ are the *optimal feature data*.

The employed method can be used as a double-layer network including two feature mapping layers (see Figure 5.2.5). The structure with double-layer network provides a better general performance than single-layer structure. In the double-layer strategy, the input data is transformed into two feature mapping layers, and the input raw data is converted into d-dimensional space. Figure 5.2.5 depicts the double layer net with subnetwork nodes as follows: (a) Mapping from initial feature set to H_f^1; (b) Mapping from H_f^1 to H_f^2; (c) Mapping from initial feature set to H_f^T using the input weights $[(\hat{\mathrm{a}}_{f_1}^1, \hat{b}_{f_1}^1), \ldots, (\hat{\mathrm{a}}_{f_1}^L, \hat{b}_{f_1}^L)]$ and $[(\hat{\mathrm{a}}_{f_2}^1, \hat{b}_{f_2}^1), \ldots, (\hat{\mathrm{a}}_{f_2}^L, \hat{b}_{f_2}^L)]$.

Mathematically, given a training set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^M$, the output of first and second feature

mapping layers are represented as $\mathbf{H}_f^1 = \sum_{i=1}^{L} \mathbf{g}(\mathbf{H}_f^1 \cdot \hat{\mathbf{a}}_f^i + \hat{b}_f^i)$, and $\mathbf{H}_f^2 = \sum_{i=1}^{L} \mathbf{g}(\mathbf{H}_f^2 \cdot \hat{\mathbf{a}}_f^i + \hat{b}_f^i)$ where $\mathbf{g}$ denotes the activation function of the hidden layers. As shown in Figure 5.2.5, the input to the double layer network contains $2DK$ features which come from the output of Fisher vectors. We target to compress the input data to an optimized dimension using DL-SNN. The evaluation of the streamlined compression rates is described with details in Section 5.3.

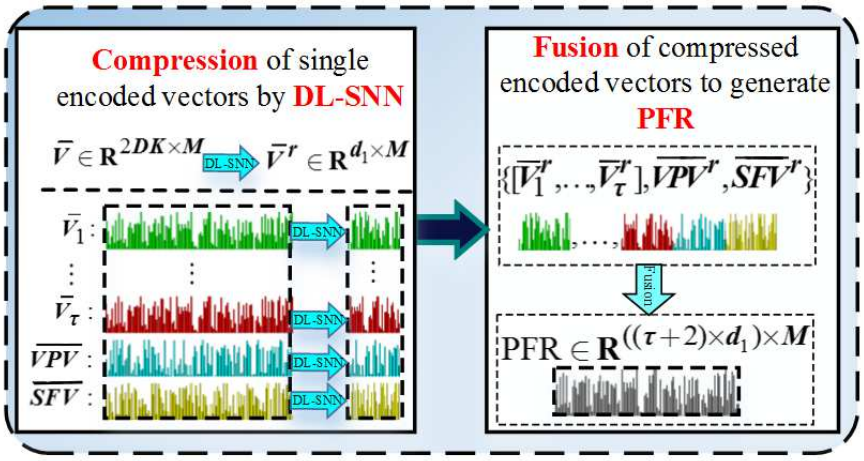
5.2.3 Pooled-Feature Representation

Fusion of multiple feature sets aims to boost the action recognition performances in computer vision applications [20]. The main advantage of fusion is that the classifier can 'see' all the distinct features at once and only one learning phase is required. However, fusion provides a high dimension of feature sets, usually accompanied by limited amounts of training data, which makes the classification more challenging and time-consuming. To address this problem, we hypothesis that the fusion of multiple feature sets can be joint with the compression stage to make a reliable and robust feature vector.

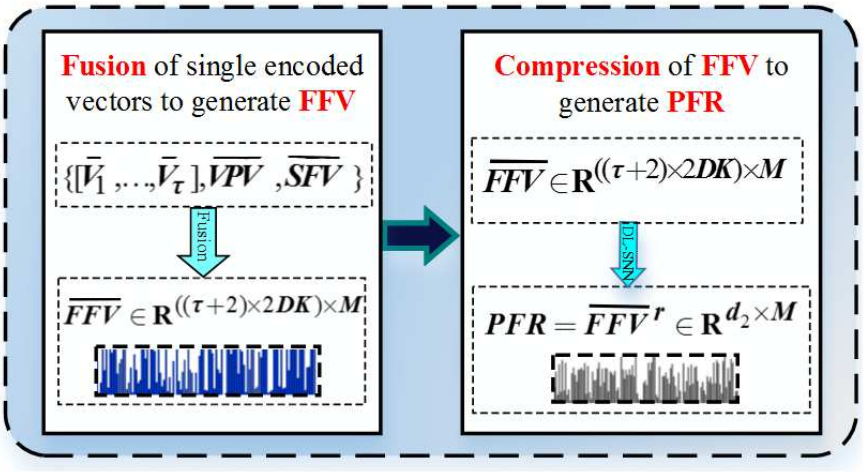
The DPE provides $\tau + 1$ feature vectors in its former and latter phases. Each of the singular vectors represents specific information related to a sub-volume or higher level video information. Therefore, the fusion of singular vectors from DPE and Fisher vector of the entire video can boost the recognition performance since the individual feature sets deliver diverse sets of information. However, early fusion provides a high dimensional feature set with dimension of $(\tau + 2) \times 2\mathrm{DK}$ which is challenging and time-consuming in the training phase. Aside from that, the high dimensionality of data requires huge space of memory and processing power in the classification stage. We adopt the DL-SNN to deal with the massive dimension of fused features based on the two following options.

First Option

As shown in Figure 5.2.6(a), the DL-SNN compresses each single represented vector individually to d_1 dimension. Then, the compressed feature vectors are concatenated to generate the PFR. The PFR is inherited from the compressed version of IRVs which are obtained



(a)



(b)

Figure 5.2.6: The first and second PFR options to obtain the PFR-LF and PFR-EF.

at the first phase of DPE ($\left[\overline{V}^r_1, \ldots, \overline{V}^r_\tau\right]$), the compressed version of video parts vector which is obtained at the second phase of DPE ($\overline{VPV}^r$), and the compressed version of standard FV ($\overline{SFV}^r$). Consequently, the PFR's dimension is become $(\tau + 2) \times d_1$ where $d_1 \ll 2DK$. This vector is then used as the input to the classifier for action recognition. We call the first option as PFR with Late Fusion (PFR-LF) since the fusion is performed as the latter step and compression is implemented as the former step.

Second Option

As shown in Figure 5.2.6(b), the single encoded vectors from the first and second phases of DPE and $\overline{SFV}$ are fused as $\overline{FFV} = \left\{[\overline{V}_1, \ldots, \overline{V}_\tau], \overline{VPV}, \overline{SFV}\right\}$. Then, the DL-SNN is used to compressing the fused vector into d_2 dimension . In other words, the dimension of fused vector is reduced from $(\tau + 2) \times 2\mathrm{DK}$ to d_2. In the Figure 5.2.6(b), the $\overline{FFV}$ and $\overline{FFV}^r$ show the fused vector and compressed version of fused vector respectively. We consider the compressed version of fused vector as PFR with Early Fusion (PFR-EF) since the fusion is performed as the former step and compression is implemented as latter step. The PFR-EF contains the most informative features, and is adopted as the input to classifier for action recognition.

The video data is usually characterized in multiple views, such as static appearance, motion pattern, and motion boundary. The essence of multi-view data requires fusing different descriptors for action recognition. Therefore, we fuse the PFR of MBHx, MBHy, HOG, and HOF descriptors to employ all the appearance and motion informative data in our methodology (see δ in Figure 5.2.3).

5.2.4 Classification Strategy

Most of the recent action recognition frameworks classify the video features using a linear support vector machine (SVM) or extreme learning machine (ELM) [106, 107, 45, 46, 20]. However, in this chapter, we employ the learning strategy with sub-network nodes (LSN) [59] to classify actions. The principle motivation behind the usage of LSN is to further improve the learning speed and accuracy compared to the common SVM and ELM classifiers.

Furthermore, different from common classifiers which are sensitive to the combination of parameters, experimental results show that the generalization performance of the LSN is not susceptible to the parameters of the networks. Thus, the random selection of parameters is not affecting the generalization performance in the learning process.

The LSN classifier is presented as follows: given M feature vectors from arbitrary distinct video samples $\{(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^{M};\ \mathbf{x}_i \in \mathbf{R}^d\ ,\ \mathbf{y}_i \in \mathbf{R}^m\}$, we have $\lim_{d \to +\infty} \|\mathbf{y} - u^{-1}(\mathbf{g}(\hat{\mathbf{a}}_1.\mathbf{x} + \hat{b}_1)).\boldsymbol{\beta}_1 + \ldots + u^{-1}(\mathbf{g}(\hat{\mathbf{a}}_d.\mathbf{x} + \hat{b}_d)).\boldsymbol{\beta}_d\| = 0$ which holds with probability one if

$$\begin{aligned} \hat{\mathbf{a}}_d &= \mathbf{g}^{-1}(u(\mathbf{e}_d - 1)).\ \mathbf{x}^T(CI + \mathbf{x}\mathbf{x}^T)^{-1},\ \ \hat{\mathbf{a}}_d \in \mathbf{R}^{d \times m} \\ \hat{b}_d &= \text{sum}(\hat{\mathbf{a}}_d.\mathbf{x} - \mathbf{g}^{-1}(u(\mathbf{e}_{d-1})))/M,\ \hat{b}_d \in\ \mathbf{R} \\ \beta_d &= \frac{\left\{\mathbf{e}_{d-1}, u^{-1}\left(\mathbf{g}\left(\hat{\mathbf{a}}_d.\mathbf{x} + \hat{b}_d\right)\right)\right\}}{\left\|u^{-1}\left(\mathbf{g}\left(\hat{\mathbf{a}}_d.\mathbf{x} + \hat{b}_d\right)\right)\right\|^2} \end{aligned} \tag{5.2.10}$$

where $\mathbf{x}^T(CI + \mathbf{x}\mathbf{x}^T)^{-1} = \mathbf{x}^{-1}$ is the MoorePenrose generalization inverse of the training samples [108]; $\mathbf{g}^{-1}$ is the inverse of an activation function; and u^{-1} is an inverse function of normalized function u. The $\hat{\mathbf{a}}_d$ and $\hat{b}_d$ denote d_{th} sub-network hidden node, where $\hat{\mathbf{a}}_d \in \mathbf{R}^{d \times m}$ and $\hat{b}_d \in \mathbf{R}$. The $\mathbf{e}_d$ denotes the residual error of current network output Θ_d with d hidden nodes, i.e., $\mathbf{e}_d = \mathbf{y} - \Theta_d$. The $\mathbf{H}$ is the hidden layer output matrix of the single layer feed-forward network (SLFN) where i_{th} column of $\mathbf{H}$ is the i_{th} hidden node output with respect to inputs. The $\mathbf{I}$ and sum($\mathbf{e}$) indicate the unit matrix and the sum of the whole elements of the matrix $\mathbf{e}$. It is proven in [59] that the sequence $\|\mathbf{e}_d\|$ is reduced and converges to zero in the LSN classifier. According to [59], the input and output weights have the minimum norm between all the least-squares solutions. From the other side, it is proven in [59] that the LSN with m hidden nodes is capable of providing a similar or much better generalization performance compared to other learning strategies with thousands of hidden node. Consequently, if equation 5.2.10 calculates $\hat{\mathbf{a}}$ and $\hat{b}$ only once for M different

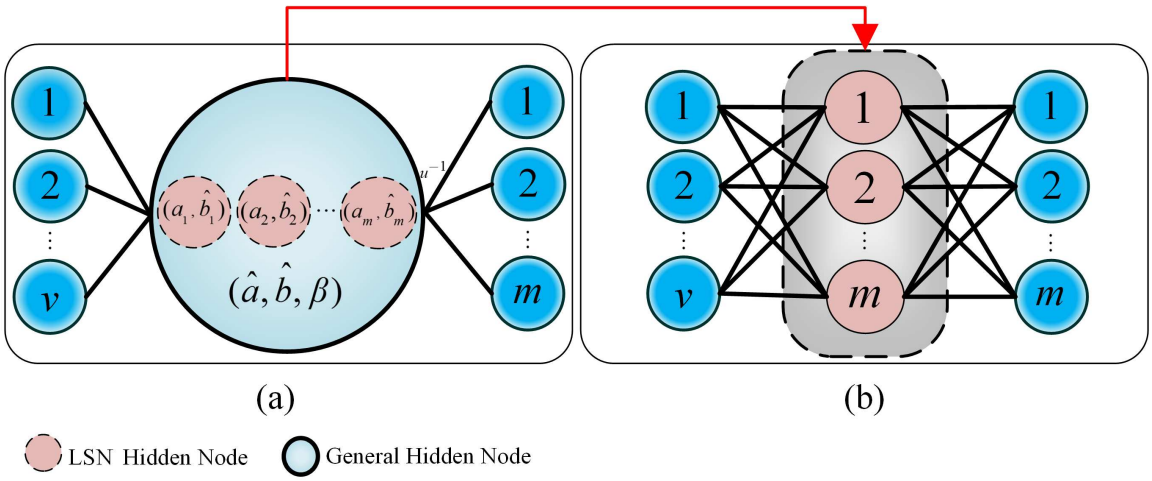


Figure 5.2.7: (a) LSN with one sub-network hidden node. (b) The sub-network hidden node can be considered as a standard SLFN with m hidden nodes.

samples $\left\{(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^{M}\right\}$, $\mathbf{x}_i \in \mathbf{R}^d$, $\mathbf{y}_i \in \mathbf{R}^m$, the network output is generated as

$$\begin{aligned} f_{(\mathbf{x})} &= \beta_1 u^{-1}(\mathbf{g}(\mathbf{a}_1.\mathbf{x}_j + \hat{b}_1)) \\ &= \beta_1 u^{-1}(\mathbf{g}([\mathbf{a}_{11}, \ldots, \mathbf{a}_{1m}].\mathbf{x}_j + \hat{b}_1)) \end{aligned} \tag{5.2.11}$$

where $j = 1, ..., M$ and $\mathbf{g}$ is an activation function. According to Bartletts theory [109], when a learning algorithm finds a network with small weights with a low squared error in the training process, the generalization performance depends on the size of the weights rather than the number of weights . In other words, a robust learning framework must be capable of reaching the smallest training error as well as the smallest norm of weights. Based on this motivation, the sub-network hidden nodes and output weights of the LSN framework provide the smallest norm of weights and lowest training error. The sub-network hidden nodes of the LSN classifier are calculated in the learning layer and not generated randomly. Figure 5.2.7(a) shows the LSN learning framework with its sub-network node as $(\mathbf{a}_1, \hat{b}_1)$, $(\mathbf{a}_2, \hat{b}_2)$, ..., $(\mathbf{a}_m, \hat{b}_m)$. Figure 5.2.7(b) shows that the strategy of the employed network with only one sub-network layer is similar to a standard SLFN. It should be noted that at the first layer of LSN, ν indicates the number of features which are provided by PFR.

5.3 Experiments and Results

This section presents a detailed experimental evaluation for complex action recognition in benchmark datasets. Section 5.3.1 describes the evaluation of IRP and Section 5.3.2 explains the experimental setup and results of PFR framework.

5.3.1 Evaluation of Improved Rank Pooling

Defining the optimized thresholds

The HMDB51 and Hollywood2 datasets contain simple and complex actions with short and long shot videos. However, the URADL dataset is composed of complex actions with very long shot videos. Therefore, the extracted features from URADL may contain more redundant and similar data compare to the HMDB51 and Hollywood2 datasets. Based on the different characteristics of available videos in each dataset, the thresholding selection is supposed to be very flexible for particular short and long shot videos. Consequently, we propose the double thresholding scheme (DTS) to efficiently remove the identical features based on the similarity score.

The recognition performances over three datasets using the sets of DTS are shown in Figure 5.3.1. We assign two thresholding values for the short and long shot videos. Based on the experiments, we consider a video as the long shot if the number of frames are more than 80. Otherwise, the videos are considered as short shot. Table 5.3.1 presents the set of higher and lower level thresholds for cosine and correlation distance metrics. These thresholding values have been evaluated for three employed datasets. For the cosine metric, the 9_{th} thresholding set outperforms others. We assign 0.18 and 0.13 as the higher and lower level threhsolds for the cosine metric. For the correlation metric, the 7_{th} thresholding set outperforms other sets. We assign 0.62 and 0.4 as the higher and lower thresholding levels for the correlation metric.

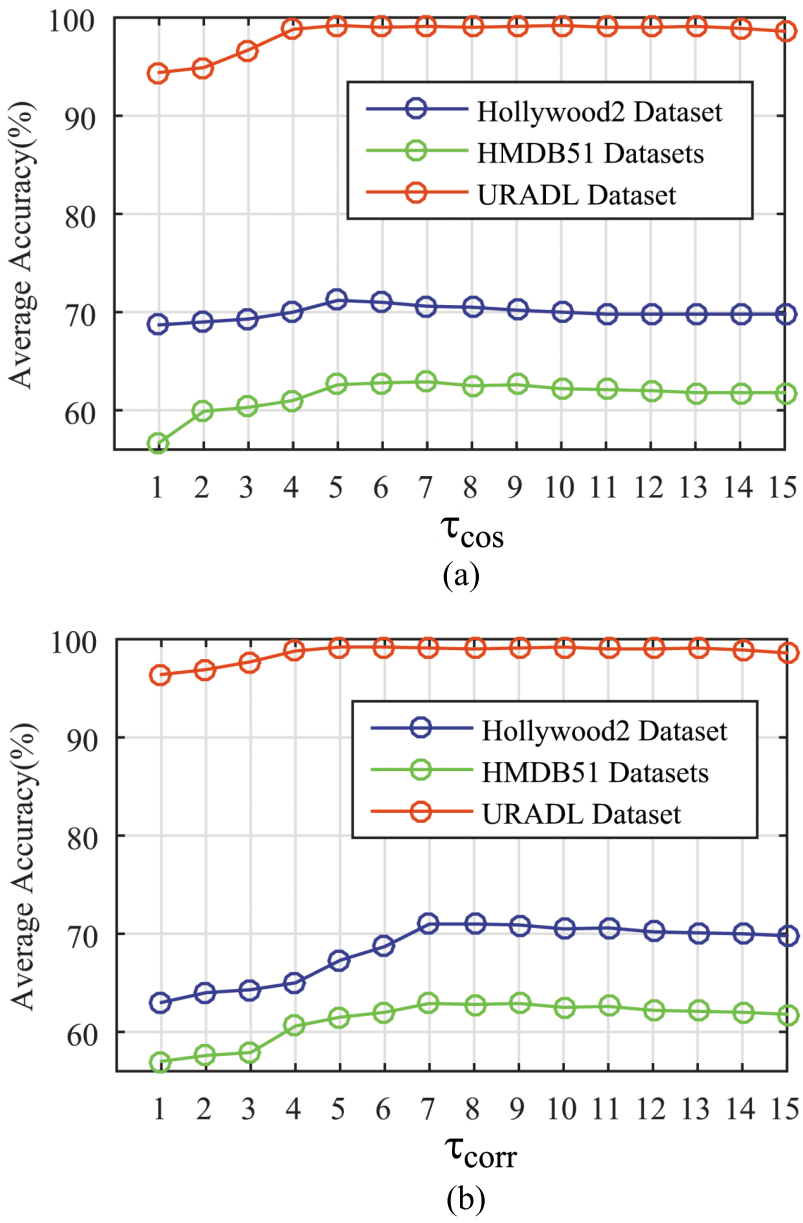


Figure 5.3.1: Action recognition performance using the improved rank pooling algorithm based on cosine (a) and correlation (b) distance metrics.

Table 5.3.1: Actual thresholding values for the similarity detection and elimination based on cosine and correlation distance metrics

Thresholding order	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Higher $\tau_{\cos}$	0.1	0.11	0.12	0.13	0.14	0.15	0.16	0.17	**0.18**	0.19	0.20	0.25	0.3	0.35	0.4
Lower $\tau_{\cos}$	0.5	0.6	0.7	0.8	0.9	0.10	0.11	0.12	**0.13**	0.14	0.15	0.17	0.19	0.21	0.23
Higher τ_{corr}	0.5	0.53	0.56	0.59	0.62	0.65	**0.68**	0.71	0.74	0.77	0.79	0.81	0.83	0.85	0.87
Lower τ_{corr}	0.2	0.25	0.3	0.35	0.4	0.45	**0.5**	0.55	0.6	0.65	0.67	0.69	0.71	0.73	0.75

Performance Evaluation

As shown in Figure 5.3.2, the IRP strategy outperforms or atleast results like regular rank pooling for all the actions of Hollywood2 dataset. It is worth to pointing out that the results for the ”hug” and ”handShake” actions are much better than the regular rank pooling performance. The mentioned actions are mainly have been performed with combination of other actions. For instance, people are talking or walking before or after the actual ”handshaking” action. We believe that the IRP is capable to remove the redundant data from this kind of complex actions efficiently before ranking the features. This can be considered as the main reason for outperforming of IRP over these actions. It should be noted that IRP works roughly the same as rank pooling for simple actions such as ”run” and ”situp”.

As shown in Figure 5.3.1, the cosine and correlation metrics are roughly results the same with different thresholding values. However, in terms of processing speed, the cosine metric is faster and can be considered as the best similarity metric for IRP. The protracted removal scheme outperforms the consequent removal scheme for all datasets. The results of consequent scheme is about the same as regular rank pooling. consequently, we can conclude that consequent scheme is not capable of removing all identical feature vectors from the original set. So, it leads to obtain the same results as regular rank pooling. However, the protracted scheme is capable of removing most similar feature vectors, and achieves reliable results.

Comparison of IRP to the state of the art

We compare the IRP performance with the state-of-the-art frameworks, as stated in Table 5.3.2. The proposed IRP methodology outperforms the regular rank pooling for all the three employed datasets. We also compare our results with improved dense trajectory (IDT) framework since we employ local features based on IDT to propose the IRP methodology. IRP results better than the IDT framework for all the datasets.

As reported in Table 5.3.2, the proposed IRP obtained the 70.6%, 63%, and 99.6% average accuracy for the Hollywood2, HMDB51, and URADL datasets. The improvement

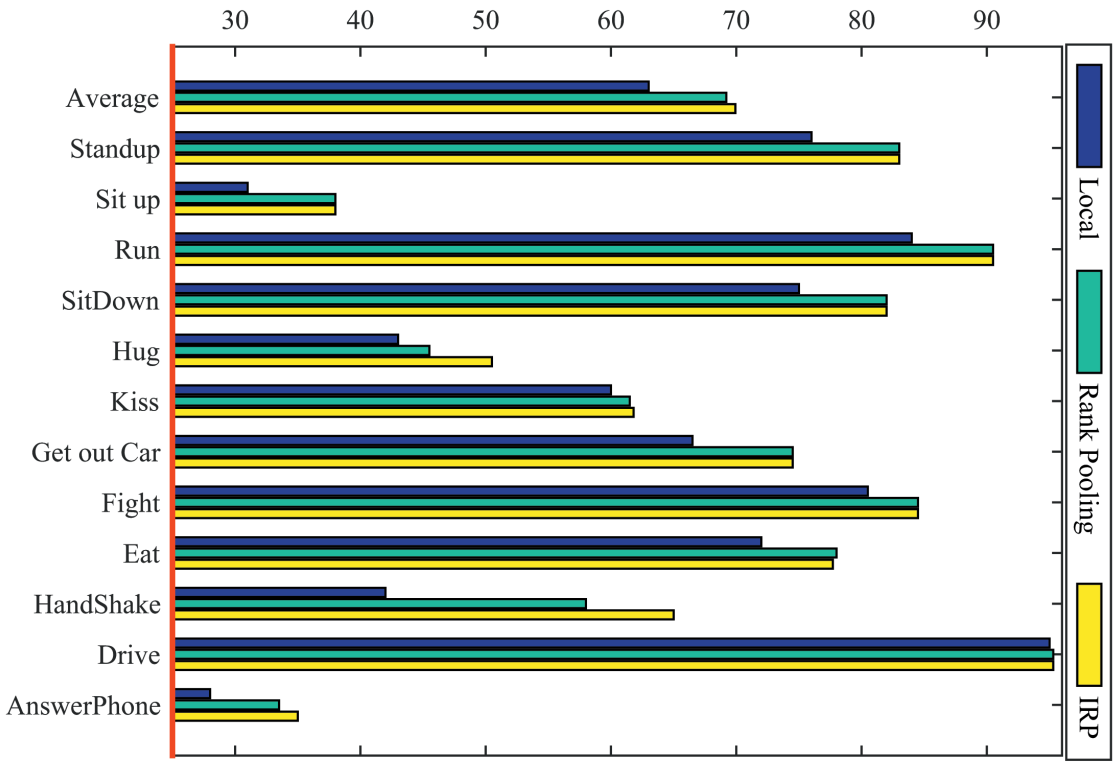


Figure 5.3.2: The action recognition performance on individual classes of Hollywood-2 dataset

Table 5.3.2: Comparison of our results, based on IRP, to the state-of-the-arts

Dataset	Method	Recognition Rate
Hollywood2	Wang et al. [31]	58.2%
	Jain et al. [32]	62.5%
	Fernando et al. [61]	69.6%
	Proposed Framework	**70.6%**
HMDB51	Wang et al. [31]	46.6%
	Jain et al. [32]	52.1%
	Fernando et al. [61]	61.6%
	Proposed Framework	**63.00%**
URADL	Prest et al. [100]	92%
	Wang et al. [31]	96%
	Eman et al. [102]	96.6%
	Proposed Framework	**99.6%**

of regular rank pooling shows a higher impact on URADL dataset since it is composed of only complex actions in very long shot videos. The average length of videos in URADL is about 16.16 seconds which is much higher than other two datasets. Consequently, we can conclude that IRP works better on long shot videos with complex actions.

5.3.2 Evaluation of Pooled-Feature Representation

Now, we describe the detailed evaluation of PFR-EF and PFR-LF trained by LSN. Finally, we conclude this section by comparing the obtained PFR results with the state-of-the-art. Since the proposed methodology is not specific to an action type or class of actions, we present experiments in a broad range of datasets. We follow the same training and testing splits as reported by the state-of-the-art methods. It should be noted that we keep the same experimental settings for training and testing over each dataset.

Experimental Setup

At the first phase of DPE framework, we extract the IDT features along with HOF, HOG, and MBH descriptors. We keep the settings of feature extraction as stated in [10]. Thus, the dimension of the HOG, HOF, and MBH descriptors are 96, 108, and 192 respectively. We split the local features, derived from each descriptor, into τ sub-volumes. Then, each temporal sub-volume is represented using the improved rank pooling (IRP) strategy. In the IRP, we employ PCA with a dimensionality reduction of size 64 before encoding the local features from τ sub-volumes. For encoding, we learn a visual dictionary by Gaussian mixture model (GMM) of codebook size K. Consequently, the dimension of encoded features is $128 \times K$ for each dataset.

At the second phase of DPE, the pool of features contain $128 \times K$ features and τ samples. We compress the pool using DL-SNN to extract the most prominent and useful features for encoding of the second DPE phase. Based on the experiments, the regularization parameter of DL-SNN can be selected randomly without affecting on recognition performance. Finally, we encode the compressed pool to generate VPV at the second phase of DPE. Additionally, we calculate the standard Fisher vectors (SFV) based on the same

parameters as for VPV. Now, we have all the required information to generate PFR-EF and PFR-LF.

The PFR-EF and PFR-LF are evaluated to determine the best option for complex action recognition. The evaluation is performed using the LSN classifier. Moreover, we compare the recognition performance of LSN, ELM, and SVM classifiers over the best option of PFR. For the SVM, linear activation function aligned with the cost of 100 is selected as the optimized parameters [10, 110, 111, 5]. For the ELM, we use 1000 hidden nodes and regularization parameter of 2^{-2} aligned with sigmoid activation function as the most reliable parameters [59]. For the LSN strategy, the optimized number of hidden nodes is equal to the output nodes. So, it is different based on the number of classes in each dataset. Thus, we use 10, 16, 51, 50, 101, and 12 hidden nodes for URADL, Olympic, HMDB51, UCF50, UCF101, and Hollywood2 datasets. The regularization parameter of the LSN is also examined, and it is shown that its changes do not affect the learning process.

Experimental Results

Effective number of temporal sub-volumes. Generating of temporal sub-volumes at the first phase of DPE framework is considered as the most important factor in PFR production. We compare different number of sub-volumes in DPE (see Table. 5.3.3). The evaluation is performed using the LSN classifier over six datasets. The best average of recognition performance is obtained by employing nine temporal sub-volumes as stated in Table. 5.3.3. For the URADL and Hollywood2 datasets, the VPV performance becomes better while increasing the number of sub-volumes. From the other side, the best VPV performance for HMDB51 is achieved by employing five sub-volumes since that dataset contains shortest video shots compared to other employed datasets.

It shows that VPV performance has a direct relation to the length of videos since the URADL and HMDB51 contain longest and shortest videos compare to other datasets. We perform all the future experiments by employing nine temporal sub-volumes since the best average of recognition performance is obtained using nine temporal sub-volumes. Thus, the maximum number of equal size sub-volumes are four on the third scale while generating the temporal sub-volumes of local features.

Table 5.3.3: Defining the best number of temporal sub-volumes to yield feature sets in the first phase of DPE.

Number of Sub-volumes	UCF50		URADL		Olympic Sports		HMDB51		UCF101		Hollywood2	
	IRV	VPV	IRV	VPV	IRV	VPV	IRV	VPV	IRV	VPV	IRV	VPV
2	91.2%	49.4%	94.0%	66.6%	90.2%	51.1%	59.4%	36.9%	89.4%	43.1%	69.4%	39.2%
5	91.9%	51.8%	95.3%	68.6%	91.0%	53.7%	59.4%	**39.9%**	90.0%	45.3%	69.6%	41.8%
9	**92.3%**	**52.1%**	**97.3%**	70.6%	**91.2%**	**59.7%**	**61.1%**	39.2%	**91.6%**	**49.2%**	69.6%	42.2%
14	92.1%	52.0%	96.6%	71.3%	91.0%	55.9%	59.4%	39.5%	90.4%	47.5%	**70.9%**	**43.5%**
20	90.5%	52.0%	94.0%	**72.6%**	89.5%	54.4%	57.2%	36.9%	89.7%	47.2%	69.5%	41.2%
27	90.0%	51.9%	94.0%	**72.6%**	85.8%	48.5%	55.9%	36.9%	88.9%	46.4%	68.9%	39.6%

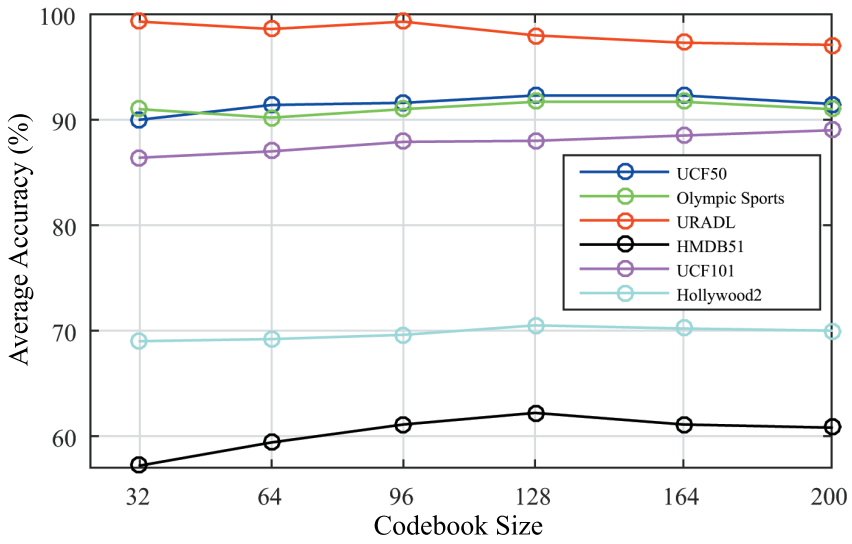


Figure 5.3.3: The effect of different codebook dimensions while encoding the features for six employed datasets.

Effective codebook dimension for encoding the features. We evaluate a set of codebook sizes, $\mathbf{K} = \{32, 64, 96, 128, 164\}$, to encode the features at the first and second phases of DPE framework (see Figure 5.3.3). For each dataset, the same codebook size is allocated to both DPE phases. The experiments have been performed by feeding the features to LSN classifier. Based on the experiments, the best codebook size for the UCF50, Olympic, HMDB51, and Hollywood2 is 128, for URADL is 32, and for UCF101 is 200. It should be noted that we already reduced the size of the features to 64 using the PCA approach. Thus, the dimension of the generated features is 4096 for URADL, 25600 for UCF101, and 16384 for the rest of datasets.

Evaluate the regularization parameter in DL-SNN. Parameter C is the only adjustable factor for compressing of features at the second phase of DPE. It is selected from the set of $\mathbf{C} = [2^{-4}, \ldots, 2^{8}]$. We analyze the effect of the parameter C on the recognition performance over six employed datasets. The fused vector, derived from VPV and IRVs, is fed to LSN for classification. We adopt the same C for DL-SNN and LSN during experiments. The encoding is performed using the best codebook size for each dataset. Experimental results show that generalization performance of the DL-SNN is not sensitive to the parameter C. Therefore, we can select the parameter randomly at the outset without

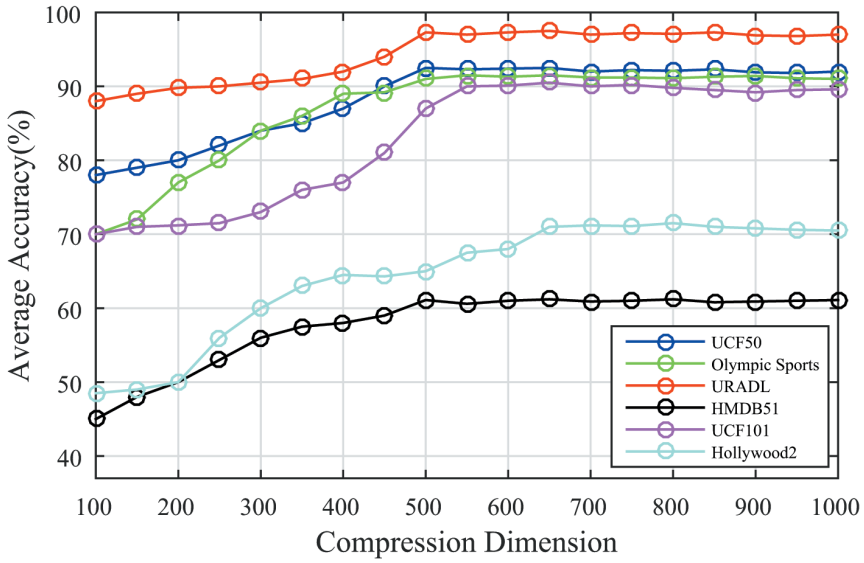


Figure 5.3.4: Evaluation of different compression rates at the second phase of DPE for the employed datasets.

affecting the generalization performance in the learning process.

Effect of compression rates on VPV performance. The same compression size, selected from $\mathbf{d} = \{100, 150, 200, \ldots, 1000\}$, is used for HOG, HOF, and MBH descriptors. Then, the compressed version of each descriptor is encoded to generate VPVs. The VPVs and IRVs of the employed descriptors are fused and then fed to LSN for classification. The evaluation is performed using different dimensions in set $\mathbf{d}$. We obtain the most accurate results by compressing the features to 500 dimensions for all datasets (see Figure 5.3.4). The experiments show that increasing the compression dimension over 500 has no improving effect on the results. The reason might be related to the sparsity of generated vectors using DL-SNN. Generally, when the compression rate is increased, the sparsity of compressed vectors is increased. And the sparsity of compressed vectors affects the encoding results at the second phase of DPE.

Effect of compression rates on PFR performance. We analyze the sets of compression scales for generating PFR-EF and PRF-LF (see Figure 5.3.5). In order to provide the PFR-EF, dimensions from the set $\mathbf{d} = [1000, 2000, ..., 10000]$ are evaluated to compress the fused features. We evaluate the compression dimensions on HOG, HOF, MBHx and MBHy separately. The experimental results show that the best compression size is 5000 for mo-

tion features and 7000 for appearance-based features. It shows that the appearance-based features are more sensitive while compressing and we may miss some vital information during the compression. However, we can extract the more useful and reliable features while compressing the motion-based features. In other words, motion-based features contain more redundant information related to action recognition. Based on the experiments, the best generated PFR-EF is composed of 22000 features and M video samples based on four descriptors.

For producing the PFR-LF, each of the single feature sets are first compressed based on the dimensions in set $\mathbf{d} = [100, 200, ..., 2000]$. Then, the compressed features are fused to generate the PFR-LF. As shown in Figure 5.3.5, 500 and 700 are roughly considered as the best compression dimensions for motion and appearance-based features respectively. We compress $\tau + 2$ encoded vectors for each descriptor where $\tau = 9$ in our experiments. Since we use four descriptors, 44 compressed vectors are fused to generate the PFR-LF. Based on the experiments, the best PFR-LF contains 24200 features and M samples based on the employed descriptors.

Effect of the length of videos on DPE framework. We analyze how the length of videos affects the VPV recognition performance over UCF50 dataset. We use the fusion of VPV and IRVs which are derived from 9 temporal sub-volumes, as the input to LSN classifier. Then, we compare the recognition accuracies achieved with different video lengths. As shown in Figure 5.3.6, the recognition performance is enhanced by increasing the length of videos. This is not surprising since longer videos are composed of more dynamic information compared to shorter videos. Also, generating sub-volumes with small scales for shorter videos will likely be more affected by outliers. It should be noted that relative recognition performance between very long and concise videos, is about 10% and 50% for IRVs and VPV respectively. We conclude that we are capable of capturing the dynamics of long-shot videos more efficiently than short videos.

Analysis of classifiers. Looking at the results of UCF50 dataset, we observe that LSN, SVM, and ELM respectively achieve 92.45%, 90.4%, and 88.6% for the first train-test group. The training time for SVM, LSN, and ELM are 9735, 1365, and 364 seconds respectively for the first train-test group. The experiments show that LSN outperforms

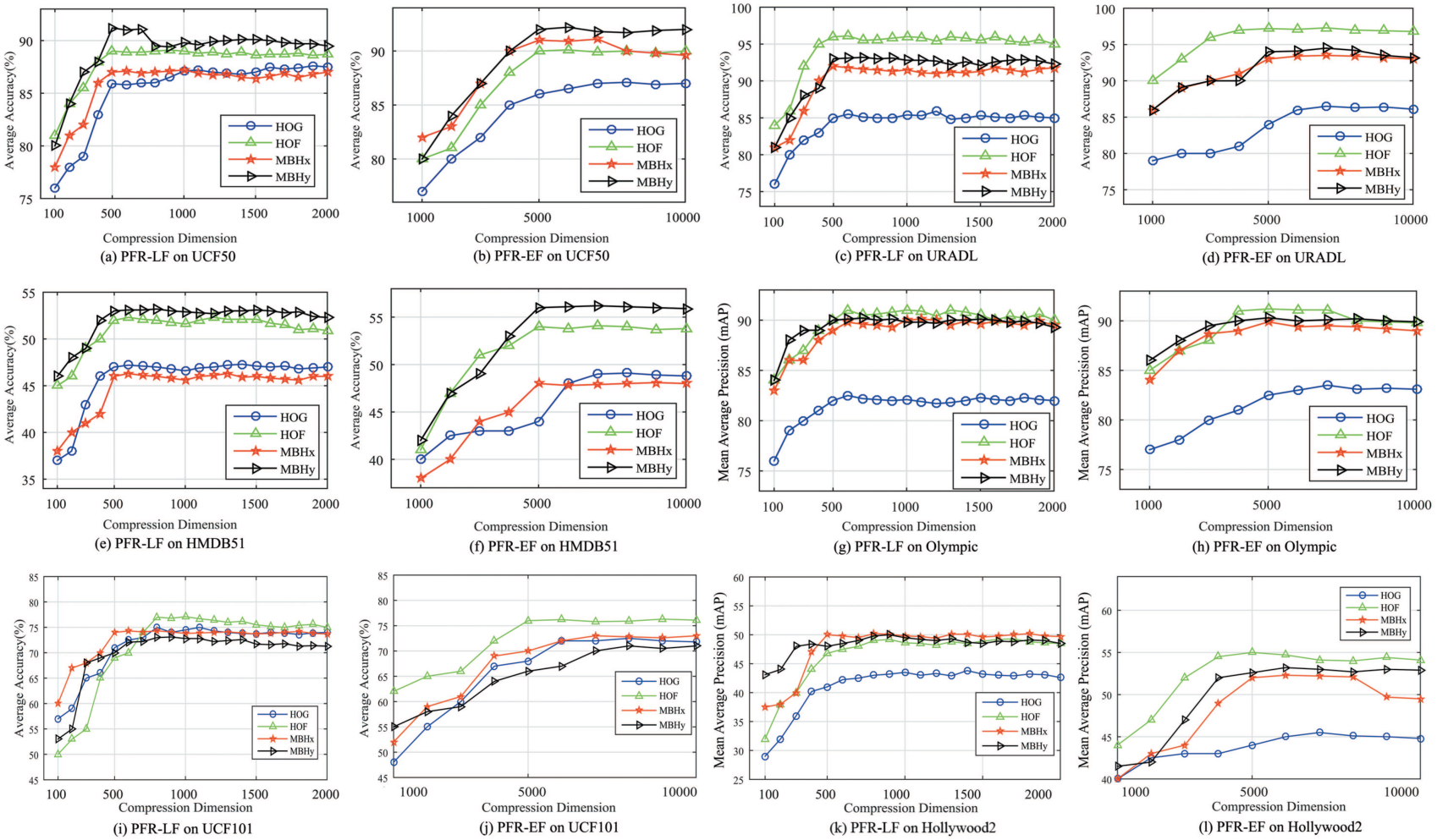


Figure 5.3.5: Evaluation of compression rates to create PFR-EF and PFR-LF feature vectors over six benchmark datasets.

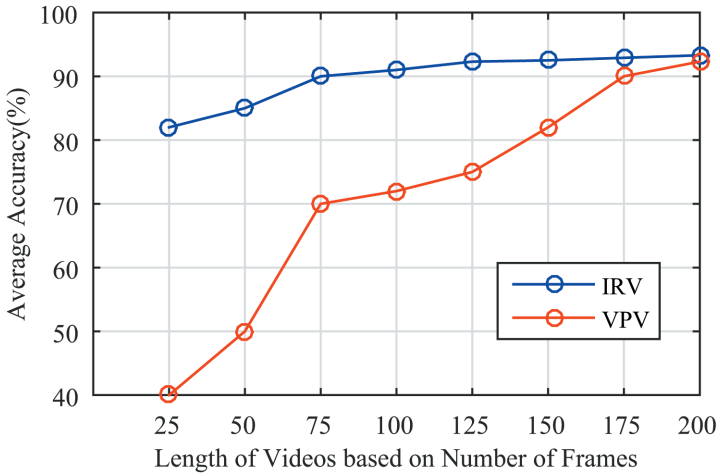


Figure 5.3.6: The effect of video length on IRVs and VPV performances over UCF50 dataset.

Table 5.3.4: Comparison of sigmoid and sine activation functions in LSN over six benchmark datasets.

Activation Function	UCF50	URADL	Olympic	HMDB51	UCF101	Hollywood2
Sigmoid	94.9%	100.0%	91.4%	63.2%	91.8%	71.4%
Sine	94.6%	100.0%	91.2%	63.0%	91.2%	70.8%

standard SVM in both accuracy and training speed. However, compared to the standard ELM, LSN is capable of boosting the recognition performance while having a slower training speed. Additionally, we evaluate the effect of sigmoid and sine activation functions in the LSN classifier (see Table. 5.3.4). Based on the obtained results using six benchmark datasets, the sigmoid activation function outperforms the sine function. Thus, we conclude that the sigmoid function is more differentiable with respect to the network parameters and this property plays a vital role in the training of LSN classifier.

Analyse the results and compare with state-of-the-arts. As shown in Figure 5.3.7, the action recognition performance using PFR-EF outperforms PFR-LF in six benchmark datasets. Additionally, PFR-EF provides better results compared to the VPV, SFV, and IRVs. The obtained features using PFR-LF and PFR-EF are trained and tested by LSN classifier. We summarize the results of proposed PFR-LF and PFR-EF in Table 5.3.5. Additionally, we compare the obtained results with state-of-the-art action recognition frameworks.

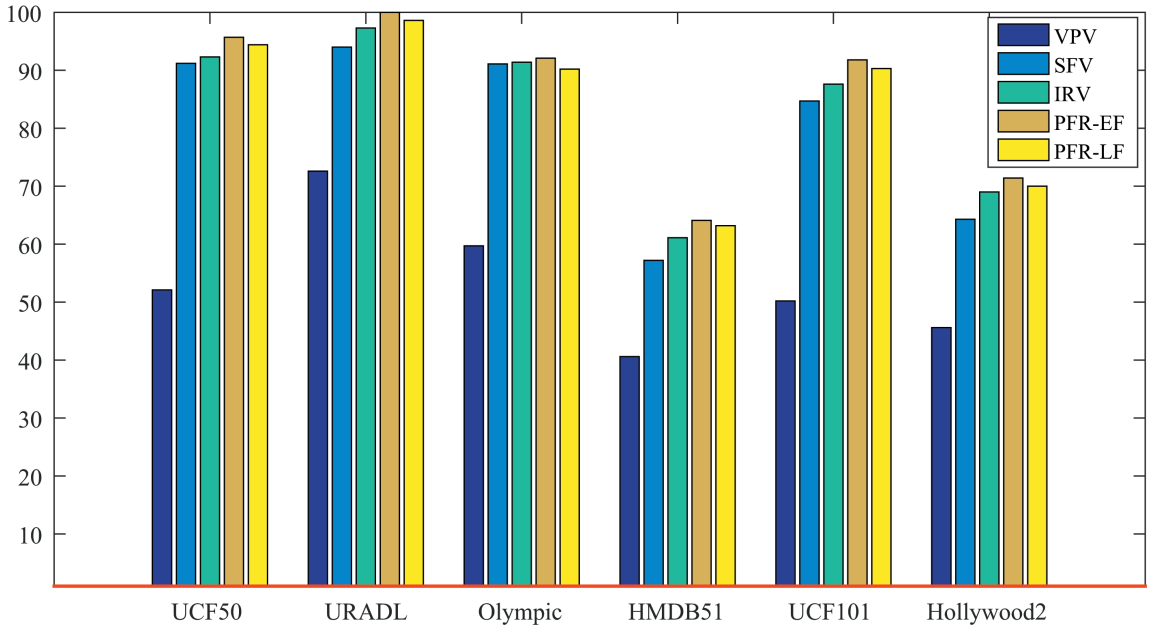


Figure 5.3.7: Compare the action recognition performances using VPV, IRVs, SFV, PFR-EF, and PFR-LF.

As shown in Table 5.3.5, the action recognition performance is surprisingly improved for URADL datasets. The URADL contains long-shot videos. Based on the lengthy complex videos, we obtain 100% accuracy for the URADL. The recognition performance is boosted for UCF50 dataset by obtaining 94.9% average accuracy over 25 groups of testing and training splits. As stated in Table 5.3.5, we boost the results compared to the low-level [19, 45, 112, 106] and mid-level [46] video representation approaches. The improvement is marginal since the average length of videos in UCF50 is 7.00 seconds. The recognition performance of the HMDB51 is boosted compare to the results of low-level and high-level video representation methods [113, 114, 61, 6]. We compare the results of HMDB51 to regular rank pooling approach [61] since we use the improved rank pooling strategy at the fisrt phase of DPE to represent the temporal sub-volumes. Since HMDB51 dataset contains shorter videos compare to other employed datasets, the recognition performance is marginally improved. For the Olympic dataset, our results are the same as the presented results by the state of the arts. Even though the length of videos of Olympic is roughly the same as UCF50, smaller temporal sub-volumes are not capable of producing informative features for some specific actions at the first phase of DPE. For instance, the initial

smaller sub-volumes of the high jump, long jump, and triple jump actions provide almost the same features since the athletes are running with the same style in the three stated actions. Thus, due to the similarity between some of the sub-volumes of specific actions, the PFRs are not easily discriminated. The involved similarities make the classification more challenging. We achieved 91.4% based on mean average precision (mAP) over 16 classes of the Olympic dataset. For the UCF101 and Hollywood2 datasets, the recognition performances are boosted as shown in Table 5.3.5. We obtained reliable results for Hollywood2 dataset even though it contains complex videos with different qualities. The average of video lengths in Hollywood2 dataset is 13.73 seconds. The long-shot video samples aim to extract more informative features using the proposed PFR-EF method. We compare the recognition performances of UCF101 and Hollywood2 with standard Fisher vector [10], and state-of-the-art low-level and mid-level video representations [115, 106, 45].

We draw several vital conclusions by the inspection of Table 5.3.5 and the results of previous experiments. First of all, PFR-EF is a robust framework to represent unconstrained videos. Second, the proposed method is complementary to local feature extraction methods such as improved dense trajectory features. Furthermore, we show that fusion of all individual represented vectors at the first phase of DPE and the VPV from the second phase of DPE can boost the recognition performance in complex videos. Consequently, the PFR-EF is capable of representing a given video by extracting the most vital information from appearance-based and motion features. Additionally, we conclude that PFR-EF impressively enhances the action recognition performance for long-shot video samples.

5.4 Summary

In this chapter, we introduce pooled-feature representation (PFR), a new methodology to represent unconstrained videos with the most prominent features. The PFR aggregates the relevant information throughout a video via fusion of low-level and high-level representations that are derived from the proposed double phase encoding (DPE) framework. DPE is a multi-stage framework to represent the higher level information by employing two encoding phases joined with the compression stage. The combination of the individ-

Table 5.3.5: Comparison of our results, based on PFR, to the state-of-the-arts

Dataset	Method	Recognition Rate
UCF-50	Wang et al. [46]	93.8%
	Tran and Torresani [45]	82.8%
	Liu et al. [19]	75.6%
	Wang [106]	91.7%
	Zhang et al. [112]	94.5
	Proposed PFR-LF	94.4%
	Proposed PFR-EF	**94.9%**
HMDB51	Liu et al.[113]	57.2%
	Xu et al. [114]	59.5 %
	Fernando et al.[61]	61.8%
	Liu et al. [6]	51.4%
	Tran and Torresani [45]	41.9%
	Proposed PFR-LF	62.2%
	Proposed PFR-EF	**63.2%**
URADL	Yuan et al. [42]	92%
	Prest et al. [100]	92%
	Bilibski et al. [101]	94.7%
	Wang et al. [5]	96%
	Eman et al. [102]	96.6%
	Proposed PFR-LF	97.33%
	Proposed PFR-EF	**100.00%**
Olympic Sports	Lan et al. [116]	91.4%
	Wang et al. [10]	91.1%
	Jiang et al.[115]	91.0%
	Wang et al.[106]	90.4%
	Li and Dai [107]	91.4%
	Proposed PFR-LF	91.0%
	Proposed PFR-EF	**91.4 %**
UCF101	Lan et al. [116]	89.1%
	Wang et al. [10]	84.7%
	Jiang et al.[115]	87.2%
	Wang et al.[106]	86.0%
	Tran and Torresani [45]	71.6%
	Proposed PFR-LF	91.3%
	Proposed PFR-EF	**91.8 %**
Hollywood2	Lan et al. [116]	68.0%
	Wang et al. [10]	64.3%
	Jiang et al.[115]	65.4%
	Wang et al.[106]	69.4%
	Tran and Torresani [45]	56.6%
	Proposed PFR-LF	70.0%
	Proposed PFR-EF	**71.4 %**

ual represented vectors from DPE phases allows preserving richer information from the temporal domain of local features.

Additionally, we propose the improved rank pooling (IRP) strategy to represent the individual temporal blocks at the first phase of DPE framework. The IRP captures the optimized latent structure of the video sequence data and describes a given video based on its temporal contents. The improvement of rank-pooling is followed by eliminating the identical features, derived from the video frames. As a case study, we compare the recognition performance by employing the cosine and correlation distance metrics in detection of identical features. Furthermore, we propose the double thresholding scheme (DTS) to efficiently measure the similarity of couple vectors. We also propose two strategies for detecting of similar features, called consequent and protracted checking methods where the later achieves the best results. The cosine and correlation metrics achieve roughly the same results with different thresholding sets.

Based on extensive experimental evaluations, we conclude that our method applies to a wide variety of datasets for capturing the prominent temporal information of a video. Our architecture is trained and evaluated over six challenging datasets, namely UCF50, URADL, Olympic, HMDB51, UCF101, and Hollywood2. The best recognition performance is achieved for URADL dataset with 100% accuracy since it contains the long-shot videos of complex actions. We conclude that our proposed methodology is novel and accurate for capturing the most salient information from the temporal domain of unconstrained videos.

CHAPTER 6

Key Action Perception for Constrained and Unconstrained Video Analysis

6.1 Overview

The conventional action recognition frameworks extract and encode the features of the entire given video. However, a given video may contain inappropriate actions and motions which are entirely different from the key action. The inappropriate motions provide noises in the feature sets of a given video and contribute challenges to the action classification part. To address this problem, we propose the key action perception (KAP) along with a robust video action clustering for unconstrained and constrained video analysis. The KAP includes two classifiers: the former detects the key action among multiple temporal clusters, and the latter recognizes the key action which is obtained by the former classifier.

The video action clustering is the essential pre-processing step for KAP implementation. The sequential relationship of the video frames and complexity of motion representations provide challenges in video action clustering. We propose two novel multi-layer subspace video action clustering (ML-VAC) techniques to encode the sequential relationships of constrained and unconstrained video frames. We obtain the expressive coding information by analyzing the motion features of a given video. Then, the affinity graph is constructed using the coding information and multiple temporal clusters are defined without having any prior knowledge about the number of temporal clusters in a given video.

The rest of the chapter is divided as follows: Section 6.1 is divided into a two subsections to discuss the proposed key action perception, followed by the proposed approach for video action clustering. Section 6.2 provides an explanation of the algorithm followed

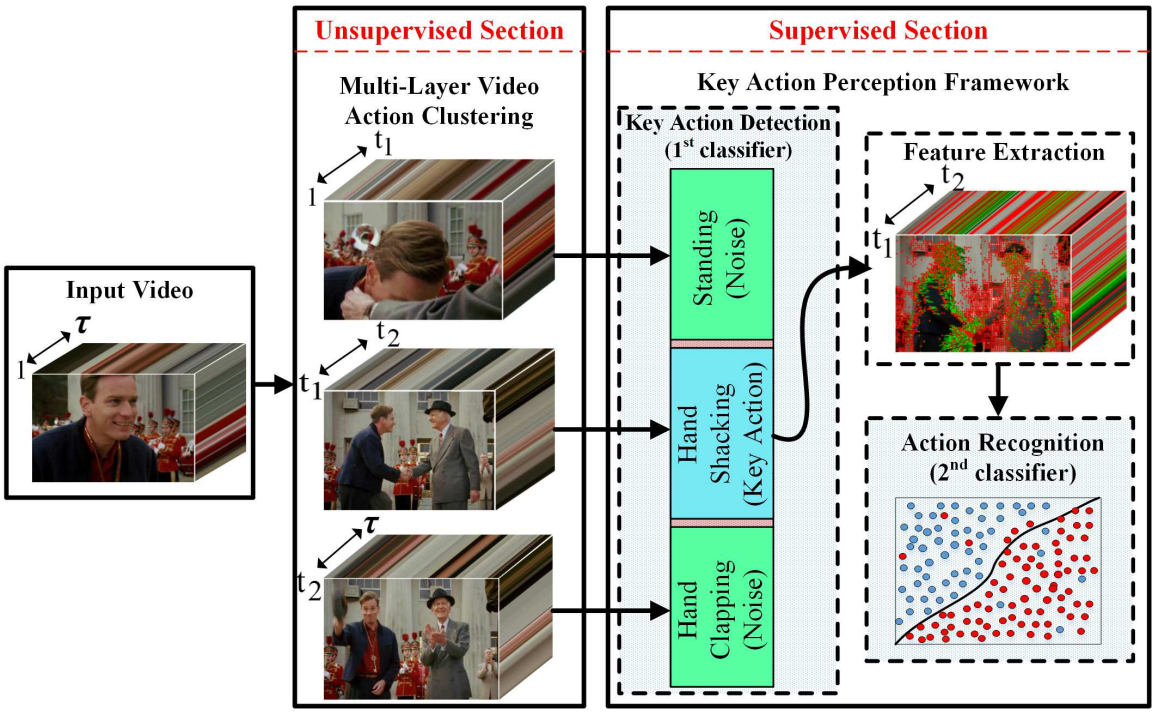


Figure 6.2.1: Proposed key action perception framework including unsupervised and supervised sections.

by several types of experimental results and comparisons with some of the state-of-the-art video clustering and action recognition strategies. Finally, Section 6.3 concludes this chapter by summarizing the proposed framework and obtained results.

6.2 Video Analysis with Key Action Perception

Localization and recognition of key actions are the fundamental problems in video analysis. In this section, we first describe the architecture of proposed KAP as shown in Figure 6.2.1, and then explain the ML-VAC framework based on the proposed UVAC and CVAC algorithms as depicted in Figure 6.2.2 and Figure 6.2.3.

6.2.1 Key Action Perception (KAP)

The ultimate goal of KAP is to detect and recognize the key action among the $\mathbf{C} = \{c_1, \ldots, c_N\}$ plausible actions using two classifiers (see Figure 6.2.1). The N clusters are classified into noise and key actions by the former classifier and $\mathbf{L} = \{l_1, \ldots, l_N\}$ labels

are produced where the l_k indicates the label for the key action cluster. Then, the features of the key action cluster $\bar{v}(c_k|l_k)$ are adopted to recognize its action by the latter classifier. In order to train the former KAP classifier, the video clusters from each dataset are supposed to be manually labelled as noise and key actions. Thus, we manually labelled the temporal clusters of the available video datasets into key and noise actions. The key action clusters along with the available labels in the employed datasets are adopted to train the latter KAP classifier. Even though the KAP is independent from the choice of video features, we adopt our proposed improved rank pooling strategy[117] as video representation for both key action detection and recognition parts.

The videos must be carefully clustered into plausible actions for boosting the action recognition performance in KAP. The complexity of motion representation, background clutter, and sequential relationship of video frames contribute challenges to the video action clustering task. We assume that each particular video application requires specific clustering algorithm. For instance, the camera and background are stable in the video surveillance applications. However, we may have dynamic camera and background with different lighting conditions and action variations in video retrieval applications. In other words, the complexity of video contents is different in constrained and unconstrained videos. We propose multi-layer video clustering approaches which consider the sequential information of video frames to perform time-series action clustering in both constrained and unconstrained videos. The following section thoroughly describes the proposed clustering methods.

6.2.2 Video Action Clustering

Subspace clustering is an effective strategy to group the video frames into low-dimensional subspaces [81, 118, 78, 119]. The traditional subspace clustering methods consider $\boldsymbol{V} = \{\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_{\boldsymbol{\tau}}\}$ as a set of data vectors in a D-dimensional Euclidean space where each column represents a sample. Subspace clustering aims to cluster $\boldsymbol{\tau}$ data vectors into their respective subspaces. In the computer vision domain, the variable $\boldsymbol{\tau}$ refers to the number of frames in a given video as shown in the unsupervised section of Figure 6.2.1. The conventional subspace clustering methods oversight the temporal information in data, which make

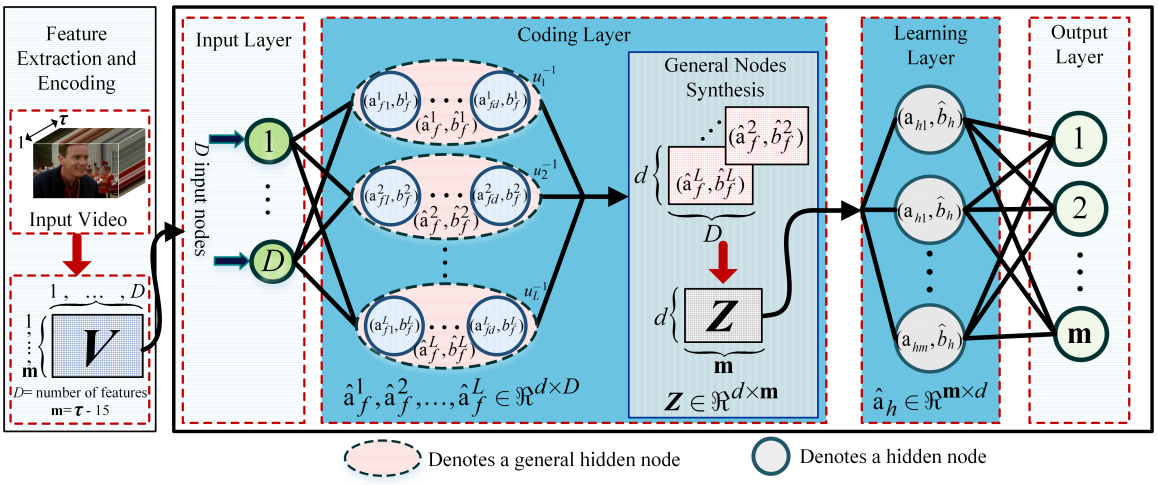


Figure 6.2.2: Proposed ML-VAC for unconstrained video action clustering (UVAC).

them inadequate for the time series clustering. We propose the multi-layer subspace video action clustering (ML-VAC) method considering the temporal relationship in data vectors. The proposed ML-VAC is capable of clustering a given video into non-overlapping plausible actions based on the three following steps.

1) Feature Extraction and Encoding. Given a video with $\boldsymbol{\tau}$ frames, let $\boldsymbol{V}$ be a sequence of feature vectors $\boldsymbol{V} = \{\bar{v}_1, \ldots, \bar{v}_{\boldsymbol{\tau}}\} \in \Re^D$ from a D-dimensional space representing a given video. It is worth pointing out that the context of the temporal change is more important than the spatial context for video action clustering. Thus, ML-VAC adopts the motion features as improved dense trajectories along with the histogram of optical flow (HOF) descriptor [10] to prepare the sequence of feature vectors.

The HOF features are represented by Fisher vector encoding for each frame. It should be noted that the length of trajectories is kept as 15 since it contributes the best action recognition performance [10]. Therefore, the features from each individual frame and its 15 consecutive frames are encoded and the whole video is presented as $\boldsymbol{V} = \{\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_{\boldsymbol{\tau}-15}\}$ where $\boldsymbol{\tau}$ indicates the number of frames in each video. The $\boldsymbol{V}$ includes D features and $\mathbf{m}$ samples where $\mathbf{m} = \boldsymbol{\tau} - 15$ as depicted in Figure 6.2.2. The video representation data is utilized for coding matrix generation in ML-VAC algorithm. We propose two techniques for coding matrix generation to properly deal with constrained and unconstrained videos.

2) Coding Matrix Generation. We employ the multi-layer learning paradigms to

produce the coding matrix ($\boldsymbol{Z}$) by analysing the extracted features from a given video. The generated coding matrix is required for affinity graph construction in the clustering framework. The achieved matrix $\boldsymbol{Z}$ contains the most informative motion features from a given video with respect to the temporal order of the frames. Each feature vector in $\boldsymbol{Z} = \{z_1, z_2, ..., z_m\}$ is an alternative representation of the associated video features.

We propose two methods for coding matrix generation in the unsupervised mode since the prior knowledge about the content of any given video is unavailable. Additionally, any information about the number of action clusters in videos is unavailable. The first framework works well for unconstrained video action clustering (UVAC). The second framework aims to generate robust coding matrices for constrained video action clustering (CVAC). The UVAC system can be employed for the video retrieval applications where the camera and backgrounds may have a dynamic form. The CVAC system can be used for the video surveillance applications where the camera and backgrounds are static. The following sections describe the two proposed methods with details.

2.1) Unconstrained Video Action Clustering (UVAC). We adopt a multilayer network with general nodes, inspired by [93], for coding matrix generation. The $\mathbf{m}$ encoded feature vectors of the video frames are labelled based on the order of frames as $\mathbf{F} = \{\Phi_1, \Phi_2, \ldots, \Phi_{\mathbf{m}}\}$ with respect to $\boldsymbol{V} = \{\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_{\mathbf{m}}\}$. The encoded features and their labels are fed to the multilayer framework and then the parameters of the model are utilized as coding matrix for clustering (see Figure 6.2.2). The learning steps are based on the inverse activation functions and L general nodes to obtain the optimized parameters in coding layer. The following five steps are performed to generate the coding matrix ($\boldsymbol{Z}$).

Step 1. Given $\mathbf{m}$ arbitrary distinct training data vectors $\{(\boldsymbol{V}_k, \mathbf{F}_k)\}_{k=1}^{\mathbf{m}}$, $\boldsymbol{V}_k \in \Re^D, \mathbf{F} \in \Re^{\mathbf{m}}$, the parameters of the initial general node in coding matrix layer are randomly generated as

$$\begin{aligned} &\boldsymbol{Z}_f^j = \mathbf{g}_s\left(\hat{\mathbf{a}}_f^j.\ \boldsymbol{V} + \hat{\mathrm{b}}_f^j\right), \left(\hat{\mathbf{a}}_f^j\right)^T.\hat{\mathbf{a}}_f^j = \mathbf{I}, \left(\hat{\mathrm{b}}_f^j\right)^T.\ \hat{\mathrm{b}}_f^j = 1 \\ &\mathbf{g}_s(\cdot) = sin(\cdot) \end{aligned} \tag{6.2.1}$$

where j is set to 1, $\boldsymbol{Z}_f^j$ is the current coding matrix, and $\hat{\mathbf{a}}_f^j \in \Re^{d\times D}$, $\hat{\mathrm{b}}_f^j \in \Re$ are the

orthogonal random weight and bias of coding matrix layer. The variable d refers to the number of sub-network nodes in each general hidden node in the coding layer of UVAC framework. In the experiments, we show the optimized variable d to generate the coding matrix of an unconstrained video.

Step 2. For any continuous desired output $\mathbf{F}_k$, the parameters in the learning layer are computed as

$$\begin{aligned} \hat{\mathbf{a}}_h &= \mathbf{g}_t(u_n(\mathbf{F})) \cdot \left(\boldsymbol{Z}_f^j\right)^{-1} \quad , \hat{\mathbf{a}}_h^j \in \Re^{d \times m} \\ \hat{b}_h &= \sqrt{\text{mse}\left(\hat{\mathbf{a}}_h^j \cdot \boldsymbol{Z}_f^j - \mathbf{g}_t(u_n(\mathbf{F}))\right)} \quad , \hat{b}_n^j \in \Re \\ \mathbf{g}_t(\cdot) &= arctan(\cdot) \end{aligned} \tag{6.2.2}$$

where $\boldsymbol{Z}^{-1} = \boldsymbol{Z}^T(C\mathbf{I} + \boldsymbol{Z}\boldsymbol{Z}^T)^{-1}$ while C is a constant value, and u_n is a normalized function while $u_n(\mathbf{F}) : \Re \rightarrow (0, 1]$. The parameter C affects the diagonal of $\boldsymbol{Z}\boldsymbol{Z}^T$ and modifies the stability of the network while producing the coding matrix. We examine a set of values for parameter C and show its effect on coding matrix generation in Section IV. Furthermore, we use the inverse of the tangent function to update the nodes of learning layer.

Step 3. Update the output error $\mathbf{e}_j$ and error feedback data as

$$\mathbf{e}_j = \mathbf{F} - u_n^{-1}\mathbf{g}_s(\boldsymbol{Z}_f^j, \hat{\mathbf{a}}_h, \hat{b}_h), \tag{6.2.3}$$

$$\mathbf{P}_j = \mathbf{g}_s(u_n(\mathbf{e}_j)) \cdot (\hat{\mathbf{a}}_h)^{-1} \tag{6.2.4}$$

where u_n^{-1} represents the reverse function of u_n.

Step 4. Set $j = j + 1$ and add a new general node $\hat{\mathbf{a}}_f^j, \hat{b}_f^j$ in the coding layer as

$$\begin{aligned} \hat{\mathbf{a}}_f^j &= \mathbf{g}_s^{-1}(u_j(\mathbf{P}_{j-1})) \cdot \boldsymbol{V}^{-1} \quad , \hat{\mathbf{a}}_f^j \in \Re^{d \times D} \\ \hat{b}_f^j &= \sqrt{\text{mse}(\hat{\mathbf{a}}_f^j \cdot \boldsymbol{V} - \mathbf{P}_{j-1})} \quad , \hat{b}_f^j \in \Re \\ \mathbf{g}_s^{-1}(\cdot) &= arcsin(\cdot) \end{aligned} \tag{6.2.5}$$

and Update the coding matrix as

$$\boldsymbol{Z}_f^j = \sum_{t=1}^{j} u_t^{-1} \mathbf{g}_s(\boldsymbol{V}, \hat{\mathbf{a}}_f^t, \hat{b}_f^t) \tag{6.2.6}$$

Step 5. Repeat steps 2 to 4 for $L-1$ times where L is equal to the number of general nodes as shown in the coding layer of Figure 6.2.2. The parameters $\{\hat{\mathbf{a}}_f^j, \hat{b}_f^j\}_{j=1}^{L}$ are *optimal projecting parameters* in the coding layer. It should be noted that L is equal to the number of feature vectors ($\mathbf{m}$) from any input video and it defers based on the length of different video streams which are used as the input to UVAC network.

The coding matrix is generated based on the features of video frames. The classes are created based on the order of video frames $\mathbf{F} = \{\Phi_1, \Phi_2, \ldots, \Phi_{\mathbf{m}}\}$. Thus, only one sample is available for each class. The first general hidden node and its bias $\{\hat{\mathbf{a}}_f^1, \hat{b}_f^1\}$ are generated randomly in the first step and the next general hidden nodes are inspired by the first general node which was created randomly. In order to remove the redundant and non-optimized parameters, we make the final coding matrix as

$$\boldsymbol{Z}_f^L = \frac{\sum_{j=2}^{L} u_j^{-1} \mathbf{g}_s(\boldsymbol{V}, \hat{\mathbf{a}}_f^j, \hat{b}_f^j)}{L-1} = \boldsymbol{Z}^* \tag{6.2.7}$$

without considering the randomly created data in the first general node. The generated coding matrix, $\boldsymbol{Z} \in \Re^{d \times \mathbf{m}}$, contains optimal data for affinity matrix calculation from unconstrained videos.

2.2) Constrained Video Action Clustering (CVAC).

The CVAC aims to produce robust coding matrices for constrained videos. Figure 6.2.3 shows the structure of the employed system. The CVAC contains the autoencoder format, inspired from [120], where the input data are encoded and decoded in sequential layers for training the coding matrix of the network. The obtained information in the encoding layer are utilized as the coding information for affinity graph construction. As shown in Figure 6.2.3, the employed model tries to construct the input data in the output layer. Thus, we have D-dimensional video data in the input and output layers. The following four steps are implemented to produce the coding matrix in the encoding layer of CVAC.

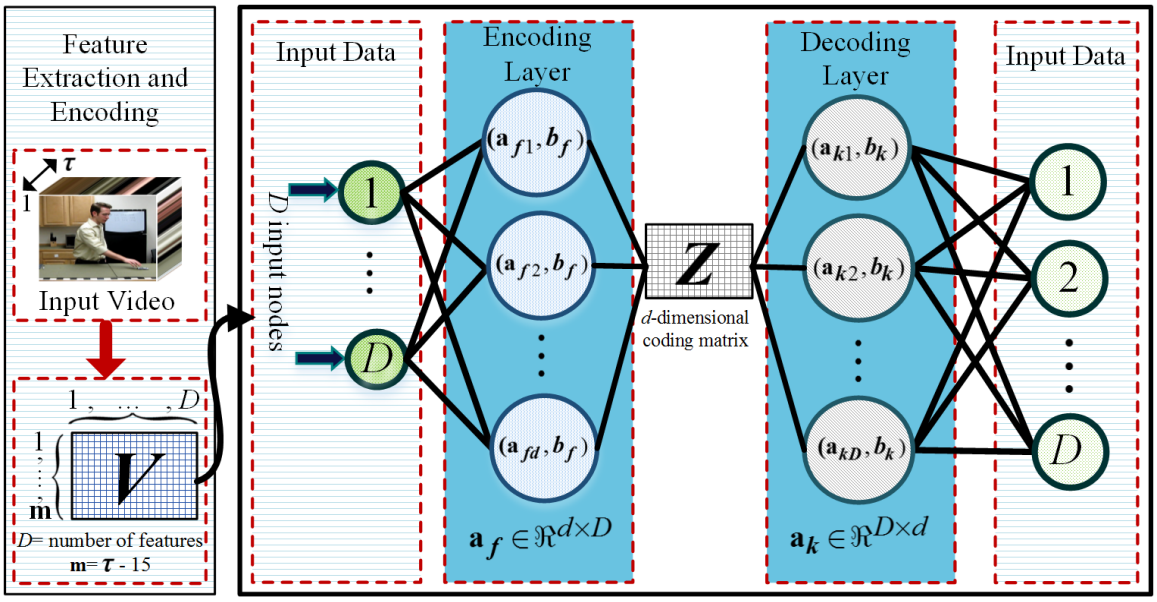


Figure 6.2.3: The framework of coding matrix generation for constrained video action clustering (CVAC).

Step 1. We randomly generate the initial parameters in the encoding layer as

$$\boldsymbol{Z}_f = \mathbf{g}_s(\mathbf{a}_f.\boldsymbol{V} + \boldsymbol{b}_f) \tag{6.2.8}$$

where $(\mathbf{a}_f)^T.\mathbf{a}_f = \mathrm{I}$ and $(\boldsymbol{b}_f)^T.\boldsymbol{b}_f = 1$. The $\mathbf{a}_f \in \Re^{d\times D}$ are the random parameters of the encoding layer in CVAC. The $\boldsymbol{Z}_f$ is the initial coding data and $\mathbf{g}_s$ is the sine activation function. It should be noted that unlike the UVAC framework, we use the inverse of the same activation function in the decoding layer of CVAC framework.

Step 2. We calculate the parameters of the decoding layer using the inverse of sine activation function as $\mathbf{a}_k = \arcsin(\mathbf{y}).(\boldsymbol{Z}_f^{-1})$ where $\mathbf{y}$ is the output of the CVAC which should be equal to the input data in the optimal format. The $\boldsymbol{Z}_f^{-1}$ is the Moore-Penrose inverse of matrix $\boldsymbol{Z}_f$. The major motivation behind usage of Moore-Penrose inverse is to achieve the minimum norm in least-squares solutions [121]. The $\boldsymbol{Z}_f^{-1}$ is calculated as

$$\boldsymbol{Z}_f^{-1} = \boldsymbol{Z}^T(C\mathbf{I} + \boldsymbol{Z}\boldsymbol{Z}^T)^{-1} \tag{6.2.9}$$

where $C\mathbf{I}$ adds the constant parameters to the diagonal of $\boldsymbol{Z}\boldsymbol{Z}^T$ according to the ridge regression theory for improving the stability of the network. However, based on the ex-

periments, the value of C is not affecting the final clustering performance in the CVAC algorithm.

Step 3. We update the coding matrix $\boldsymbol{Z}$ as

$$\boldsymbol{Z}_f = \mathbf{g}_s(\mathbf{a}_f.\boldsymbol{V} + \boldsymbol{b}_f) \tag{6.2.10}$$

where the $\mathbf{a}_f$ and $\boldsymbol{b}_f$ are calculated as $\mathbf{a}_f = (\mathbf{a}_k)^T$ and $\boldsymbol{b}_f = \boldsymbol{b}_k$.

Step 4. We iterate the second and third steps multiple times for adjusting the parameters of the coding matrix $\boldsymbol{Z}$. Based on the experiments, the clustering results are boosted upto four iterations. However, increasing the number of iterations over four times is not boosting the clustering performances. Thus, we update the parameters of the coding matrix in four iterations since the parameters are not improved in the further iteration steps.

The CVAC has few advantages and limitations in comparison with UVAC. We update the parameters of the encoding layer based on the weights of the decoding layer in the CVAC algorithm. In other words, the parameters of the coding matrix are not calculated and they are achieved from the decoding layer. This procedure aims CVAC to generate the coding matrix much faster than the UVAC algorithm. We calculate the parameters in each layer of UVAC independently. Aside from that, the UVAC contains general nodes including sub-network nodes in the coding layer which makes more accurate coding matrices for unconstrained video clustering. However, it shows that CVAC can produce more reliable coding matrices for constrained video clustering problems. The efficiency of generated coding matrices by CVAC and UVAC algorithms is evaluated by performing clustering experiments on Weizmann and Keck Gesture datasets as stated in Section IV.

3) Action Clustering based on Provided Coding Matrix $\boldsymbol{Z}$. Previous subspace video clustering methods [122, 83, 82] generate the affinity graph as $\boldsymbol{G} = \left(|\boldsymbol{Z}| + \left|\boldsymbol{Z}^T\right|\right)/2$ where $\boldsymbol{Z}$ contains the video features. The features of the sequential video frames are highly correlated to each other [123] since they are extracted from time-series domain. Due to the underlying relationships of within-cluster samples, the obtained affinity graph $\boldsymbol{G}$ does not well present consequent blocks of actions. Thus, we utilize the proposed similarity measurement in [81] which employs the advantages of the correlation among within-cluster

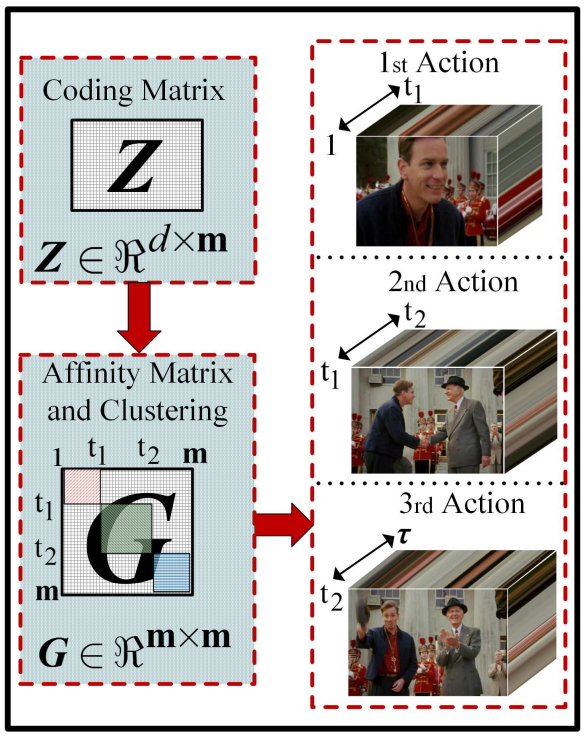


Figure 6.2.4: The affinity graph ($\boldsymbol{G}$) which is derived from the generated coding matrix ($\boldsymbol{Z}$).

samples. Additionally, we utilize the proposed coding matrices from constrained and unconstrained videos instead of using raw features for affinity graph construction. The affinity graph $\boldsymbol{G}$ is produced by

$$\boldsymbol{G}_{(i,j)} = \frac{z_i^T z_j}{\|z_i\|_2 \|z_j\|_2} \tag{6.2.11}$$

where $\boldsymbol{G} \in \Re^{\mathbf{m}\times\mathbf{m}}$ includes dense squares on the diagonal of the affinity matrix. The variable $\mathbf{m}$ refers to the index of the frames in a given video. Each dense square on the diagonal of affinity graph demonstrates an individual action in a given video as depicted in Figures 6.2.4 and 6.3.1.

The conventional video clustering methods try to segment the affinity matrix using the Normalized Cuts (NC) algorithm [124, 122, 83, 82, 81]. However, we cannot confidently estimate the number of clusters in a given video by employing the NC algorithm. From the other side, the generated affinity graphs contain some noises when the coding matrix comes from an unconstrained video. In this scenario, the NC is not capable of clustering the data into plausible segments even though we properly estimate the number of clusters

in a given video. To address this problem, we propose a simple yet robust methodology to deal with clustering a video using the generated affinity graph $\boldsymbol{G}$.

We analyse the diagonal of the affinity graph for clustering the data into plausible non-overlapping actions. We apply the kernel $\boldsymbol{K}$ on the diagonal of $\boldsymbol{G}$ and then extract the values of the diagonal of $\boldsymbol{G}$ for clustering stage.

$$\boldsymbol{K} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{6.2.12}$$

The kernel $\boldsymbol{K}$ aims to extract the edges with enough sensitivity on the diagonal of affinity graph. The 1-dimensional vector from the diagonal of $\boldsymbol{G}$ is then employed for clustering the consequent video frames into plausible actions. The available high peaks in the achieved 1-dimensional signal indicate the frame borders between different actions of a video. It is worth pointing out that a given point of the obtained signal is a high-peak value if the two neighbouring values contains lower amplitudes.

The calculated 1-dimensional signal from the diagonal of affinity graph may contain noises due to the complexity of motion patterns and coding matrix. We discover the most important patterns to detect the border frames by smoothing the signal using the Gaussian blurring filter. We consider the length of the Gaussian filter vector as 40 since each individual action needs atleast two seconds (40 frames) to be implemented in the employed datasets. Thus, the employed Gaussian filter is capable of discriminating different motions by obtaining the most crucial high peaks in the 1-D signal. Finally, we generate the distance clustering matrix (DCM) based on the indexes of frame borders. We represent the features of temporal blocks which are obtained based on frame borders. The features are represented using the improved rank pooling method [117]. The represented features of consequent blocks are compared using the product of cosine and standard Euclidean distance metrics. The mixing of two similarity distance metrics aims to confidently compare the represented features from different points of views. The similarity scores of the represented features of temporal blocks generate the distance clustering matrix (DCM) $\in \Re^{B \times B}$

where B is the number of temporal blocks from a given video based on the known high-peaks in 1-D signal.

We remove the redundant consequent frame borders in few iterations until reaching the convergence of DCM. It is worth pointing out that discriminated consequent video blocks and similar blocks which are not located next to each other are remained and fed to KAP framework for key action detection and recognition. In other words, if the same action blocks are happened by a gap of different actions among them, we keep the identical actions since we want to check the different sequential blocks in real-world applications.

The reason of having similar consequent action blocks in DCM is that some actions contain repeatable motions. For instance, in the jumping action from Weizmann dataset, we have several jumping in a given video sample. Thus, the smoothed 1-D signal from diagonal of affinity graph may contain several high peaks based on repetition of the same motion. We target to remove the redundant frame borders and extract the features of the entire video block instead of performing action detection and recognition for separated identical video blocks. All the frame borders based on similar neighbourhood temporal blocks are removed. Then, the represented features of each block are fed to the first classifier of KAP for key action detection. Later, if a temporal block is detected as the key action, we feed its features to the second classifier for the action recognition. In this scenario, we do not need any prior knowledge about the number of action clusters in a given video. Aside from that, we can confidently define the clusters from constrained and unconstrained videos for video surveillance and video retrieval applications.

6.3 Experimental Results

The KAP along with the multilayer video action clustering, based on CVAC and UVAC algorithms, are evaluated on Hollywood2 and URADL datasets. The employed datasets include long-shot videos compare to other available presegmented datasets such as UCF101 [66] and HMDB51 [67]. Additionally, the CVAC and UVAC frameworks are evaluated on Weizmann [2] and Keck Gesture datasets to show their robustness for video clustering in different conditions. This section thoroughly presents the datasets, experiments of action

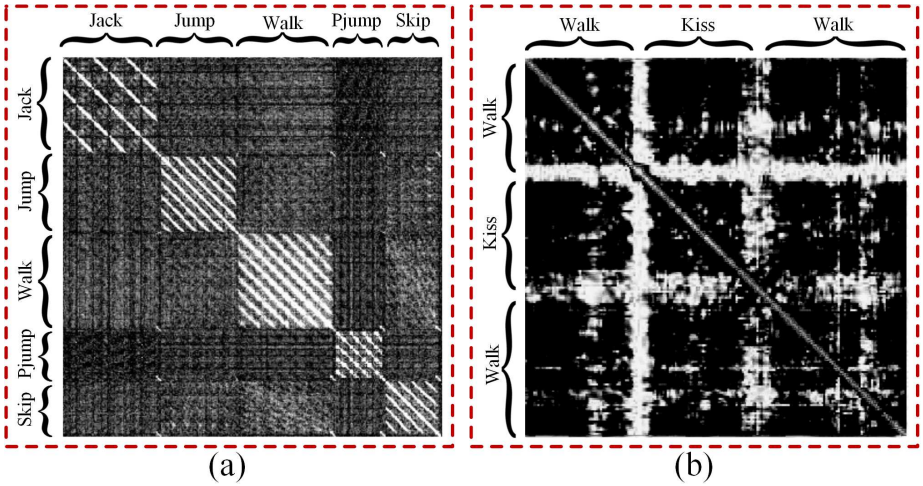


Figure 6.3.1: (a) The affinity graph $\boldsymbol{G}$ based on combination of 5 actions from Weizmann dataset. (b) The affinity graph of the sample from Hollywood2 dataset containing 3 discriminated actions.

detection and recognition performances, and evaluation of video clustering algorithms.

6.3.1 Datasets

Majority of the available action recognition datasets contain the fine-grained videos including a sole action in a couple of seconds. The proposed KAP approach is capable of detecting the key action among multiple actions in a given video. For the evaluation of KAP performance, we selected the Hollywood2 and URADL datasets which include complex contents and multiple actions in the single-shot video samples. Additionally, we evaluate the proposed CVAC and UVAC frameworks on constrained (Weizmann) and unconstrained (Keck Gesture) video datasets.

6.3.2 Evaluation of Video Action Clustering Algorithms

In this section, we evaluate the accuracy of the proposed multi-layer video action clustering framework on Weizmann and Keck Gesture datasets. We target to boost the clustering performances on constrained and unconstrained videos using the CVAC and UVAC algorithms. Later, we adopt the CVAC and UVAC algorithms in the KAP framework to perceive the key action in a given video.

Each video sample of the Weizmann dataset contains a unique action. Thus, we randomly concatenate five videos of different classes into a more extended video sequence to evaluate the clustering approaches. This procedure is repeated for 10 times. For evaluation of the clustering approaches on Keck Gesture dataset, we create the long-shot videos by concatenating of 14 video samples of different classes. It is worth pointing out that Keck Gesture dataset contains the videos which are captured with stable and dynamic cameras. We only use the videos which are captured with dynamic cameras since we want to evaluate the unconstrained video action clustering using the Keck Gesture dataset. Additionally, we employ the Hollywood2 and URADL to evaluate the KAP framework. Since the KAP performance is highly dependant on the video action clustering section, we visually evaluate the clustering performance on some samples of Hollywood2 dataset.

The actions of the merged constrained and unconstrained videos can be easily discriminated in the obtained affinity graph $\boldsymbol{G}$ as depicted in Figure 6.3.1. The parallel lines in left or right sides of the diagonal in affinity graph $\boldsymbol{G}$ show the repetition of each action in a sequence. For instance, in Figure 6.3.1(a), the first square indicate the Jack action performed by Daria in the Weizmann dataset which includes 3 repeated Jack actions. This condition is valid for all the squares in the affinity graph $\boldsymbol{G}$. It should be noted that Figure 6.3.1(a) and (b) show the affinity graphs from 5 actions of Wiezmann (including jack, jump, walk, and skip) and a video from Hollywood2 dataset (including walk, kiss, and walk) respectively. Since the camera and background are stable in the Weizmann dataset, the produced affinity graph has less noise and the squares on the diagonal can be easily discriminated to localize different actions. For the video from Hollywood2 dataset, the obtained affinity graph consists of more noises since the camera and background are dynamic while capturing the video. We propose the CVAC and UVAC for constrained and unconstrained video clustering tasks. Thus, regardless of the complexity of a given video, we produce temporal clusters of different actions. The dense block diagonals in the affinity graphs imply that with-in cluster structures are well analysed since our approach is very flexible to control the sequential neighbours.

For the evaluation of CVAC and UVAC algorithms, we manually label the ground truth of the available temporal clusters in the testing video samples. The labeling is performed on

the individual frames of the testing videos. Then, we follow the pair-counting measurement proposed in [125] to measure the accuracy of the clustering approaches. The employed pair-counting measurement includes two major variables (p_1, p_2). The p_1 refers to the percentage of pairs where both the ground truth and clustered frames are assigned to the same cluster. The p_2 refers to the percentage of pairs where the clustered and ground truth frames are assigned to non-identical clusters considering all pairs of different class video frames. Furthermore, the average of p_1 and p_2 indicates the clustering performance.

Table 6.3.1 states the accuracy of action clustering performances on the Weizmann and Keck Gesture datasets. The proposed video clustering approaches outperform the state-of-the-art methodologies. Based on the experiments, the CVAC marginally outperforms the UVAC algorithm for constrained video clustering, but its processing speed is much faster compare to UVAC. Thus, we conclude that the CVAC algorithm is capable of clustering the constrained videos with a high level of confidence. Furthermore, we find that CVAC is appropriate for the applications where the camera and background are stable such as video surveillance. And the UVAC is suitable for the applications with the dynamic camera and cluttered background such as video retrieval frameworks.

Optimizing the Parameters of CVAC and UVAC algorithms: The raw feature vectors along with their created labels are fed to the multi-layer video action clustering (ML-VAC) framework for clustering a given video into plausible actions. The CVAC and UVAC methods contain two important parameters. The first one is the regularization parameter and the second parameter refers to the dimension of the coding matrices.

We evaluate the regularization parameter C on the video clustering results. The regularization parameter C in both CVAC and UVAC is selected from $C \in \{2^{-4}, ..., 2^{8}\}$. Figure 6.3.2 shows the results of video action clustering by employing a set of parameter C on Weizmann and Keck Gesture datasets. The ML-VAC performs well with the optimal $C = 2^3$ based on obtained experimental results. Additionally, we evaluate the KAP performance on Hollywood2 and URADL datasets where the video action clustering is the crucial preprocessing step. We employ the set of regularization parameters $C \in \{2^{-4}, ..., 2^{8}\}$ for KAP evaluation. Based on the experiments, we achieve the best action recognition performance using $C = 2^3$ in the CVAC and UVAC algorithms.

Table 6.3.1: Comparison of proposed ML-VAC to the state-of-the-arts

Dataset	Method	Accuracy
Weizmann	SSC [82]	38.8%±3.2%
	LSR [122]	40.1%±2.9%
	OSC [80]	65.8%±3.3%
	ML-VAC based on CVAC	**78.9%±1.6%**
	ML-VAC based on UVAC	78.6%±1.9%
Keck Gesture	SSC [82]	26.8%±2.4%
	LSR [122]	38.2%±2.1%
	OSC [80]	41.9%±2.3%
	ML-VAC based on CVAC	58.2%±1.8%
	ML-VAC based on UVAC	**60.9%±1.3%**

Furthermore, we evaluate the set of dimensions for generation of coding matrices via CVAC and UVAC algorithms. Figure 6.3.3 shows the obtained results of video clustering and key action perception based on different coding matrix dimensions. The UVAC algorithm outputs the best results while setting the dimension of coding matrices as 600 for Keck Gesture and Hollywood2 datasets. For unconstrained video analysis, the dimension may be different based on the length and complexity of the video frames. It should be noted that the concatenated testing videos from the Keck Gesture dataset contain 14 actions in few minutes. Thus, we may loose some informative features while creating the coding matrix with a low dimension on Keck Gesture dataset. Finally, we confidently conclude that the coding matrix with dimension of 400 outputs the best results for constrained video analysis.

6.3.3 Evaluation of KAP Performance

The Hollywood2 and URADL datasets are employed to evaluate the overall action recognition performance using KAP along with the CVAC and UVAC algorithms. The Hollywood2 dataset contains unconstrained videos. Some of the video samples of the mentioned dataset include several actions in few seconds. However, each video sample has a sole label even though several actions are occurred in a given video. The KAP framework includes two classifiers to detect and recognize the key action among a set of temporal clusters.

The first KAP classifier is trained by positive key actions which are labelled manually

from Hollywood2 and URADL datasets. For the negative samples, we randomly collect the same number of videos from other datasets such as UCF101 [66] and HMDB51 [67]. The sets of negative and positive samples include different categories and the actions of negative samples are not existed in the positive samples. We obtain reliable key action detection results based on the first KAP classifier. We achieve 95.2% and 100.0% for the Hollywood2 and URADL datasets respectively. Later, the second classifier of KAP recognizes the key actions. The same settings for train-test splits of Hollywood2 [89] and URADL [84] datasets are used for training and testing the second classifier in KAP framework. We use the linear SVM classifier in both classification stages of KAP framework. We follow the same SVM parameters as in [10, 68, 117] since we compare the obtained KAP results with [10, 68, 117].

Effect of the Codebook Dimension on Video Clustering and Key Action Perception: The feature encoding of the temporal blocks from a given video has an impressive effect on the ML-VAC and KAP frameworks. During the experiments, we provide a set of codebook dimensions for training of the Gaussian mixture model in Fisher vector encoding methodology. The best results are obtained by 32, 64, 24, and 48 codebook sizes for URADL, Hollywood2, Weizmann, and Keck Gesture datasets respectively. It should be noted that the obtained results for the URADL and Hollywood2 datasets are achieved based on the key action perception along with the video clustering algorithms. However, the demonstrated results for the Weizmann and Keck Gesture datasets show the accuracy of the proposed ML-VAC based on CVAC and UVAC algorithms respectively.

We believe that the optimized codebook dimension has a direct relation to the number of available classes and the complexity of the video features in each dataset. For instance, the URADL and Weizmann datasets contain 10 classes in multiple constrained videos. From the other side, Hollywood2 and Keck Gesture Datasets include 12 and 14 classes in several unconstrained videos. Thus, we conclude that the higher codebook dimension aims to boost the key action perception in complex and unconstrained videos.

Evaluation of different similarity scores on KAP performance: The ML-VAC algorithm aims to figure out the starting and ending frames for each action by defining the high peaks on the 1-dimensional signal. However, several border frames may be defined

in case of having the same repeated sequential actions in a video. Figure 6.3.4 shows the 1-dimensional signal from a video sample of Hollywood2 dataset. The employed video sample includes 235 video frames as shown in Figure 6.3.4. The 1-dimensional signal includes seven high peaks. But, only three actions have been clustered after comparing the consequent blocks by our distance metric. As shown in Figure 6.3.4, the temporal block between the first and second high peaks refers to the walking action. The temporal cluster between the second and forth high peaks refers to the action of putting dishes on the table. Since two dishes have been placed on the table based on two different motions, we got two high peaks in the signal. The sitting action is clustered based on the fourth to seventh high peaks. The reason of having several high peaks for the sitting action is that the video is captured from different points of views while the girl is sitting on the chair. So, the ML-VAC defines several high peaks based on different views of the camera. However, the temporal blocks between the high peaks are compared and if the consequent blocks have a similarity score below the threshold, we merge them into a unique temporal block. All in all, we obtain three clusters of different actions from the given video of Hollywood2 dataset. In the Figure 6.3.4, ρ refers to the values from the smoothed diagonal of the affinity graph.

Table 6.3.2 states the set of similarity scores for comparing the consequent temporal blocks while clustering a given video into plausible actions. The experiments show that the thresholding score of 24 results the best KAP performance for the Hollywood2 and URADL datasets. In other words, if the similarity score of two video blocks is less than 24, we merge the two consequent blocks together as the same action. Alternatively, we keep the video blocks separated as two discriminated actions.

Effect of Video Length on Key Action Perception: The length of videos has a direct effect on KAP performance. As the video length is expanded, the number of temporal clusters is increased. Consequently, the feature set from a given video may contain more redundant data and noises which makes the action classification more challenging. in this scenario, the proposed ML-VAC algorithm must cluster a given long-shot video into non-overlapping actions. Then, the KAP framework analyses the produced temporal clusters for key action perception.

Based on the extensive experiments on Hollywood2 and URADL datasets, we boost the

Table 6.3.2: KAP evaluation based on different similarity scores for merging the consequent identical temporal blocks.

Similarity Score	6	8	10	12	14	16	18	20	22	**24**	26	28	30	32	34
KAP performance on Hollywood2 (MAP)	47.8	57.4	62.9	65.6	67.9	67.9	67.9	69.6	69.6	**72.0**	69.6	68.7	68.7	67.6	64.8
KAP performance on URADL (Accuracy)	96.0	96.0	97.5	97.5	97.5	98.8	98.8	100.0	100.0	**100.0**	100.0	97.5	96.0	95.0	95.0

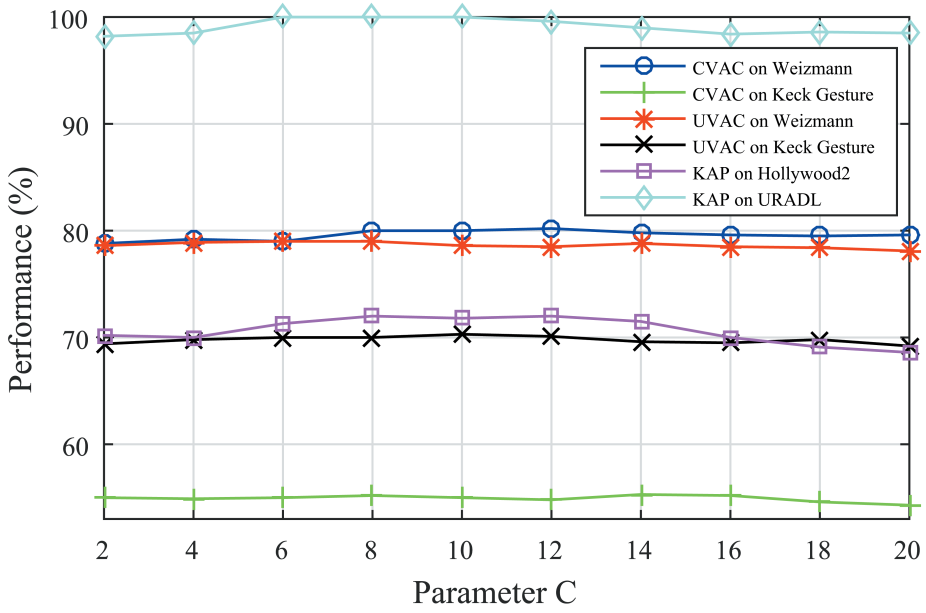


Figure 6.3.2: Evaluation of parameter C for video action clustering and key action perception.

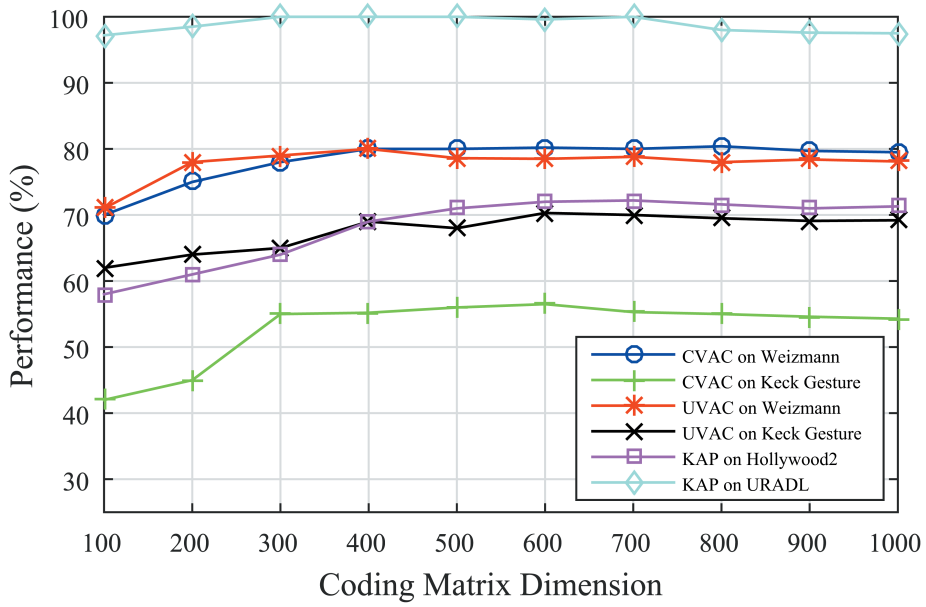


Figure 6.3.3: Effect of different coding matrix dimensions on video action clustering and key action perception algorithms.

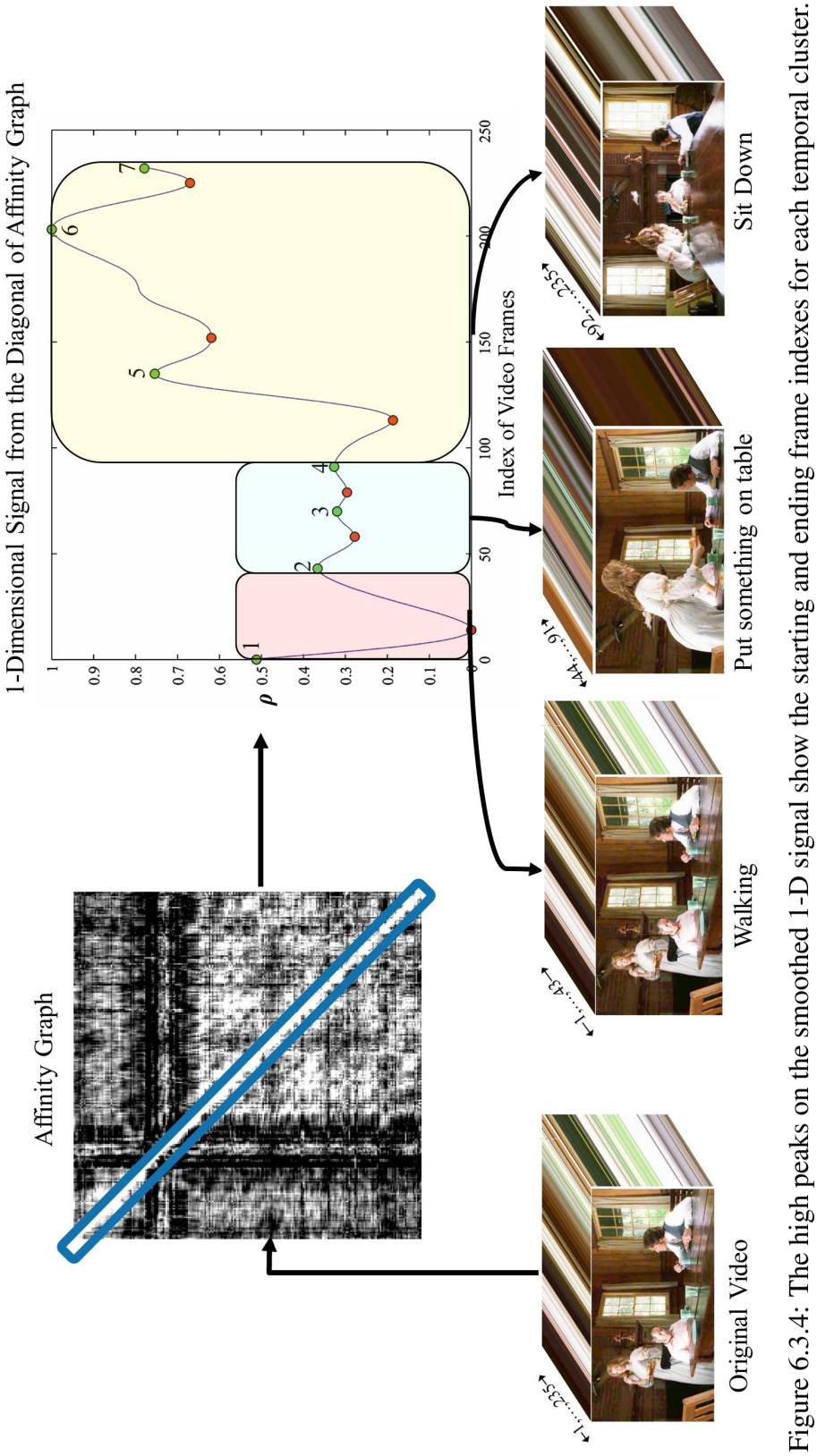


Figure 6.3.4: The high peaks on the smoothed 1-D signal show the starting and ending frame indexes for each temporal cluster.

action recognition performance while the length of videos is increased upto 200 frames. The achieved results demonstrate that KAP framework along with ML-VAC algorithm is capable of detecting and recognizing the key action in the long-shot videos of Hollywood2 dataset (see Figure 6.3.5). We found that video samples with above 200 frames are mostly from routine actions such as drive a car, fight, and walk. In other words, the same action is performed and repeated in a long-shot video. In this cases, the action recognition is not boosted since the overall feature sets from the entire video contain the structured information from a unique action. However, KAP is very powerful to perceive the key action in the video samples where few actions are occurred in one video shot which contain below 200 frames.

Additionally, we evaluate the effect of the video length on CVAC and UVAC algorithms. The experiments show that the accuracy of video clustering algorithms is decreased if the video length is highly increased. We believe that this is due to the property of the encoded features from individual frames at the first step of our framework. As mentioned before, we encode the features of the single frames from a given video. Then, we generate the coding matrix based on the encoded features. If the number of frames is huge, the CVAC and UVAC algorithms cannot provide informative coding matrices for the video clustering. In other words, the generated 1-dimensional signal from the affinity graph contains a lot of noises which leads to a weak clustering stage in a long-shot video. Thus, we recommend to shorten the videos to maximum of 200 frames before extraction and encoding of features. In this case, the CVAC and UVAC algorithms are capable of producing informative coding matrices regardless of complexity of a given video.

6.3.4 Discussion

We compare the obtained action recognition results with state-of-the-arts as stated in Table 6.3.3. Since we employed the improved dense trajectories (iDT) [10] at the first steps of UVAC and CVAC algorithms, we compare the KAP results with iDT framework. Additionally, we compare the results with the stacked fisher vectors (SFV) [68] since SFV combines the features of different temporal segments into a unique feature vector. Furthermore, we

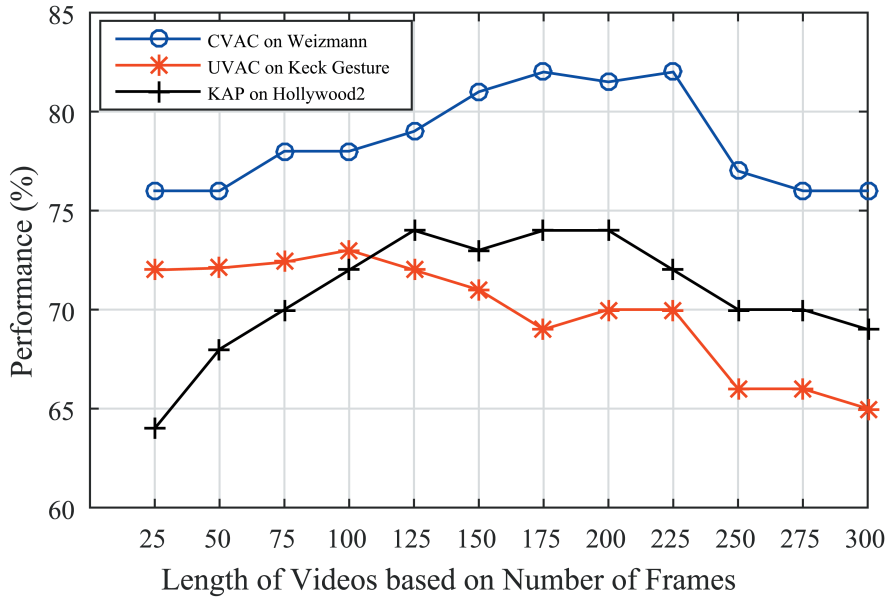


Figure 6.3.5: Effect of the video length on the clustering approaches and KAP performance.

compare the KAP results with improved rank pooling (IRP)[117] since we employ the IRP in the KAP framework to detect and recognize actions in constrained and unconstrained videos. Furthermore, we compare the KAP performance with fuzzy topic model [95], genetic programming [113], STAP framework [126], fusion of global dynamics and local appearance [127], explicit modelling [100], and velocity histories [84].

Based on the extensive experiments, we obtained reliable results on Hollywood2 (72.0%) and URADL (100.0 %) datasets. As stated in Table 6.3.3, the KAP is capable of boosting the action recognition performances compare to other state-of-the-arts since the redundant video clusters are removed and the action is recognized by applying the IRP on the key action cluster. The achieved results demonstrate that a robust video action clustering framework impressively affects the action detection and recognition performance. Aside from that, for the real-world applications, such as video surveillance and retrieval problems, the video frames consist of a complex motion structure containing multiple actions. So, the proposed clustering approaches can be employed to cluster a given video into multiple plausible non-overlapping actions and then perform the action detection and recognition tasks efficiently.

Compared to the state-of-the-arts, the most reliable results are achieved on Answer

Table 6.3.3: Comparison of KAP results to the state-of-the-arts

Dataset	Method	Recog. Rate
Hollywood2	Improved Dense Trajectories [10]	64.8%
	Stacked Fisher Vectors [68]	67.6%
	Improved Rank Pooling [117]	70.6%
	Fuzzy Topic Model [95]	47.9%
	Genetic Programming Model [113]	46.8%
	STAP Framework [126]	62.5%
	Global and Local Features[127]	58.61%
	Proposed KAP	**72.0%**
URADL	Improved Dense Trajectories [10]	93.3%
	Stacked Fisher Vectors [68]	95.6%
	Improved Rank Pooling [117]	99.6%
	Explicit Modeling Approach[100]	92.0%
	Velocity History (full method)[84]	89.0%
	Velocity History (point tracks)[84]	67.0%
	Proposed KAP	**100.0%**

phone, Handshake and Kiss actions from Hollywood2 dataset since these video samples usually contain redundant motions and non-relevant actions. Since the KAP is capable of detecting the key action, we could improve the results of the stated actions by at least eight percentages. It is worth pointing out that the KAP evaluation is performed on Hollywood2 and URADL datasets since they roughly contain long shot videos including several actions in a video.

6.4 Summary

In this chapter, we propose the key action perception (KAP) to detect and recognize the key actions in constrained and unconstrained videos. As the crucial preprocessing step, a given video is supposed to be clustered into plausible actions for KAP implementation. Thus, additionally, we propose the multi-layer video action clustering for constrained and unconstrained videos. The proposed clustering approaches provide the sequential temporal clusters via analysis of time-series video data. Extensive experiments demonstrate the robustness of the proposed KAP and clustering frameworks which achieve state-of-the-art results on benchmark datasets.

CHAPTER 7

Conclusion and Perspectives

This chapter presents a summary of the implemented efforts and the conclusions inspired by them. The local features are very popular in the action recognition domain due to their superior performance on constrained and unconstrained videos. In this dissertation, action recognition systems through local features have been extensively studied and several advances have been proposed. Chapter 3 presents the evaluation of ensemble learning systems on simple and complex action recognition performances. Chapter 4 proposes a robust hybrid classifier to recognize human activities in constrained videos. Chapter 5 presents a hierarchical framework to represent a given unconstrained video including low-level and higher-level information which are achieved on the top of low-level features. Chapter 6 explores the key action perception along with the constrained and unconstrained video action clustering systems. Finally, the possible future research directions that could be pursued as the extensions of these efforts are pointed out in this chapter.

7.1 Contributions and Limitations

The major contributions of this dissertation are divided into three groups: 1) Enhancing the classification module as ensemble learning and hybrid classifier. 2) Improving the video representation module by extracting new higher-level information and fusion of features. 3) Proposing a robust video action clustering approach followed by the key action perception. The following subsections describe the detailed major and minor contributions of this dissertation.

7.1.1 Ensemble Learning for Action Recognition

The third chapter of this dissertation evaluates the effect of ensemble learning in action recognition domain. It is shown that fusion of appearance-based and motion features aims to boost the action recognition performance in constrained and unconstrained videos. However, concatenation of individual features generates a huge fused vector which makes the classification very challenging. The employed ensemble approach trains several classifiers for individual features. Then, the scores of single classifiers are combined to make the final decision about the class of a given video. We evaluate the Dempster-Shafer and algebraic fusion strategies to combine the classifiers' scores.

The experimental results demonstrate that score-level fusion of single classifiers improves the action recognition performance in constrained and unconstrained videos. We can save memory while classifying a given video using the employed ensemble approach since the features from different perspectives are combined as a late fusion method.

This approach has the limitation of slow training time for several classifiers. In the early fusion approach, in case of having enough memory, we can train a single classifier even though its accuracy is marginally lower than the ensemble approach. However, in the ensemble method, we have to spend much more time to train individual classifiers using a variety of separated feature sets. Aside from that, we may need to adjust the parameters of individual classifiers separately since the features are usually from different perspectives.

7.1.2 Hybrid Classifier

The recognition of similar actions, such as walking, running and jogging is considered as a challenging task in the action recognition domain. Generally, the classifiers cannot confidently assign the proper label to a given sample while having similar classes in a dataset. To address this problem, the fourth chapter of this dissertation makes the following contributions for action recognition:

1) The hybrid classifier is proposed to efficiently assign the labels to similar actions. The hybrid classifier aims to check the confidence of a produced label. In case of producing a non-confidence label, the hybrid classifier compresses the features and

recognizes the action with polynomial or sigmoid activation functions.

2) Additionally, we evaluate the effect of motion saliency map detection on motion features for action recognition. We employ the 3D-DWT to extract the motion saliency maps of constrained videos.

The proposed approach obtains impressive action recognition results on constrained video datasets. However, this approach cannot improve the results in unconstrained videos. Thus, we conclude that the hybrid classifier can be employed for the video surveillance applications where the camera and background are stable.

7.1.3 Hierarchical Feature Representation

The fifth chapter introduces pooled-feature representation (PFR) which is derived from a double phase encoding framework (DPE) to represent an unconstrained video using low-level and higher-level information. In particular, the fifth chapter of this dissertation proposes the following contributions:

1) Enhancing the general rank pooling strategy by removing the redundant features from identical frames of a given video.

2) Employing the cosine and correlation distance metrics for detecting and removing the redundant features using the proposed consequent and protracted checking methods. Furthermore, we define the optimized threshold via introduced double thresholding scheme for removing the identical features in a short or long shot video.

3) Proposing the DPE framework to provide useful information for complex video analysis based on the assumption that a complex video is composed of a sequence of simple actions. The early fusion of individual represented vectors (IRVs), at the first phase of DPE, outperforms the traditional encoded vectors obtained from features of entire video.

4) Introducing VPV at the second phase of DPE to represent higher-level information for unconstrained video analysis. The VPV is achieved by encoding the compressed

version of IRVs which are obtained at the first phase of DPE. The VPV consists of higher-level information to represent a video since the IRVs contain low-level information of individual blocks of a video. In other words, encoding of IRVs aims to obtain higher-level information about the overall motions in a video.

5) Proposing the PFR with early and late fusion strategies to leverage the motion and appearance-based information from a given video. It is worth pointing out that PFR with early fusion (PFR-EF) provides the most significant features with least dimension and highest efficiency for complex action recognition.

6) Adopting the LSN classifier in the action recognition domain and compare its training speed and accuracy with traditional extreme learning machine (ELM) and support vector machine (SVM) over six challenging datasets.

The PFR-EF is appropriate for representing of constrained and unconstrained videos. However, it may not enhance the action recognition performance on the actions with routine and specific motions in short shot videos. This framework achieves impressive results on the long-shot videos including multiple simple motions such as the videos in URADL and Hollywood2 dataset.

7.1.4 Key Action Perception

The conventional action recognition algorithms have been developed and tested on pre-segmented video datasets which contain a sole action in few seconds. However, the videos contain long and complicated temporal contents in real-world scenarios. We hypothesize that clustering a captured video into plausible actions is a crucial pre-processing step to action recognition task in real-world applications such as video surveillance and video retrieval. In particular, the sixth chapter of this dissertation proposes the following contributions:

1) We propose a multi-layer clustering approach to temporally segment a video into plausible actions. The proposed approach consists of two different unsupervised learning paradigms for constrained and unconstrained video action clustering.

2) We propose and evaluate the unconstrained video action clustering (UVAC) and constrained video action clustering (CVAC) methods to produce coding matrices for analysis of the motion features in a video. It should be noted that the proposed clustering approach using the CVAC and UVAC algorithms is capable of segmenting a video without having any prior knowledge about the number of clusters.

3) The methodology of key action perception (KAP) is proposed and evaluated to detect the key action among multiple plausible clusters and recognize the detected key action. The KAP consists of two classifiers to detect and recognize the key actions in constrained and unconstrained videos.

4) We have manually labelled the key and noise temporal clusters in the Hollywood2 and URADL datasets to train the first classifier of the KAP framework. The second classifier of the KAP framework is trained using the original labels of the employed datasets. The Hollywood2 is utilized as an unconstrained video dataset which consists of complex video streams with sophisticated temporal contents. The URADL contains complex actions in constrained videos.

The KAP is capable of perceiving the key action in short or long shot videos. However, the length of videos has a direct effect on the proposed video action clustering. The extensive experiments demonstrate that the proposed approaches work well on the videos with the maximum of 200 frames. Therefore, the input videos must be shortened to 200 frames before employing the proposed clustering approaches.

7.1.5 General Summary

This dissertation has been targeted towards enhancing the simple and complex video action recognition performances. The effort was spent to encompass both constrained and unconstrained video analysis. In the real-world applications, videos consist of long and complex contents. The conventional algorithms fail to recognize actions in long-shot videos since each sample may contain several actions. To address this problem, we propose a multi-layer clustering approach to temporally segment the videos into plausible actions. Then,

the proposed key action perception and hierarchical feature representation along with the hybrid classifier can be employed to detect and recognize key actions among multiple plausible temporal clusters in a video. Next section provides some directions for future possible researches to enhance the proposed frameworks and modify the action recognition performance.

7.2 Scope for Future Work

We have probable future directions for carrying forward the research and findings of our contributions. This section thoroughly presents the future works for boosting and evaluating the developed action recognition frameworks as follows:

1) The double phase encoding approach can be implemented by considering the spatial and temporal windows. We only utilize the temporal windows to represent a given video. However, considering the spatial windows aims to robustly represent complex videos. This is due to having actions in different spatial locations in real-world scenarios. Selection of appropriate spatial windows is an open research problem and it is directly related to localization of an action in the spatial domain of video frames.

2) The achieved PFR-EFs are independent of the type of video features. We utilize the improved dense trajectories as the initial step in our algorithm. We target to test the proposed framework using more advanced video-based features.

3) It should be noted that we utilize SVM to produce ranking vectors for video representation in IRP framework. In the future, we target to perform more researches on ranking machines and employ convolutional neural networks and extreme learning machine to represent a given video based on the evolution of video frames.

4) The UVAC and CVAC algorithms are proposed in the ML-VAC framework to cluster unconstrained and constrained videos. We can select the appropriate algorithm based on the video analysis application. For instance, we utilize CVAC in the video surveillance applications where the camera and background are stable. From the other side,

we can use UVAC for video retrieval applications where the camera and background could be in the dynamic format. Based on the extensive experiments, it is shown that the video length has a direct effect on the proposed clustering approach.

We target to focus on developing a more robust algorithm to cluster the actions regardless of the complexity, length, and content of a given video. Different feature mapping frameworks such as deeper auto encoders can be employed to make advancements in the video clustering domain.

5) We target to evaluate the proposed methodologies on larger datasets such as ActivityNet and YouTube-8M. These datasets have been updated recently and contain a huge number of samples for individual action classes.

6) Currently, there is a lack of video datasets including constrained videos and complex actions. We plan to create our video surveillance dataset containing the long-shot videos and perform the KAP framework along with the ML-VAC for video analysis.

References

[1] Aaron F. Bobick and James W. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on pattern analysis and machine intelligence*, 23(3):257–267, 2001.

[2] Lena Gorelick, Moshe Blank, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2247–2253, 2007.

[3] Ivan Laptev and Tony Lindeberg. Space-time interest points. In *International Conference on Computer Vision*, pages 432–439, 2003.

[4] Geert Willems, Tinne Tuytelaars, and Luc Van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. *Computer Vision–ECCV*, pages 650–663, 2008.

[5] Heng Wang, Alexander Klser, Cordelia Schmid, and Liu Cheng-Lin. Action recognition by dense trajectories. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3169–3176, Jun. 2011.

[6] An Liu, Yu Su, Wei Nie, and Mohan Kankanhalli. Hierarchical clustering multi-task learning for joint human action grouping and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(1):102–114, 2017.

[7] Quan Chen and Yu Zhang. Sequential segment networks for action recognition. *IEEE Signal Processing Letters*, 24(5):712–716, 2017.

[8] Yang Xian, Xuejian Rong, Xiaodong Yang, and Yingli Tian. Evaluation of low-level features for real-world surveillance event detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(3):624–634, 2017.

[9] Kang Li, Sheng Li, Sangmin Oh, and Yun Fu. Videography-based unconstrained video analysis. *IEEE Transactions on Image Processing*, 26(5):2261–2273, 2017.

[10] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *IEEE International Conference on Computer Vision*, pages 3551–3558, 2013.

[11] Samitha Herath, Mehrtash Harandi, and Fatih Porikli. Going deeper into action recognition: A survey. *Image and Vision Computing*, 60:4–21, 2017.

[12] Tangquan Qi, Yong Xu, Yuhui Quan, Yaodong Wang, and Haibin Ling. Image-based action recognition using hint-enhanced deep neural networks. *Neurocomputing*, 267:475–488, 2017.

[13] Jian Dong, Changyin Sun, and Wankou Yang. A supervised dictionary learning and discriminative weighting model for action recognition. *Neurocomputing*, 158:246–256, 2015.

[14] Haojie Li, Lijuan Liu, Fuming Sun, Yu Bao, and Chenxin Liu. Multi-level feature representations for video semantic concept detection. *Neurocomputing*, 172:64–70, 2016.

[15] Yong Wang and Shiqiang Hu. Exploiting high level feature for dynamic textures recognition. *Neurocomputing*, 154:217–224, 2015.

[16] Guoliang Lu and Mineichi Kudo. Learning action patterns in difference images for efficient action recognition. *Neurocomputing*, 123:328–336, 2014.

[17] Haibin Yan. Discriminative sparse projections for activity-based person recognition. *Neurocomputing*, 208:183–192, 2016.

[18] Zengmin Xu, Ruimin Hu, Jun Chen, Chen Chen, Huafeng Chen, Hongyang Li, and Qingquan Sun. Action recognition by saliency-based dense sampling. *Neurocomputing*, 236:82–92, 2017.

[19] Fang Liu, Xiangmin Xu, Shuoyang Qiu, Chunmei Qing, and Dacheng Tao. Simple to complex transfer learning for action recognition. *IEEE Transactions on Image Processing*, 25(2):949–960, 2016.

[20] Xiaojiang Peng, Limin Wang, Xingxing Wang, and Yu Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *Computer Vision and Image Understanding*, 150:109–125, 2016.

[21] Junji Yamato, Jun Ohya, and Kenichiro Ishii. Recognizing human action in time-sequential images using hidden markov model. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 379–385, 1992.

[22] Xingxing Wang, Limin Wang, and Yu Qiao. A comparative study of encoding, pooling and normalization methods for action recognition. In *Asian Conference on Computer Vision*, pages 572–585, 2012.

[23] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244, 1988.

[24] Piotr Dollár, Vincent Rabaud, Garrison Cottrell, and Serge Belongie. Behavior recognition via sparse spatio-temporal features. In *IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 65–72, 2005.

[25] Heng Wang, Muhammad Muneeb Ullah, Alexander Klaser, Ivan Laptev, and Cordelia Schmid. Evaluation of local spatio-temporal features for action recognition. In *British Machine Vision Conference*, pages 124–1, 2009.

[26] Alexander Klaser, Marcin Marszaek, and Cordelia Schmid. A spatio-temporal descriptor based on 3d-gradients. In *British Machine Vision Conference*, pages 1–8, Sep. 2008.

[27] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[28] Heng Wang, Muhammad Muneeb, Alexander Klser, Ivan Laptev, and Cordelia Schmid. Evaluation of local spatio-temporal features for action recognition. In *British Machine Vision Conference*, pages 124.1–124.11, 2009.

[29] Ivan Laptev and Tony Lindeberg. Local descriptors for spatio-temporal recognition. *Lecture notes in computer science*, 3667:91–103, 2006.

[30] Ivan Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2):107–123, Sep. 2005.

[31] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Dense trajectories and motion boundary descriptors for action recognition. *International Journal of Computer Vision*, 103(1):60–79, 2013.

[32] Mihir Jain, Herve Jegou, and Patrick Bouthemy. Better exploiting motion for better action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2555–2562, 2013.

[33] Lingqiao Liu, Lei Wang, and Xinwang LIU. In defense of soft-assignment coding. In *IEEE International Conference on Computer Vision*, pages 2486–2493, Nov. 2011.

[34] Florent Perronnin, Jorge Snchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *European Conference on Computer Vision*, Sep. 2010.

[35] Dan Oneata, Jacob Verbeek, and Cordiela Schmid. Action and event recognition with fisher vectors on a compact feature set. In *IEEE International Conference on Computer Vision*, pages 1817–1824, Dec. 2013.

[36] Xiaojiang Peng, Limin Wang, Zhuowei Cai, and Yu Qiao. Action and gesture temporal spotting with super vector representation. In *European Conference on Computer Vision*, pages 518–527, 2014.

[37] Mihir Jain, Jan C van Gemert, and Cees GM Snoek. What do 15,000 object categories tell us about classifying and localizing actions? In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 46–55, 2015.

[38] Herve Jegou, Matthijs Douze, Cordelia Schmid, and Patrick Perez. Aggregating local descriptors into a compact image representation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3304–3311, Jun. 2010.

[39] Jingen Liu, Benjamin Kuipers, and Silvio Savarese. Recognizing human actions by attributes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3337–3344, 2011.

[40] Jun Zhu, Baoyuan Wang, Xiaokang Yang, Wenjun Zhang, and Zhuowen Tu. Action recognition with actons. In *IEEE International Conference on Computer Vision*, pages 3559–3566, Dec. 2013.

[41] LiMin Wang, Yu Qiao, and Xiaoou Tang. Motionlets: Mid-level 3d parts for human motion recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2674–2681, 2013.

[42] Fei Yuana, Gui-Song Xiac, and Veronique Sahbib, Hichem Prinet. Mid-level features and spatio-temporal context for activity recognition. *Pattern Recognition*, 45(12):4182–4191, Dec. 2012.

[43] Michael Sapienza, Fabio Cuzzolin, and Philip H.S. Torr. Learning discriminative space?time action parts from weakly labelled videos. *International Journal of Computer Vision*, 110(1):30–47, Oct. 2014.

[44] Arpit Jain, Abhinav Gupta, Mikel Rodriguez, and Larry S. Davis. Representing videos using midlevel discriminative patches. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2571–2578, Jun. 2013.

[45] Du Tran and Lorenzo Torresani. Exmoves: Mid-level features for efficient action recognition and video analysis. *International Journal of Computer Vision*, 119(3):239–253, 2016.

[46] Limin Wang, Yu Qiao, and Xiaoou Tang. Mofap: A multi-level representation for action recognition. *International Journal of Computer Vision*, 119(3):254–271, 2016.

[47] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, 2016.

[48] Yifan Wang, Jie Song, Limin Wang, Luc Van Gool, and Otmar Hilliges. Two-stream sr-cnns for action recognition in videos. In *British Machine Vision Conference*, pages 1–12, Sep. 2016.

[49] 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 recognition. In *European Conference on Computer Vision*, pages 20–36, 2016.

[50] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Neural Information Processing Systems*, Dec. 2012.

[51] Christoph Feichtenhofer, Axel Pinz, and Richard P Wildes. Dynamically encoded actions based on spacetime saliency. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2755–2764, 2015.

[52] Basura Fernando, Efstratios Gavves, Jose M Oramas, Amir Ghodrati, and Tinne Tuytelaars. Modeling video evolution for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5378–5387, 2015.

[53] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques, 2007.

[54] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks and Learning Systems*, 13(2):415–425, Mar. 2002.

[55] Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, and Rui Zhang. Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 42(2):513–529, Apr. 2012.

[56] Rashid Minhas, Abdul Adeel Mohammed, and QM Jonathan Wu. Incremental learning in human action recognition based on snippets. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(11):1529–1541, 2012.

[57] Ho-Ta Lin, Tsorng-Juu Liang, and Shih-Ming Chen. Estimation of battery state of health using probabilistic neural network. *IEEE Transactions on Industrial Informatics*, 9(2):679–685, May. 2013.

[58] Rui Zhang, Yuan Lan, Huang Guang-Bin, and Zong-Ben Xu. Universal approximation of extreme learning machine with adaptive growth of hidden nodes. *IEEE Transactions on Industrial Informatics*, 23(2):365–371, Feb. 2012.

[59] QM Wu Yimin Yang. Extreme learning machine with subnetwork hidden nodes for regression and classification. *IEEE Transactions on Cybernetics*, 46(12):2885–2898, 2016.

[60] Cha Zhang and Yunqian Ma. *Ensemble machine learning: methods and applications*. Springer, 2012.

[61] Basura Fernando, Efstratios Gavves, José Oramas, Amir Ghodrati, and Tinne Tuytelaars. Rank pooling for action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):773–787, 2016.

[62] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4305–4314, 2015.

[63] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *International Conference on Neural Information Processing*, pages 568–576, 2014.

[64] Wangjiang Zhu, Jie Hu, Gang Sun, Xudong Cao, and Yu Qiao. A key volume mining deep framework for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1991–1999, 2016.

[65] Xuanhan Wang, Lianli Gao, Jingkuan Song, and Hengtao Shen. Beyond frame-level cnn: Saliency-aware 3-d cnn with lstm for video action recognition. *IEEE Signal Processing Letters*, 24(4):510–514, 2017.

[66] 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.

[67] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *IEEE International Conference on Computer Vision*, pages 2556–2563, 2011.

[68] Xiaojiang Peng, Changqing Zou, Yu Qiao, and Qiang Peng. Action recognition with stacked fisher vectors. In *European Conference on Computer Vision*, pages 581–595, 2014.

[69] Min-Ling Zhang and Lei Wu. Lift: Multi-label learning with label-specific features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(1):107–120, 2015.

[70] A Gorban, H Idrees, YG Jiang, A Roshan Zamir, I Laptev, M Shah, and R Sukthankar. Thumos challenge: Action recognition with a large number of classes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[71] Marcus Rohrbach, Sikandar Amin, Mykhaylo Andriluka, and Bernt Schiele. A database for fine grained activity detection of cooking activities. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1194–1201, 2012.

[72] Alexander Richard and Juergen Gall. Temporal action detection using a statistical language model. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3131–3140, 2016.

[73] Haoran Wang, Chunfeng Yuan, Weiming Hu, Haibin Ling, Wankou Yang, and Changyin Sun. Action recognition using nonnegative action component representation and sparse basis selection. *IEEE Transactions on Image Processing*, 23(2):570–581, 2014.

[74] Roeland De Geest, Efstratios Gavves, Amir Ghodrati, Zhenyang Li, Cees Snoek, and Tinne Tuytelaars. Online action detection. In *European Conference on Computer Vision*, pages 269–284, 2016.

[75] Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *IEEE International Conference on Computer Vision*, pages 2678–2687, 2016.

[76] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *IEEE International Conference on Computer Vision*, pages 1049–1058, 2016.

[77] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. Time-series clustering–a decade review. *Information Systems*, 53:16–38, 2015.

[78] Fei Wu, Yongli Hu, Junbin Gao, Yanfeng Sun, and Baocai Yin. Ordered subspace clustering with block-diagonal priors. *IEEE Transactions on Cybernetics*, 46(12):3209–3219, 2016.

[79] Xi Peng, Lei Zhang, and Zhang Yi. Scalable sparse subspace clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 430–437, 2013.

[80] Stephen Tierney, Junbin Gao, and Yi Guo. Subspace clustering for sequential data. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1019–1026, 2014.

[81] Sheng Li, Kang Li, and Yun Fu. Temporal subspace clustering for human motion segmentation. In *IEEE International Conference on Computer Vision*, pages 4453–4461, 2015.

[82] Ehsan Elhamifar and René Vidal. Sparse subspace clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2797, 2009.

[83] Guangcan Liu, Zhouchen Lin, and Yong Yu. Robust subspace segmentation by low-rank representation. In *International Conference on Machine Learning*, pages 663–670, 2010.

[84] Ross Messing, Chris Pal, and Henry Kautz. Activity recognition using the velocity histories of tracked keypoints. In *IEEE International Conference on Computer Vision*, pages 104–111, 2009.

[85] Juan Carlos Niebles, Chih-Wei Chen, and Li Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *European Conference on Computer Vision*, pages 392–405, 2010.

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

[87] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. In *International Conference on Pattern Recognition*, pages 32–36, 2004.

[88] Zhuolin Jiang, Zhe Lin, and Larry Davis. Recognizing human actions by learning and matching shape-motion prototype trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):533–547, 2012.

[89] Marcin Marszalek, Ivan Laptev, and Cordelia Schmid. Actions in context. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2929–2936, 2009.

[90] Ludmila I Kuncheva. *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2004.

[91] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. *Image analysis*, pages 363–370, 2003.

[92] Malay Ranjan Tripathy, Kapil Sachdeva, and Rachid Talhi. 3d discrete wavelet transform vlsi architecture for image processing. In *Progress in Electromagnetics Research Symposium*, pages 1569–1573, 2009.

[93] Yimin Yang and QM Jonathan Wu. Multilayer extreme learning machine with subnetwork nodes for representation learning. *IEEE Transactions on Cybernetics*, 46(11):2570–2583, 2016.

[94] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[95] Xiao-Qin Cao and Zhi-Qiang Liu. Type-2 fuzzy topic models for human action recognition. *IEEE Transactions on Fuzzy Systems*, 23(5):1581–1593, 2015.

[96] Jun Lei, Guohui Li, Jun Zhang, Qiang Guo, and Dan Tu. Continuous action segmentation and recognition using hybrid convolutional neural network-hidden markov model model. *IET Computer Vision*, 2016.

[97] Soumitra Samanta and Bhabatosh Chanda. Space-time facet model for human activity classification. *IEEE Transactions on Multimedia*, 16(6):1525–1535, 2014.

[98] Sushma Bomma and Neil M Robertson. Joint classification of actions with matrix completion. In *International Conference on Image Processing*, pages 2766–2770, 2015.

[99] Daniel Paul Barrett and Jeffrey Mark Siskind. Action recognition by time-series of retinotopic appearance and motion features. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(12):2250–2263, 2016.

[100] Alessandro Prest, Vittorio Ferrari, and Cordelia Schmid. Explicit modeling of human-object interactions in realistic videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(4):835–848, 2013.

[101] Piotr Bilinski and Francois Bremond. Video covariance matrix logarithm for human action recognition in videos. In *International Joint Conference on Artificial Intelligence*, pages 2140–2147, Jul. 2015.

[102] Eman Mohammadi, Wu Q. M. J., and Mehrdad Saif. Human activity recognition using an ensemble of support vector machines. In *IEEE International Conference on High Performance Computing and Simulation*, pages 549–554, 2016.

[103] Amit Singhal. Modern information retrieval: A brief overview. *IEEE Data Engineering Bulletin*, 24(4):35–43, 2001.

[104] Tie-Yan Liu et al. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[105] Xiaofei He, Ming Ji, Chiyuan Zhang, and Hujun Bao. A variance minimization criterion to feature selection using laplacian regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(10):2013–2025, Oct. 2011.

[106] Heng Wang, Dan Oneata, Jakob Verbeek, and Cordelia Schmid. A robust and efficient video representation for action recognition. *International Journal of Computer Vision*, 119(3):219–238, 2016.

[107] Lijun Li and Shuling Dai. Action recognition with spatio-temporal augmented descriptor and fusion method. *Multimedia Tools and Applications*, pages 1–17, 2016.

[108] Arthur E Hoerl and Robert W Kennard. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 42(1):80–86, 2000.

[109] Peter L Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):525–536, 1998.

[110] LiMin Wang, Yu Qiao, and Xiaoou Tang. Mining motion atoms and phrases for complex action recognition. In *IEEE International Conference on Computer Vision*, pages 2680–2687, 2013.

[111] Limin Wang, Yu Qiao, and Xiaoou Tang. Latent hierarchical model of temporal structure for complex activity classification. *IEEE Transactions on Image Processing*, 23(2):810–822, Feb. 2014.

[112] Shiwei Zhang, Changxin Gao, Feifei Chen, Sihui Luo, and Nong Sang. Group sparse-based mid-level representation for action recognition. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(4):660–672, 2017.

[113] Li Liu, Ling Shao, Xuelong Li, and Ke Lu. Learning spatio-temporal representations for action recognition: A genetic programming approach. *IEEE Transactions on Cybernetics*, 46(1):158–170, 2016.

[114] Ke Xu, Xinghao Jiang, and Tanfeng Sun. Two-stream dictionary learning architecture for action recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(3):567–576, 2017.

[115] Yu-Gang Jiang, Qi Dai, Wei Liu, Xiangyang Xue, and Chong-Wah Ngo. Human action recognition in unconstrained videos by explicit motion modeling. *IEEE Transactions on Image Processing*, 24(11):3781–3795, 2015.

[116] Zhenzhong Lan, Ming Lin, Xuanchong Li, Alexander G. Hauptmann, and Bhiksha Raj. Beyond gaussian pyramid: Multi-skip feature stacking for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2015.

[117] Eman Mohammadi, QM Jonathan Wu, and Mehrdad Saif. Improved rank pooling strategy for complex action recognition. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 1351–1356, 2017.

[118] Xin Zhang, Fuchun Sun, Guangcan Liu, and Yi Ma. Fast low-rank subspace segmentation. *IEEE Transactions on Knowledge and Data Engineering*, 26(5):1293–1297, 2014.

[119] Ulrike Von Luxburg. A tutorial on spectral clustering. *Computational Statistics*, 17(4):395–416, 2007.

[120] Yimin Yang, QM Jonathan Wu, and Yaonan Wang. Autoencoder with invertible functions for dimension reduction and image reconstruction. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2016.

[121] Yimin Yang, Yaonan Wang, and Xiaofang Yuan. Bidirectional extreme learning machine for regression problem and its learning effectiveness. *IEEE Transactions on Neural Networks and Learning Systems*, 23(9):1498–1505, 2012.

[122] Can-Yi Lu, Hai Min, Zhong-Qiu Zhao, Lin Zhu, De-Shuang Huang, and Shuicheng Yan. Robust and efficient subspace segmentation via least squares regression. In *European Conference on Computer Vision*, pages 347–360, 2012.

[123] Sheng Li and Yun Fu. Learning balanced and unbalanced graphs via low-rank coding. *IEEE Transactions on Knowledge and Data Engineering*, 27(5):1274–1287, 2015.

[124] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[125] Junsong Yuan, Zicheng Liu, and Ying Wu. Discriminative video pattern search for efficient action detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1728–1743, 2011.

[126] Tam V Nguyen, Zheng Song, and Shuicheng Yan. Stap: Spatial-temporal attention-aware pooling for action recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(1):77–86, 2015.

[127] Guan Luo, Shuang Yang, Guodong Tian, Chunfeng Yuan, Weiming Hu, and Stephen J Maybank. Learning human actions by combining global dynamics and local appearance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(12):2466–2482, 2014.

Vita Auctoris

NAME:	Eman Mohammadi Nejad
DATE OF BIRTH:	1984
PLACE OF BIRTH:	Iran
EDUCATION:	Doctor of Philosophy in Electrical and Computer Engineering, University of Windsor, Windsor, Ontario, Canada, 2018.
	Master of Science in Electronics and Communications Engineering, De La Salle University, Manila, Philippines, 2014.
	Bachelor of Science in Electronics Engineering, National Aviation University, Kiev, Ukraine, 2009.