EFFICIENT ACTION AND EVENT RECOGNITION IN VIDEOS USING EXTREME LEARNING MACHINES

by

Gül Varol

B.S., Computer Engineering, Boğaziçi University, 2013

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University
2015

EFFICIENT ACTION AND EVENT RECOGNITION IN VIDEOS USING EXTREME LEARNING MACHINES

APPROVED BY:

Assist. Prof. Albert Ali Salah .....
(Thesis Supervisor)

Prof. Lale Akarun .....

Assoc. Prof. Yusuf Sinan Akgül .....

DATE OF APPROVAL: 27.03.2015

To my grand-father

ACKNOWLEDGEMENTS

With my deepest gratitude, I would like to thank to my thesis supervisor Albert Ali Salah for his insight, guidance and endless support during the course of my thesis.

I would like to thank Lale Akarun and Yusuf Sinan Akgül for participating in my thesis committee and their valuable comments. I also thank my professors Ethem Alpaydın and Fatih Alagöz for their support throughout my academic life in Boğaziçi.

I would like to thank the members of Perceptual Intelligence Laboratory, Alp Kındıroğlu, Barış Evrim Demiröz, Barış Kurt, Cihan Camgöz, Doğa Sıylı, Hakan Güldaş, Heysem Kaya, Orhan Sönmez and Umut Şimşekli for providing a nice working environment.

I would especially like to express my gratitude to my family for their endless support throughout my life.

I would like to thank my friends Başak Çaprak, Ece Uslu, İlayda Kalaycı, İlker İnanç and Ömür Turan for their valuable friendship. Last but not least, I would like to thank Umut Şimşekli for his encouragement, patience and guidance in all parts of my life.

ABSTRACT

EFFICIENT ACTION AND EVENT RECOGNITION IN VIDEOS USING EXTREME LEARNING MACHINES

A great deal of research in computer vision community has gone into action and event recognition studies. Automatic video understanding for actions are crucial for application areas such as video indexing, surveillance and video summarization. In this thesis, we explore action and event recognition on RGB videos both in terms of feature extraction and classification. We propose a novel approach for large-scale action recognition in a realistic setting. After reviewing the technical background about recent popular video description methods, we present our approach in which improved dense trajectory features in combination with Fisher vector encoding are fed to extreme learning machine classifier. It is shown that extreme learning machine provides a fast and accurate alternative to other traditional classifiers such as support vector machines. Additionally, we investigate the usability of some mid-level features that we introduce to encode information about human part regions. We extensively study each step of our pipeline in a comparative manner. We evaluate our approach on recently published benchmarks which were introduced as challenge datasets: UCF101, THUMOS 2014 and ChaLearn Looking at People 2014 Track 2. Videos in the first dataset contain cropped actions while the ones in the last two datasets are temporally untrimmed, introducing more challenge. On 102 action classes of THUMOS 2014 dataset, we achieve 63.37% mean average precision using the challenge protocol, which has ranked 3rd among other participants. Our results show that, using extreme learning machine, efficient learning can be performed in terms of both time and computational complexity while preserving high performance.

ÖZET

AŞIRI ÖĞRENME MAKİNELERİ İLE VİDEOLARDA ETKİN HAREKET VE ETKİNLİK TANIMA

Bilgisayarla görme alanında, hareket ve etkinlik tanıma üzerine birçok araştırma yapılmıştır. Video indeksleme, gözetim ve video özetleme gibi uygulama alanları için videolarda hareket tanıma oldukça önem taşır. Bu tezde, KYM videolarda hareket ve etkinlik tanıma, hem öznitelik çıkarma hem de sınıflandırma açısından araştırılmaktadır. Gerçekçi ortamda büyük ölçekli hareket tanıma problemi için yeni bir yaklaşım önerilmektedir. Video betimleme yöntemlerinin üzerinden geçildikten sonra, önerilen yaklaşım tanıtılmaktadır. Bu yaklaşımda, Fisher vektörleri ile tanımlanmış yerel yörünge öznitelikleri, aşırı öğrenme makinesi (extreme learning machine) sınıflandırıcısına verilmektedir. Aşırı öğrenme makinesinin, destek vektör makinesi gibi diğer sınıflandırıcılara göre daha hızlı ve başarılı bir alternatif olduğu gösterilmiştir. Ek olarak bu çalışmada, insan vücudu bölümleri hakkında bilgi içeren bazı orta seviye özniteliklerin bu problem için kullanılabilirliği araştırılmaktadır. Önerilen yaklaşımın her basamağı yoğun ve karşılaştırmalı bir şekilde incelenmektedir. Değerlendirmeler yakın zamanda ilk olarak yarışmalar için yayınlanmış gösterge veri kümeleri üzerinde yapılmaktadır. Bunlar UCF101, THUMOS 2014 ve ChaLearn Looking at People 2014 Track 2 olarak sıralanabilir. İlk veri kümesindeki videolar sadece hareket içerecek şekilde kırpılmışken, diğer iki veri kümesindekiler zamansal olarak kırpılmamıştır ve dolayısıyla daha zor koşullar içerir. THUMOS 2014 veri kümesindeki 102 hareket sınıfı üzerinde %63.37 ortalama başarı elde edilmiştir. Bu sistem THUMOS yarışmasında üçüncülük almıştır. Sonuçlarımız aşırı öğrenme makinesinin hem hesaplama açısından etkin, hem de yüksek başarılı bir sınıflandırıcı olduğunu göstermiştir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iv
ABSTRACT v
ÖZET vi
LIST OF FIGURES x
LIST OF TABLES xii
LIST OF SYMBOLS xiii
LIST OF ACRONYMS/ABBREVIATIONS xiv
1. INTRODUCTION 1
1.1. Motivation 1
1.2. Related Work 3
1.3. Contributions 5
1.4. Organization of the Thesis 6
2. VIDEO DESCRIPTION WITH LOCAL DESCRIPTORS 7
2.1. Detectors 7
2.1.1. Spatio-Temporal Interest Points 8
2.1.2. Dense Sampling 9
2.2. Descriptors 10
2.2.1. Histogram of Oriented Gradients 10
2.2.2. Histogram of Optical Flow 10
2.2.3. Motion Boundary Histogram 11
2.3. Trajectories 11
2.3.1. Point Tracking and Optical Flow 11
2.3.2. Dense Trajectories 12
2.3.3. Improved Trajectories 15
2.3.3.1. Homography Matrix 15
2.3.3.2. Feature matching 16
2.3.3.3. RANSAC 16
2.4. Local Feature Aggregation 16
2.4.1. Unsupervised Clustering Methods 17

2.4.1.1. K-means 17
2.4.1.2. Gaussian Mixture Model 18
2.4.2. Bag of Features 20
2.4.3. Fisher Vector 21
2.4.4. Vector of Locally Aggregated Descriptors 24
3. ACTION CLASSIFICATION 26
3.1. Support Vector Machines 26
3.2. Extreme Learning Machines 29
3.2.1. Random projections 31
3.2.2. Kernels 32
4. PROPOSED METHODOLOGY 34
4.1. Motion Features 34
4.2. Mid-Level Features 36
4.3. Classification 38
4.4. Localization 39
5. EXPERIMENTS 41
5.1. Datasets 41
5.1.1. ChaLearn Looking at People 2014 Track 2 41
5.1.2. UCF101 43
5.1.3. THUMOS 2014 45
5.2. Comparison of Descriptor Types 47
5.3. Contribution of Mid-Level Features 48
5.4. Comparison of Different Encodings 50
5.5. Comparison of ELM and SVM 51
5.6. Comparison of Frame-Level and Window-Level Classifications 53
5.7. The Effect of Fisher Vector Parameters 55
5.8. Model Selection in Extreme Learning Machine 56
5.9. Results 59
6. CONCLUSION 61
6.1. Remarks 61
6.2. Future Work 63

REFERENCES 65

LIST OF FIGURES

Figure 1.1. Examples of actions, interactions and events. 1

Figure 1.2. Sample action categories. 2

Figure 2.1. STIP extraction. 9

Figure 2.2. Optical flow visualization. 12

Figure 2.3. Improved dense trajectories pipeline. 14

Figure 2.4. Improved trajectories visualization. 15

Figure 2.5. K-means algorithm. 18

Figure 2.6. Bag of features. 20

Figure 3.1. Single-hidden-layer feed-forward network architecture. 30

Figure 4.1. Proposed pipeline. 35

Figure 4.2. Color histograms for sample video frames. 37

Figure 4.3. Body part detections. 38

Figure 5.1. ChaLearn LAP 2014 Track 2 dataset overview. 42

Figure 5.2. UCF101 and THUMOS 2014 datasets overview. 45

Figure 5.3. Challenge results (THUMOS). 48

Figure 5.4. Contribution of mid-level features (THUMOS) 49

Figure 5.5. Comparison of BOF and FV encodings (THUMOS). 51

Figure 5.6. Comparison of SVM and ELM (ChaLearn). 53

Figure 5.7. Frame-level classification and post-processing (ChaLearn). 54

Figure 5.8. Frame-level classification and FV parameters (ChaLearn). 55

Figure 5.9. Window-level classification and FV parameters (ChaLearn). 56

Figure 5.10. Hyperparameters in ELM with kernels (THUMOS). 57

Figure 5.11. Hyperparameters in ELM with random projections (THUMOS). . 58

Figure 5.12. Confidence scores and predictions (ChaLearn). 58

Figure 5.13. Confusion matrix (ChaLearn). 59

Figure 5.14. Confusion matrices (UCF101, THUMOS). 59

LIST OF TABLES

Table 1.1. Action recognition datasets and benchmarks. 3

Table 2.1. Dense trajectory parameters. 14

Table 2.2. Typical attributes of aggregation methods BOF, VLAD and FV. . 24

Table 5.1. List of classes in UCF101 and THUMOS 2014 datasets. 44

Table 5.2. Comparison of descriptor types MBH, HOF and HOG (THUMOS). 46

Table 5.3. Comparison of descriptor types MBH, HOF and HOG (UCF101). . 47

Table 5.4. Mid-level feature concatenation performance (THUMOS). 50

Table 5.5. Comparison of SVM and ELM (THUMOS). 52

LIST OF SYMBOLS

C_{ELM}	Regularization parameter of ELM
C_{SVM}	Cost parameter of SVM
D	Dimensionality
$g(x; u, \Sigma)$	Gaussian component
$G(w, b, x)$	Activation function
H	Hidden layer matrix
$\mathrm{H}^{\dagger}$	Inverse of H
I	Identity matrix
$I(x, y, t)$	Optical flow
K	Number of cluster components
$K(x_i, x_j)$	Kernel function
n_s	Number of spatial divisions
n_t	Number of temporal divisions
N	Number of instances
$p(x\|\lambda)$	Probability of x given λ
P_t	Point at frame t
T	Training label matrix
y_i	Class label of instance i
β	Hidden layer output weight matrix
$\mathcal{L}$	Log-likelihood
μ	Mean
π	Pi number
σ	Variance
Σ	Covariance matrix
Ω	Kernel matrix
∇_{λ}	Gradient vector with respect to parameter λ

LIST OF ACRONYMS/ABBREVIATIONS

2D	Two Dimensional
3D	Three Dimensional
BOF	Bag of Features
DS	Detector Statistics
ELM	Extreme Learning Machine
EM	Expectation-Maximization
FV	Fisher Vector
GMM	Gaussian Mixture Model
HOF	Histogram of Oriented Gradients
HOG	Histogram of Optical Flow
HOG3D	Histograms of 3D Gradient Orientations
HSVH	Hue-Saturation-Value Histogram
IT	Improved Trajectories
JI	Jaccard Index
KLT	Kanade-Lucas-Tomasi
LAP	Looking at People
LDOF	Large Displacement Optical Flow
mAP	Mean Average Precision
MBH	Motion Boundary Histograms
NN	Nearest Neighbor
PCA	Principal Component Analysis
RAM	Random Access Memory
RANSAC	Random Sample Consensus
RBF	Radial-basis Function
RGB	Red-Green-Blue
RGBH	Red-Green-Blue Histogram
SLFN	Single-hidden-layer Feed-forward Network
STIP	Spatio-Temporal Interest Point

SURF	Speeded Up Robust Features
SVM	Support Vector Machine
UCF101	University of Central Florida 101
VLAD	Vector of Locally Aggregated Descriptors

1. INTRODUCTION

1.1. Motivation

Automatic video analysis has attracted increasing interest due to the exponential growth of video data over the recent years. The focus of the research community is mostly on human activity recognition in videos in consideration of the potential application areas such as video surveillance, video indexing/retrieval [1], ambient assisted living [2], and video summarization [3].

Action is defined as "a semantically defined set of movements of the agent" [4], such as walking. Activity or event is defined as "an action of the agent in a particular context" [4], such as parade. The notion of action can be extended to interaction when two performers have two-way effect. In Figure 1.1, we see examples of an action, interaction and activity. In this thesis, we refer to all three levels of knowledge when we mention action recognition.

Action recognition in computer vision domain aims to understand the human activity in a given image or video. Inferring actions from still images is a widely studied problem [5–7]. However, it lacks motion information which can be extracted in the presence of an image sequence. Recognizing actions from image sequences (i.e. videos) is a broader problem and we address this problem in our work.

(a) Point (b) Shake Hands (c) Military Parade

Figure 1.1. An action category can be performed by one person (a), two people (b) or more (c).

(a) Horse Riding (b) Horse Race (c) Mopping Floor (d) Mopping Floor

Figure 1.2. Low between-class difference (a-b), high within-class difference (c-d).

Despite the great effort that computer vision scientists invested in the action recognition task, robust inference on actions from image sequences remains a challenge. Human action is complex due to a variety of reasons such as viewpoint variance, illumination, background complexity, camera motion, occlusion, and performance difference among different people. The resolution of videos is another issue when videos collected from the internet are considered. Low quality compression makes it difficult to track objects or points in a scene.

Designing robust, informative and discriminative features for video representation is essential to overcome the difficulties listed above. Such features should take into account the within-class differences (such as fast walking and slow walking) and between-class similarities (such as walking and jogging). Figure 1.2 illustrates this issue. Although "horse riding" and "horse race" denote two different action categories, they have high similarity in terms of motion and context. On the other hand, two distinct "mopping floor" examples may demonstrate dissimilar performances when performed by an adult and a child.

Besides the difficulty of action classification, there exists the action spotting problem which is less addressed by the research community [8–10]. However, in some applications such as surveillance and summarization, it is a natural requirement to segment large videos automatically. Most of the standard human action datasets involve pre-segmented clips of single actions [11] although real-world video streams involve uncontrolled action sequences. In this study, we propose a novel approach for recognizing actions in such a setting, from a single camera.

Table 1.1. Action recognition datasets and benchmarks.

Dataset		# Videos	# Actions	Source	Year
KTH	[12]	600	6	recorded	2004
Weizmann	[13]	90	10	recorded	2005
UCF Sports	[14]	200	9	stock footage websites	2008
Hollywood	[1]	663	8	32 movies	2008
UCF11(Youtube)	[15]	800	11	Youtube	2009
Hollywood 2	[16]	3669	12	69 movies	2009
UCF50	[17]	6676	50	Youtube	2010
HMDB51	[18]	6849	51	mostly from movies	2011
ASLAN	[19]	3697	432	Youtube	2012
UCF101	[20]	13320	101	Youtube	2012
TRECVID HAVIC	[21]	8000 hours	20	internet	2012
THUMOS 2014	[22]	18404	102	Youtube	2014

1.2. Related Work

The progress in action and event recognition research is evolving with the development in video datasets. Chaquet *et al.* [23] recently published a survey on human event datasets. Table 1.1 summarizes some benchmark datasets for action recognition in videos with up-to-date entries. Schuldt *et al.* [12] introduced KTH dataset in 2004 for studying human action from videos. This database contains grayscale videos of six human actions, namely walking, jogging, running, boxing, hand waving and hand clapping. The background in these videos is uniform and stable. In each video, there is one performer. Weizmann dataset [13] is similar in terms of simplicity because it is also considered as recordings in lab conditions.

The datasets published each year are growing in terms of number of video content as well as the action class variety. Recently published datasets, such as HMDB51 [18] and UCF101 [20], have over 50 action categories and over 5000 videos. These videos are sourced usually from the internet. Therefore, they have large complexity.

Below, we briefly review the well-known techniques in the literature that are used to treat action recognition problem. These techniques will be detailed in Chapter 2.

In computer vision, the action recognition pipeline typically involves feature sampling, description, aggregation and classification steps. Various approaches have been proposed for sampling feature points. Laptev [24] introduced space-time interest point (STIP) detectors. Wang *et al.* [25] showed that dense sampling outperforms sparse sampling. In a follow up study [26], they introduced improved trajectories by camera motion calibration, which further improved recognition performance in unconstrained videos.

A key problem for a successful recognition is how to treat the temporal motion information and spatial gradient information to extract informative and robust features. Some of the popular low-level descriptors are histograms of 3D gradient orientations (HOG3D) [27, 28], oriented histograms of flow (HOF) [1] and motion boundary histograms (MBH) [25, 29]. Action banks [8], action attributes [30], actons [31] and motionlets [32] are proposed as mid-level features, while recently higher-level representations are being investigated. Wang and Schmid [26] recently proposed an approach based on MBH, HOG, HOF descriptors sampled along improved dense trajectories, which yields state-of-the-art results on most action datasets.

Pooling local descriptors with bag of features (BOF) technique has been widely used [33, 34] and has been previously adopted as the main paradigm for video representation [1]. The pipeline for traditional BOF consists of local feature extraction, visual dictionary construction with a clustering algorithm such as k-means and feature encoding in an aggregated manner. A recent study demonstrated that a better encoding, namely Fisher vector (FV) representation, significantly increases recognition performance [35]. Other recent encoding methods, such as stacked Fisher vectors [36] and super sparse coding vectors [37] are introduced to embed some structure information.

The last step in the pipeline is classification, where most studies use the popular support vector machine (SVM) approach, with single or multiple kernel learning [11,

26,35,38,39]. In our study, we propose the extreme learning machine (ELM) approach, which is recently gaining popularity, and which tends to have better generalization performance for multiclass classification with much shortened learning times [40].

1.3. Contributions

One of the most common learning algorithms for action recognition is the support vector machine. SVM is a powerful tool for the classification task; however, it requires iterative learning and extensive parameter tuning. Therefore the training might be very slow depending on the feature dimensionality and the number of instances. Hyperparameter optimization, while using SVM, plays a key role in obtaining high performance. Thus, searching over a parameter grid for optimization purposes might take a long time.

In this thesis, we propose to use extreme learning machines which is suitable for large-scale action recognition. It is effective because there is no iterative tuning required. The training time is reduced compared to SVM. We contrast several SVM approaches with ELM, and illustrate the strengths and weaknesses of these classifiers for this problem. Our results show that more accurate performances are obtained with ELM. To the best of our knowledge, we are the first to apply ELM on a large-scale problem of action recognition with a large number of classes.

Low-level features perform well in action recognition problems. Higher level representations are less addressed by the research community. We introduce some mid-level features which carry information about the color and the regions occupied by body part regions in a given video. In realistic videos, these types of mid-level representations may help reducing the problem from 101-classes to 101-class problem to several smaller problems, which are easier to deal with. Although we start with the motivation of breaking down the large-scale problem into smaller problems, we end up using the designed mid-level features for early-fusion with the motion features extracted from improved dense trajectories.

In 2014, the THUMOS Challenge[1] at the European Conference on Computer Vision (ECCV) addressed large-scale action recognition with large number of classes from open source videos in a realistic setting [22]. Many algorithms competed in this challenge. We have implemented an entry to this contest, using ELM as the classifier, and Fisher vectors as the main feature representation, and took the 3rd place among 11 groups [41]. We have also submitted a journal paper as an extension to this study [42]. In this thesis, we describe our approach in detail, and then extend it, using the formal challenge protocol to report comparative results.

1.4. Organization of the Thesis

The rest of this thesis is organized as follows. We review technical background of action recognition in videos in Chapters 2 and 3. In Chapter 2, we present popular techniques for video description with local descriptors. The notions of interest points, and local motion descriptors in space-time volumes are detailed. The evolution of trajectory-based methods are described, and finally, the local feature aggregation methods are explained. In Chapter 3, two classification methods, support vector machines and extreme learning machines are presented. We detail our proposed methodology in terms of feature extraction and classification in Chapter 4. We report and discuss our results in Chapter 5, where we extensively study the effects of certain parameters. Finally, we conclude our work and state some possible future work in Chapter 6.

[1] http://crcv.ucf.edu/THUMOS14/home.html

2. VIDEO DESCRIPTION WITH LOCAL DESCRIPTORS

Given a video clip, we need to represent the raw data of pixels into a compact form, which is informative in terms of action content. This representation needs to be distinctive, low-dimensional, interpretable and with a fixed size. Distinctive representations are more successfully recognized by classifiers. Low-dimensional vectors are favorable because they require less storage and do not suffer from curse of dimensionality. Interpretability can be important, for instance when one needs to know which information is encoded in which part of the feature vector or what type of distance measure should be used. Finally, a global representation for a video clip should be fixed size for each instance to be able to use a classification algorithm.

Bottom-up approaches are frequently used for video description in action recognition and they are based on low-level features [4]. In this chapter, we review several methods for encoding local motion and structure information into compact forms. A typical low-level representation involves detecting feature points, describing them in a local neighborhood and summarizing them with some statistics.

2.1. Detectors

Sampling is the first step for determining which feature points to describe. There are sparse and dense approaches for feature sampling. Sparse methods assume that sampling only the interest points is sufficient for describing the video content. Interest points, known also as keypoints or corners, have rich local information because they have high variation in space and/or time. Dense methods simply sample points from a regular grid with or without multiscale pyramid. The motivation is that dense coverage of the video domain ensures that the context information is also captured. There are also methods which use random sampling strategies [43], but we do not address these approaches in this work.

2.1.1. Spatio-Temporal Interest Points

Local neighborhood of interest points are believed to be informative regions. Similar to the spatial domain, interest point detection can be applied to the spatio-temporal domain. Laptev [24] introduced spatio-temporal interest points (STIP) (also called Harris3D detector) which extends the local image features into the video domain integrating temporal information. Corners in a spatio-temporal volume correspond to points where local variations occur in space as well as in time. Spatio-temporal interest points are expected to have non-constant motion in their local neighborhood. Descriptors extracted along these keypoints enable to compactly represent image sequences.

STIP detection is based on Harris corner function [44] which is defined for the spatial domain. An image sequence is defined as a function $f : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}$. Let g be an anisotropic Gaussian kernel (i.e. different variance in different dimensions) with spatial variance σ^2 and temporal variance τ^2. Then, the convolution

$$L = g * f, \tag{2.1}$$

becomes the linear scale-space representation of f. Averaging the first-order spatial and temporal derivatives by the weighting function g gives the second-moment matrix μ, which is defined as

$$\mu = g * \begin{pmatrix} L_x^2 & L_x L_y & L_x L_t \\ L_x L_y & L_y^2 & L_y L_t \\ L_x L_t & L_y L_t & L_t^2 \end{pmatrix}. \tag{2.2}$$

Here, L_x, L_y and L_t denote the first-order derivatives of L. Significant eigenvalues of μ correspond to interest points in f. Let H be

$$H = \det \mu - k \operatorname{trace}^3(\mu) \tag{2.3}$$

$$= \lambda_1 \lambda_2 \lambda_3 - k(\lambda_1 + \lambda_2 + \lambda_3)^3, \tag{2.4}$$

(a) Wave (b) Jump (c) Fight (d) Point

Figure 2.1. STIP extraction on ChaLearn dataset.

where λ_1, λ_2, and λ_3 are eigenvalues of μ. Spatio-temporal interest points can be found in positive local maxima of H.

Schuldt *et al.* [12] used STIP for action recognition by extracting local descriptors along the computed interest points. Laptev *et al.* [1] improved STIP performance for action recognition by dense scale sampling instead of scale selection as in [24]. In Section 2.2, we will review the types of several local descriptors.

2.1.2. Dense Sampling

In dense sampling, the feature points are extracted at a multi-scale regular grid. The video blocks are sampled from 5 dimensions (x, y, t, σ, τ). That means a feature point p is positioned at (x, y, t) in the space-time volume and the video patch centered at p has size determined by the scale (σ, τ).

In an evaluation by Wang *et al.* [33], we see that dense sampling outperforms the space-time interest point detection. The explanation is related to the importance of context information in recognition. In real-world datasets, background may involve cues about the performed action (e.g. basketball player next to the basketball hoop). Moreover, some actions cannot be recognized by motion but with appearance (e.g. playing flute).

2.2. Descriptors

Space-time volumes extracted around sampled points are commonly used for action recognition. Most of the local descriptors rely on gradient information; that is, the change in spatial and/or temporal space. Depending on the definition of the descriptor, it may capture appearance and/or motion information.

2.2.1. Histogram of Oriented Gradients

Computing a histogram for the gradient orientations (HOG) for the image domain was first proposed by [45] for human detection problem. Laptev *et al.* [1] introduced an application of a similar approach to spatio-temporal space.

The idea is to divide the space-time volume to be described into $n_x \times n_y \times n_t$ cells and for each cell to count the quantized gradient orientations into a number of bins. For the case of [1], 4 orientations are considered whereas in [25], all 8 orientations are used. These histograms are normalized and concatenated to form HOG descriptor.

HOG encodes local appearance features since it uses structure information. Action categories, which are recognized better with appearance information, benefit from this descriptor.

2.2.2. Histogram of Optical Flow

Similar to HOG, in histogram of optical flow (HOF) [1] we divide the volume into regular cells and compute histograms for each of these cuboids. Instead of gradient vectors, we count the optical flow vectors (See Section 2.3.1 for optical flow). These motion vectors are quantized into 5 bins in [1] and 9 bins (8 orientations + zero bin) in [25]. Normalized histograms for each cell are concatenated to form HOF descriptor.

HOF captures first order local motion information since it uses absolute motion. Actions which involve characteristic motion information benefit from this descriptor.

2.2.3. Motion Boundary Histogram

Motion boundary histograms (MBH) are first proposed by Dalal *et al.* [29] for human detection problem and were later successfully used by Wang *et al.* [25] for action recognition.

Instead of the optical flow vectors, the derivatives of the optical flow are counted for each spatio-temporal cell. The spatial derivatives of optical flow $I_w = (I_x, I_y)$ are computed for both x and y components. The rest is similar to HOF. The orientations are quantized into 8 bins for each components and the normalized histograms are concatenated.

MBH captures second order local motion information since it uses the derivative of the motion. It is robust to camera motion because constant motion is suppressed. Therefore; the motion information on the boundaries of moving objects is encoded.

2.3. Trajectories

2.3.1. Point Tracking and Optical Flow

Point tracking problem is the basis of many computer vision applications. Given a video and the initial position of a certain point, we want to estimate the position of this point in the next frame(s).

Point tracking is directly related to estimating optical flow in which a distribution of velocities in an image are determined. Figure 2.2 is a visualization of optical flow, where we can see the magnitudes of velocity vectors. Optical flow is described in terms of intensity $I(x, y, t)$, by introducing brightness constancy constraint:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t), \tag{2.5}$$

where $\Delta x, \Delta y$ and Δt denote the displacement of point (x, y, t) between two frames.

Figure 2.2. Optical flow visualization on a "hula hoop" instance on UCF101 dataset.

A well-known method for optical flow estimation is Lucas-Kanade method [46] which makes the assumption of constant flow under local neighborhood. By building on this method, Shi and Tomasi developed the frequently-used KLT tracking method [47]. KLT tracker considers cornerness values of tracked points, so it is a sparse tracking technique.

A more recent optical flow method is introduced by Farnebäck [48] and its implementation is integrated in OpenCV [49]. In contrast to KLT tracker, this algorithm produces dense optical flow. It approximates each neighborhood of two frames by quadratic polynomials and the global displacement vector is computed by using the coefficients in these quadratic polynomials.

A state-of-the-art dense optical flow algorithm is large displacement optical flow (LDOF) by Brox and Malik [50]. This method is able to capture the movement of small structures such as hands; however, it is computationally extensive.

2.3.2. Dense Trajectories

Action recognition with dense trajectories is proposed by Wang *et al.* [25] and has recently been widely used in the literature [51, 52]. Following the success of dense sampling over sparse sampling methods in the image domain, in this method, points are sampled densely from multiscale pyramid from each frame of the video. These points are then tracked for a certain time window. The tracking is based on dense

optical field computation [48] and is applied on each spatial scale separately. Local descriptors, which are aligned with the trajectories, are extracted to encode local motion information. Dense trajectories method is illustrated in Figure 2.3, which is a re-drawn illustration from the original paper [25].

A trajectory is defined as the concatenation of $\{P_t, P_{t+1}, P_{t+2}, ...\}$ in [25], where

$$P_{t+1} = (x_{t+1}, y_{t+1}) \tag{2.6}$$

$$= (x_t, y_t) + (M * \omega)|_{(\hat{x}_t, \hat{y}_t)}. \tag{2.7}$$

Here, $(\hat{x}_t, \hat{y}_t)$ denotes the rounded position of (x_t, y_t). The optical flow field ω is convolved with median filtering kernel M to track point P_t to the next frame.

In order to remove false trajectories, some additional processes are applied. To avoid large jumps between consecutive frames, points with displacement larger than a threshold are removed. The points which lie in uniform regions are unlikely to be tracked. These points are not tracked if their cornerness value is below a certain threshold. To avoid drifting in tracking, the points are tracked up to a certain time window of length L frames. Trajectories which do not involve any motion are also removed once the tracking is completed. Lastly, to ensure there are tracked points in each $W \times W$ neighborhood, in every frame, a new feature point is sampled if there aren't any.

The parameters to be defined are listed in Table 2.1. The effect of each of these parameters are extensively studied [25] and in the original implementation[2] , the default values for these parameters are set as stated in Table 2.1.

[2]http://lear.inrialpes.fr/people/wang/dense_trajectories

Table 2.1. Dense trajectory parameters.

Parameter	Definition	Default
W	sampling step of the regular grid	5 pixels
Σ	height of the multiscale pyramid for sampling	8
s	factor between spatial scales	$1/\sqrt{2}$
L	tracking time	15 frames
N	size of the volume for computing descriptors	32 pixels
n_s	number of divisions of the descriptor volume in spatial space	2
n_t	number of divisions of the descriptor volume in temporal space	3

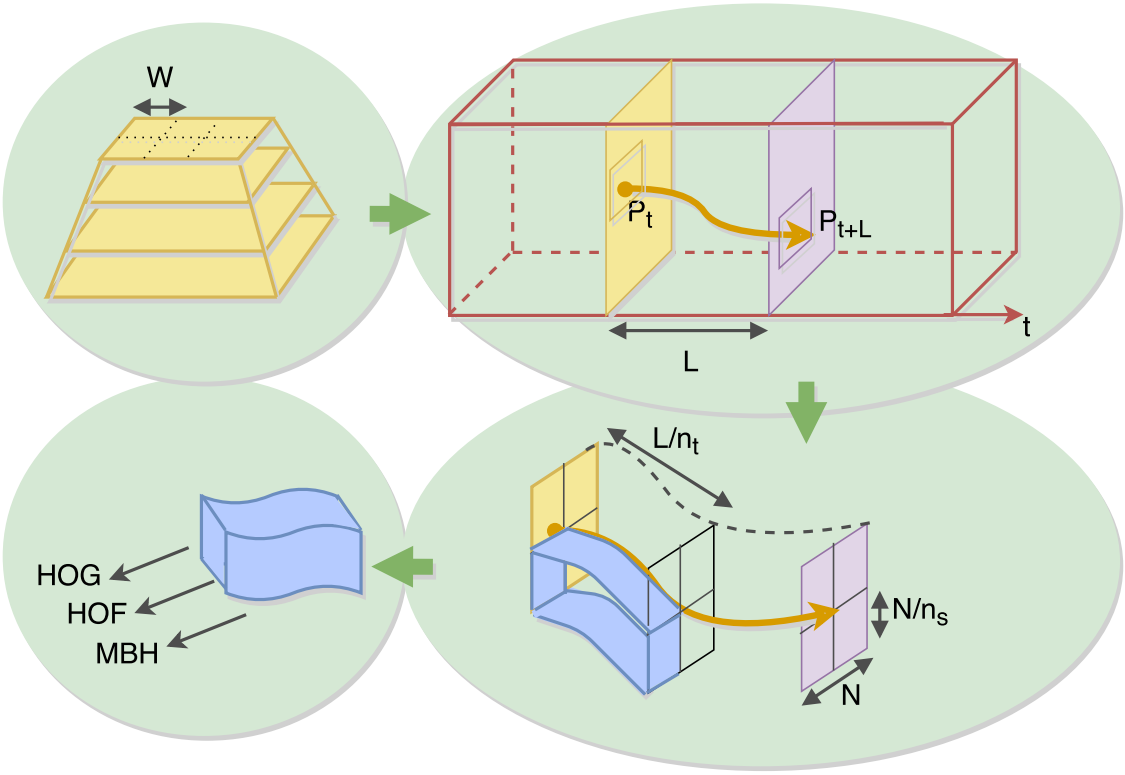


Figure 2.3. Re-drawn figure of the original illustration in [25]. Points are densely sampled from multiple scales and tracked separately. $N \times N$ neighborhood of each trajectory is divided into $n_s \times n_s \times n_t$ blocks from which the descriptors are extracted.

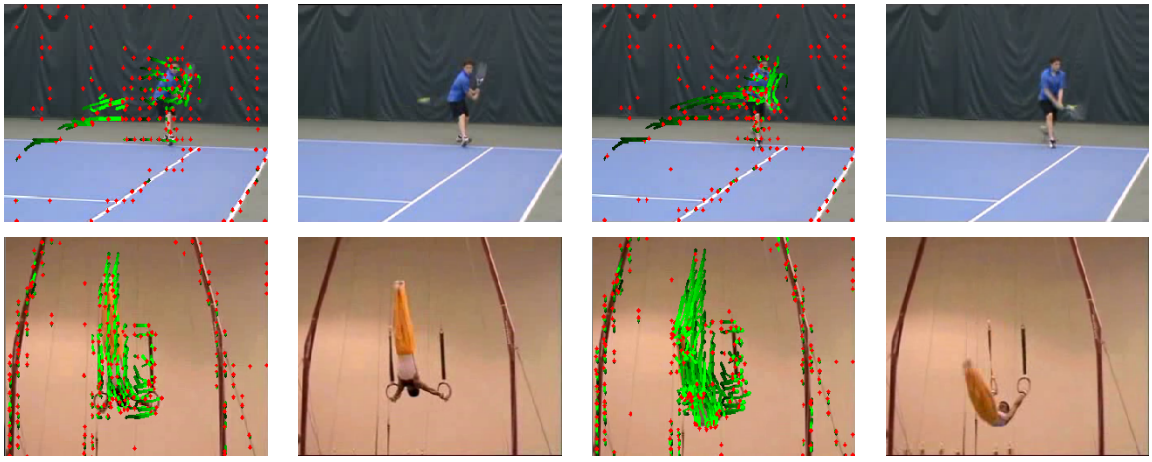


Figure 2.4. Original video frame and corresponding improved trajectories visualization for "tennis swing" and "still rings" classes of UCF101 dataset.

2.3.3. Improved Trajectories

The improvement over the dense trajectory features is achieved by compensating the camera motion effect with homography matrix computation between consecutive frames. These warped features, proposed by Wang *et al.* [26], are shown to perform better than many local descriptors on several benchmark action recognition datasets. Figure 2.4 illustrates the trajectories computed on two UCF101 videos. This visualization provided by authors' software[3] means that red points in the current frame have moved with trajectories denoted by green points.

The motivation behind the improvement is to get rid of the spurious trajectories created by camera motion in realistic videos and to improve the motion descriptors' performance. Once the camera motion is known, the optical field can be re-computed to correct for such motion. Let us revisit homography matrix estimation from feature matching in order to predict camera motion.

2.3.3.1. Homography Matrix. Homographies are 3×3 matrices which define projective transformations. Homography matrix relates two images x and x' on the same planar surface. Two consecutive frames of a video are assumed to have a homography relation

[3] http://lear.inrialpes.fr/people/wang/improved_trajectories

since they usually cover tiny motion. If the camera rotates between views with rotation matrices R_1 and R_2, the relation between 3D world coordinates X and the 2D image plane becomes as follows:

$$\begin{aligned} x &= K[R_1 \quad 0]X \quad \text{and} \quad x' = K[R_2 \quad 0]X \\ x' &= KR_2R_1^{-1}Kx \\ x' &= Hx, \end{aligned} \tag{2.8}$$

where H is the homography and K is the camera intrinsic matrix which depends on the focal length and camera center. In Equation (2.8), x and x' are in homogeneous coordinates.

2.3.3.2. Feature matching. Correspondence between two images is the key to compute a transformation. A selected descriptor type is chosen and extracted from both images. Then, these are matched according to a similarity measure using a matching algorithm. In improved trajectories [26], authors use both Speeded Up Robust Features (SURF) [53] matches and motion vector matches.

2.3.3.3. RANSAC. Random Sample Consensus (RANSAC) [54] is a popular algorithm for robustly estimating homography matrix avoiding the effect of outlier matches. It is an iterative method in which a random sample of feature correspondences is selected at each iteration. The homography matrix is computed, then each correspondence is assigned as inlier or outlier depending on the current homography. At the end of iterations, the homography with the largest number of inliers is selected as the estimate.

2.4. Local Feature Aggregation

The set of local descriptors sampled from the spatio-temporal volume should be represented compactly in a fixed size vector. For this purpose, traditional approaches learn a codebook from the training descriptors and use this codebook as a visual vocabulary. The descriptors are then encoded according to their distribution with respect

to the visual vocabulary. Three encoding techniques are well-known and frequently used: bag of features [1, 25], vector of locally aggregated descriptors [55] and Fisher vectors [35, 36, 56].

2.4.1. Unsupervised Clustering Methods

In computer vision, unsupervised clustering methods are widely used for learning visual vocabularies of local descriptors [5, 25, 33, 35, 57]. In this section, we review two most frequently used clustering methods, namely k-means and Gaussian mixture models. The choice of clustering method is done depending on the encoding technique to be used.

2.4.1.1. K-means. First proposed by Macqueen in 1967 [58], k-means clustering is an unsupervised vector quantization method. A set of features are assumed to be partitioned into groups of data. Our aim is to find such a grouping so that the within-group sum of squares is minimized.

Mathematically speaking, the objective function to be minimized for having optimum clusters $C = \{c_1, ...c_K\}$ is defined as follows:

$$\arg\min_{C} \sum_{i=1}^{K} \sum_{x \in c_i} \|x - \mu_i\|^2, \tag{2.9}$$

where μ_i is the cluster mean of c_i and x are data points. The algorithm to solve this optimization is described in Figure 2.5.

Since k-means algorithm finds a local optimum, commonly it is ran multiple times. Another drawback is that one may not know the number of data groups beforehand. Hence, the choice of K can be made depending on the application and multiple K candidates can be experimentally evaluated for the task at hand.

initialize cluster centers $\mu_{\mathbf{j}},\ j = 1, 2, \ldots, K$ randomly

repeat

  for all i **do**

    $c_i := \arg\min_j \|x_i - \mu_{\mathbf{j}}\|^2$

  end for

  for all j **do**

    $\mu_j := \dfrac{\sum_{i=1}^{m} 1\{c_i = j\} x_i}{\sum_{i=1}^{m} 1\{c_i = j\}}$

  end for

until convergence

Figure 2.5. K-means algorithm.

2.4.1.2. Gaussian Mixture Model. Gaussian mixture model (GMM), when used for clustering, can be seen as a more general version of k-means. K-means is a special case, where the covariances are spherical, the priors are equal, and the cluster memberships are hard, rather than probabilistic. GMM is a generative model which tries to fit a number of Gaussian components on the data. GMM clustering is convenient when data groups have different sizes and different covariance structure between feature dimensions. Different than k-means, the data points do not belong to cluster centers by hard assignment, but the probability of belonging to each cluster is considered.

A Gaussian mixture model is defined by three parameters $\lambda = \{\omega_i, \mu_i, \Sigma_i\}$; mixing weights, means and variances, respectively. K is the number of components and $i = 1, ..., K$. A GMM for D-dimensional x vectors is defined as:

$$f(x) = \sum_{i=1}^{K} \omega_i g(x; \mu_i, \Sigma_i), \quad \text{with} \quad \sum_{i=1}^{K} w_i = 1, \tag{2.10}$$

where $g(x; \mu_i, \Sigma_i)$ is a Gaussian component of the form,

$$g(x; \mu, \Sigma) = (2\pi)^{-D/2} \det(\Sigma)^{-1/2} \exp\{-\frac{1}{2}(x - \mu)^\top \Sigma^{-1} (x - \mu)\}. \tag{2.11}$$

Depending on assumptions about the data, GMM can be constrained to have specific forms. For example, Σ_i can be full rank or diagonal. Moreover, parameters within λ can be shared among components. If all components are equally likely, there is no need to have ω_i parameters. The number of parameters to estimate depend heavily on these assumptions.

A common method to estimate Gaussian mixture model parameters is Expectation-Maximization (EM) algorithm [59]. It is an iterative algorithm alternating between expectation and maximization steps. In expectation step, the log-likelihood is computed using the current values of parameters. In maximization step, the parameters are computed maximizing the expected log-likelihood.

For the context of Gaussian mixture models, EM algorithm is used to learn the parameters λ of a mixture given data $\{x_1, ..., x_N\}$. For each component i, the following updates are applied on the parameters:

$$P(i|x_j) := \omega_i g(x_i; \mu_i, \Sigma_i)/f(x_i) \tag{2.12}$$

$$\omega_i := \sum_{j=1}^{N} P(i|x_j)/N, \tag{2.13}$$

$$\mu_i := \sum_{j=1}^{N} P(i|x_j)x_j/(N\omega_i), \tag{2.14}$$

$$\Sigma_i := \sum_{j=1}^{N} P(i|x_j)(x_j - \mu_i)(x_j - \mu_i)^\top/(N\omega_i). \tag{2.15}$$

The above re-estimation equations guarantee a monotonic increase in the Gaussian mixture model's likelihood on each EM iteration. However, EM finds a local optimum so the initialization is crucial to reach the global optimum. Initialization by k-means is a common choice for GMM learning.

Among various methods to learn GMM parameters, the standard EM algorithm is the best if the number of components is known. For real-world datasets, however,

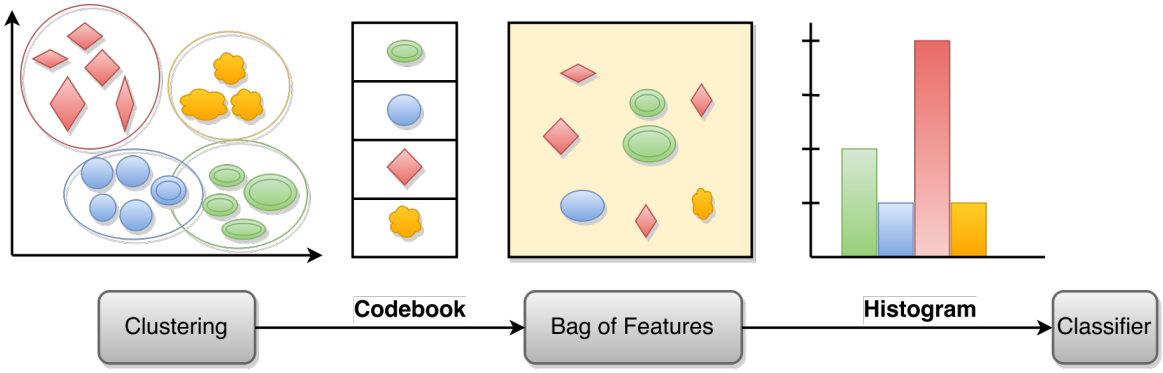


Figure 2.6. Bag of features.

that is usually not the case. Greedy EM method by Verbeek *et al.* [60] and the method by Figueiredo *et al.* [61] are two popular methods for GMM estimation.

2.4.2. Bag of Features

Bag of features (BOF) (or bag of visual words) model, originally called bag of words is a technique originally developed for text retrieval problem [62]. In computer vision, this technique is widely used [63] and in image domain the term "word" is replaced by "visual word".

The idea is to represent an instance with a distribution of local features. To do this, a procedure as illustrated in Figure 2.6 is followed. First, features are sampled from the training set and the feature space is divided into a number of clusters. K-means is a typical choice as the clustering method. The cluster centers then form a codebook referred as the "vocabulary". Once a new instance of data arrives, the local features are extracted and each feature is assigned to the closest cluster center in the codebook. By counting the number of occurrences of each codebook element, a histogram is computed for the input instance.

BOF modeling is extended to the video domain by using motion features extracted from the spatio-temporal volume as local descriptors. Typically, descriptors such as HOG, HOF and MBH are clustered separately and different histograms for each modality is calculated. The resulting histogram vectors are normalized so that

they are comparable in classification. ℓ_1 normalization is widely used since probability distributions add up to one; however, ℓ_2 normalization is also an option.

ℓ_1 normalization is achieved by dividing every element of a vector by its ℓ_1-norm (i.e. sum of the absolute values of vector elements). On the other hand, in ℓ_2 normalization, we divide every element of a vector by its ℓ_2-norm (i.e. the distance of the vector from origin). The formal definitions of ℓ_1- and ℓ_2-norms are done in Equations 2.16 and 2.17, respectively.

$$\|z\|_1 := \sum_{i=1}^{n} |z_i| \tag{2.16}$$

$$\|z\|_2 := \sqrt{\sum_{i=1}^{n} |z_i|^2} \tag{2.17}$$

When used in classifiers such as support vector machines, which will be detailed in Section 3.1, complex kernels suitable for histogram-based features give higher performance. χ^2 or histogram intersection kernels are examples for such kernels.

2.4.3. Fisher Vector

The notion of Fisher kernel [64] is not new, but applying Fisher vector (FV) encoding on visual vocabularies is proposed recently by Perronnin and Dance [56].

Fisher vector framework is analogous to the traditional bag of features representation. Instead of k-means clustering, Gaussian mixture models are used for quantizing local descriptors to be able to capture other statistics such as means and variances about the visual word distribution. BOF representation considers only 0-th order statistics, i.e. word frequencies. The FV encodes both first and second order statistics.

Given a generative visual word vocabulary, FV considers the direction in parameter space into which the vocabulary should be modified to better fit the data [55].

For this purpose, the gradient of the likelihood is encoded by applying derivative operations with respect to the distribution parameters of the vocabulary. FV stores the differences between pooled local features and dictionary items.

There are many advantages of using FV over other descriptor aggregation methods. Its power lies in combining both generative and discriminative approaches. Combined with simple linear classifiers, FV encoding yields significant performance.

An important advantage of FV over BOF is that much fewer vocabulary components are needed for satisfactory performance. Table 2.2 summarizes various aggregation methods mentioned in this section. It can easily be observed that the comparison between BOF and FV shows the superiority of FV.

In order to obtain the Fisher vector representation for a set of descriptors, we first train a K component GMM to learn the parameters $\lambda = \{\omega_i, \mu_i, \Sigma_i\}_{|_{i=1}^{K}}$ over training features. The covariance matrices are assumed to be diagonal.

In order both to decrease the size of the resulting FV and to decorrelate features (to support diagonal covariance assumption), one can apply principal components analysis (PCA) prior to building the dictionary.

Given a set of descriptors $X = \{x_1 ... x_N\}$, the following gradient vector can be defined in terms of a distribution p with parameters λ,

$$\nabla_\lambda \log p(X|\lambda). \tag{2.18}$$

In the case of visual vocabularies for action recognition, X can be a set of MBH descriptors or other. The distribution p is GMM. Hence, each component keeps information about frequency, mean and variation of a visual word. Under independence

assumption,

$$\mathcal{L}(X|\lambda) = \sum_{j=1}^{N} \log p(x_j|\lambda) \tag{2.19}$$

$$= \sum_{j=1}^{N} \log \left(\sum_{i=1}^{N} \omega_i p_i(x_j|\lambda) \right), \tag{2.20}$$

where $p_i(x_j|\lambda)$ is given by Equation 2.11.

The gradient of $\mathcal{L}$ with respect to μ and Σ parameters constitutes the Fisher vector encoding. The gradient with respect to ω is discarded since it brings little additional information. Normalized partial derivatives of means and deviations are approximated (see [56] for derivations) as follows:

$$\mathbf{u}_i = \frac{1}{N\sqrt{\omega_i}} \sum_{j=1}^{N} \gamma_{ij} \left(\frac{x_j - \mu_i}{\sigma_i} \right), \tag{2.21}$$

$$\mathbf{v}_i = \frac{1}{N\sqrt{2\omega_i}} \sum_{j=1}^{N} \gamma_{ij} \left[\left(\frac{x_j - \mu_k}{\sigma_i} \right)^2 - 1 \right], \tag{2.22}$$

where γ_{ij} denotes the posterior probability associating each vector x_j with a component i in the GMM and $\sigma_i^2 = \text{diag}(\Sigma_i)$. The FV becomes the concatenation of the vectors $\mathbf{u}_i$ and $\mathbf{v}_i$.

Since the covariances are diagonal, the final dimension of the FV becomes $2DK$, where D is the dimension of descriptors.

Perronnin *et al.* [38] proposed an improvement over the Fisher kernel and introduced power normalization on Fisher vectors. The vectors are first power-normalized and ℓ_2-normalized, so that linear classifiers can be used more successfully. Power nor-

Table 2.2. Typical attributes of aggregation methods. Note that some fields are not mandatory. (e.g. ℓ_2 normalization can be done on BOF; GMM can also be used for BOF; however, k-means is not sufficient for FV etc.)

	BOF	VLAD	FV
clustering	k-means	k-means	GMM(diag. cov.)
statistics	0th order	1st order	1st, 2nd order
basis	frequency	difference	difference
number of components	1000,...,4000	64,...,256	64,...,256
normalization	ℓ-1	power, ℓ-2	power, ℓ-2
dimensionality	K	D×K	2×D×K
kernel	χ^2, hist. intersection	linear	linear
preprocessing	-	PCA	PCA

malization is defined as follows:

$$f(z) = \text{sign}(z)|z|^{\alpha}, \tag{2.23}$$

where the parameter α is in the range $[0, 1]$. When $\alpha = 0.5$, which is a typical choice in most studies, the normalization becomes signed square-rooting.

2.4.4. Vector of Locally Aggregated Descriptors

A simplified version of the Fisher kernel is proposed by Jégou *et al.* [55] for efficient image search. In vector of locally aggregated descriptors (VLAD) representation, memory and computation requirements are less compared to BOF and FV. According to an evaluation [57], it is observed that, in terms of performance, FV and VLAD yield competitive results while both are outperforming BOF.

The mathematical formulation of a VLAD descriptor $v_{i,j}$ is as follows:

$$v_{i,j} = \sum_{x \text{ such that } \text{NN}(x)=c_i} x_j - c_{i,j}, \qquad i = 1...K \text{ and } j = 1...D, \tag{2.24}$$

where K is the number of clusters in k-means and D is the dimensionality of the local descriptor x. The vocabulary consists of cluster centers $\{c_1, ...c_K\}$. Each local descriptor x is assigned to its nearest neighbor (NN) cluster center. Different from BOF, instead of accumulating the frequencies of visual words, the differences between x and its nearest cluster center are kept. The resulting VLAD descriptor is the linearized version of $v_{i,j}$ and thus has dimensionality $K \times D$. In [57], ℓ_2 normalization is applied on VLAD descriptors.

The advantage of VLAD descriptors is their efficiency when used in large-scale image searches. The authors of [57] propose a joint optimization for dimensionality reduction and indexing. Their results show that compact representation can be achieved (i.e. low time complexity) with VLAD vectors while preserving high accuracy.

Having reviewed the technical background on video description using local visual features, we can now see how we use representative vectors for action classification in the next chapter.

3. ACTION CLASSIFICATION

Given video features and their corresponding action category labels for some training data, we can run a machine learning algorithm for the classification task. Among various classification algorithms, support vector machines (SVM) are widely used in the problem of action recognition from video. Extreme learning machine (ELM) is a much less popular method for action recognition. In this chapter, we will provide an overview of SVM and explore the basics of ELM.

3.1. Support Vector Machines

Support vector machines are first proposed by Cortes and Vapnik in 1995 [65] as a supervised learning algorithm used for classification and regression. The idea is to find a discriminant-based method without modeling densities generatively. This approach originates from Vapnik's principle to never solve a more complex problem as a first step before the actual problem [66].

In general, training instances of two classes are not linearly separable in their original finite dimensional space. Therefore, the idea in SVM is to map the vectors into a higher dimensional space to be able to simplify discrimination function. SVM model parameters are defined in terms of "support vectors". This term represents training data points which lie close to class boundaries.

The goal of SVM is to find a hyperplane in a high-dimensional space which maximizes the margin that separates two classes. Although SVM was first designed to be a two-class problem, now there are many variants in which multi-class classification can be achieved.

Non-linear classification can be performed with SVM by using the "kernel trick". Kernel functions enable to represent data points in high-dimensional feature spaces by using similarity measures.

SVM is a kernel-based algorithm. These algorithms are formulated as convex optimization problems; therefore there is a single optimum to be solved [67].

The formulation of the optimal hyperplane that separates two classes is as follows:

$$\begin{aligned} \mathbf{w}\mathbf{x}_i + w_0 \geq +1 \quad \text{if} \quad y_i = 1, \\ \mathbf{w}\mathbf{x}_i + w_0 \leq -1 \quad \text{if} \quad y_i = -1, \end{aligned} \tag{3.1}$$

where $X = \{\mathbf{x}_i, y_i\}_{|_{i=1}^{N}}$ defines the training features and labels correspondence. y_i takes the value $\{-1, +1\}$ depending on the class label.

We want to find $\mathbf{w}$ and w_0 such that the inequalities 3.1 hold for all X. These inequalities can be compactly written in the form:

$$y_i(\mathbf{w}\mathbf{x}_i + w_0) \geq 1. \tag{3.2}$$

The optimal hyperplane has maximum distance from the discriminant function to the closest instances. This distance defines the margin $\frac{y_i(\mathbf{w}\mathbf{x}_i + w_0)}{\|\mathbf{w}\|}$. We want to minimize $\|w\|$ in order to maximize this margin. Hence, the optimization becomes:

$$\min \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{s.t.} \quad y_i(\mathbf{w}\mathbf{x}_i + w_0) \geq 1, \forall i. \tag{3.3}$$

$\mathbf{w}$ and w_0 can be found directly with a standard quadratic optimization solution. The complexity depends on the dimensionality D of the point $\mathbf{x}$.

When the points are not linearly separable, which is generally the case, a so-called soft margin is defined:

$$y_i(\mathbf{w}\mathbf{x}_i + w_0) \geq 1 - \epsilon_i. \tag{3.4}$$

In this case, some of the support vectors may fall within the margin. The soft error $\sum_i \epsilon_i$ is added to the objective function as a penalty term:

$$\min \frac{1}{2}\|\mathbf{w}\|^2 + C_{SVM} \sum_i \epsilon_i, \tag{3.5}$$

where C_{SVM} is a hyperparameter of the model serving as a complexity parameter.

It can be shown that, using Lagrange multipliers α_i, the dual of this problem becomes:

$$\begin{aligned} \max_{\alpha_i} \quad & \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j, \\ \text{subject to} \quad & \sum_i \alpha_i y_i = 0 \quad \text{and} \quad 0 \leq \alpha_i \leq C_{SVM}, \ \forall i. \end{aligned} \tag{3.6}$$

When the dual is used instead of the primal, the complexity depends on the number of instances N instead of D.

If we define a mapping $\phi(\mathbf{x})$, the factor $\mathbf{x}_i^\top \mathbf{x}_j$ in Equation 3.6 is replaced by $\phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$. This dot product can be defined as a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ in the D-dimensional space without actually going to the mapped space of ϕ. Thus, non-linear kernel functions can be integrated to solve more complex problems. Usually, kernel functions are considered to measure similarity; therefore, any valid kernel function can be defined specific to the application.

Some commonly used kernels are:

(i) Linear kernel (i.e. no kernel)

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j. \tag{3.7}$$

(ii) Gaussian radial basis function (RBF) kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2). \tag{3.8}$$

(iii) χ^2 kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = 1 - \sum_{d=1}^{D} \frac{(\mathbf{x}_i^{(d)} - \mathbf{x}_j^{(d)})^2}{\frac{1}{2}(\mathbf{x}_i^{(d)} + \mathbf{x}_j^{(d)})}. \tag{3.9}$$

Many algorithms have been proposed to solve the optimization in Equation 3.6. Some popular implementations include liblinear [68] and libsvm [69]. The former decreases time complexity for the linear SVM by using an approximation. The latter has the option to use kernels. Both are iterative algorithms.

3.2. Extreme Learning Machines

Extreme learning machine is a classification and regression method proposed by Huang *et al.* [40, 70] as an extremely fast alternative to other conventional popular learning algorithms. In the literature, [71] and [72] used ELM for action classification from visual vocabularies (experimenting with 13- to 16-class problems), but the usage of ELM in this area is novel, and this study is the first application of ELM to large-scale action recognition with over 100 action classes.

In this section, we present methods based on ELM. The proposed algorithm works for the generalized single-hidden-layer feed-forward networks (SLFN), but the main difference is that the hidden layer in ELM need not be tuned, but is assumed to be known.

Figure 3.1 illustrates a SLFN. The mapping from the input layer to the hidden layer in traditional artificial neural networks is not known and tuned with backpropagation algorithms. In ELM, this mapping is known. Hence, the weights $\boldsymbol{\beta}$ which map hidden nodes to the output nodes can be solved analytically by least-squares solution.

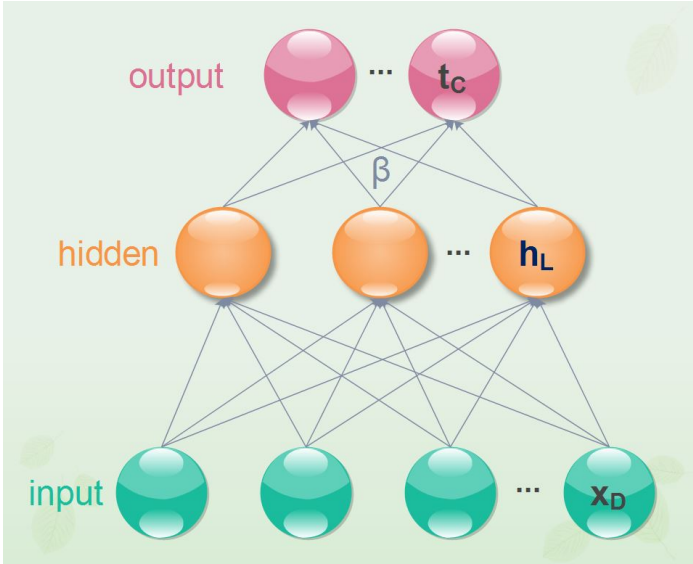


Figure 3.1. A single-hidden-layer feed-forward network. In Extreme Learning Machine, the mapping from input layer to hidden layer is known; therefore, the output weights $\boldsymbol{\beta}$ are solved without iterative tuning.

The formulation of C-class ELM classifier relies on an optimization problem with two objectives: minimizing the training error and minimizing the norm of the output weights for better generalization. Let $\boldsymbol{h}(\boldsymbol{x})$ be the feature mapping from the D-dimensional input node $\boldsymbol{x}$ to the L-dimensional hidden-layer feature space, and $\boldsymbol{\beta}_{(L\times C)}$ be the output weight matrix between the hidden layer of L nodes and the output nodes. Then, the optimization takes the form:

$$\min_{\beta} \|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|^2 \text{ and } \|\boldsymbol{\beta}\|, \tag{3.10}$$

where $\mathbf{T}_{(N\times C)}$ is the training label matrix of N training instances, and $\mathbf{H}_{(N\times L)}$ is the hidden-layer output matrix as follows:

$$\mathbf{H} = \begin{bmatrix} \boldsymbol{h}(\boldsymbol{x}_1) \\ \dots \\ \boldsymbol{h}(\boldsymbol{x}_N) \end{bmatrix} = \begin{bmatrix} h_1(x_1) & \cdots & h_L(x_1) \\ \vdots & \ddots & \vdots \\ h_1(x_N) & \dots & h_L(x_N) \end{bmatrix}. \tag{3.11}$$

Assuming $\mathbf{H}$ is known, the solution to this problem becomes:

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T}, \tag{3.12}$$

where $\mathbf{H}^{\dagger}$ is the *Moore-Penrose generalized inverse* of $\mathbf{H}$. The reader is referred to [70] for more detail.

3.2.1. Random projections

The initial version of ELM [70] assumes random projections for mapping to the hidden space. The motivation relies on the following theorems which are proven in [70]:

Theorem 3.1. *Given a standard SLFN with N hidden nodes and an activation function G which is infinitely differentiable in any interval, for N instances, for any $\boldsymbol{w}_i$ and b_i randomly chosen from any intervals according to any probability distribution, then the hidden layer output matrix $\boldsymbol{H}$ is invertible and $\|\boldsymbol{H\beta} - \boldsymbol{T}\| = 0$.*

Theorem 3.2. *Given a small value $\epsilon > 0$ and an activation function G which is infinitely differentiable in any interval, there exists $\hat{N} \leq N$ such that for N instances, for any $\boldsymbol{w}_i$ and b_i randomly chosen from any intervals according to any continuous probability distribution, then with probability one, $\|\boldsymbol{H}_{N\times\hat{N}}\boldsymbol{\beta}_{\hat{N}\times C} - \boldsymbol{T}_{N\times C}\| < \epsilon$.*

We randomly generate $\boldsymbol{w}$ and b from any continuous probability distribution so that the hidden layer output becomes,

$$\boldsymbol{h}(\boldsymbol{x}) = [G(\boldsymbol{w}_1, b_1, \boldsymbol{x}), ..., G(\boldsymbol{w}_L, b_L, \boldsymbol{x})], \tag{3.13}$$

where $G(\boldsymbol{w}, b, \boldsymbol{x})$ is a nonlinear piecewise continuous activation function such as:

(i) Sigmoid function

$$G(\boldsymbol{w}, b, \boldsymbol{x}) = \frac{1}{1 + \exp\left(-(\boldsymbol{wx} + b)\right)}. \tag{3.14}$$

(ii) Sine function

$$G(\boldsymbol{w}, b, \boldsymbol{x}) = \sin(\boldsymbol{w}\boldsymbol{x} + b). \tag{3.15}$$

(iii) Hard-limit function

$$G(\boldsymbol{w}, b, \boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{w}\boldsymbol{x} - b \geq 0, \\ 0, & \text{otherwise.} \end{cases} \tag{3.16}$$

(iv) Gaussian function

$$G(\boldsymbol{w}, b, \boldsymbol{x}) = \exp(-b\|\boldsymbol{x} - \boldsymbol{w}\|^2). \tag{3.17}$$

(v) Multiquadric function

$$G(\boldsymbol{w}, b, \boldsymbol{x}) = (\|\boldsymbol{x} - \boldsymbol{w}\|^2 + b^2)^{1/2}. \tag{3.18}$$

In feedforward neural networks, sigmoid and Gaussian functions are commonly used. In ELM, hard-limit and multiquadric functions also perform with good generalization [40].

3.2.2. Kernels

Similar to support vector machines, we can integrate kernels in extreme learning machines. Applying Mercer's condition on extreme learning machine, a kernel function can be defined as,

$$\boldsymbol{\Omega} = \mathbf{H}\mathbf{H}^\top, \tag{3.19}$$

where $\Omega_{i,j} = \boldsymbol{h}(\boldsymbol{x}_i).\boldsymbol{h}(\boldsymbol{x}_j) = K(\boldsymbol{x_i}, \boldsymbol{x_j})$.

We go to the kernel space directly without computing the feature mapping $\boldsymbol{h}(\boldsymbol{x})$. Hence, the hidden-layer output computation becomes a kernel operation. During training, we add a regularization coefficient C_{ELM} to the kernel matrix calculated with training data $\mathbf{X}_{(N\times D)} = [\boldsymbol{x_1} \dots \boldsymbol{x_N}]^T$ as follows:

$$\mathbf{H} = \left(\frac{\mathbf{I}}{C_{ELM}} + \boldsymbol{\Omega} \right). \tag{3.20}$$

One can plug in any valid kernel for $\boldsymbol{\Omega}$. For the case of linear kernel, feature mapping weights become the input features $\mathbf{X}$ themselves. Hence, the output of the kernel ELM classifier becomes:

$$\begin{aligned} f(x) &= K(\boldsymbol{x}, \mathbf{X})^\top \boldsymbol{\beta} \\ &= K(\boldsymbol{x}, \mathbf{X})^\top \left(\frac{\mathbf{I}}{C_{ELM}} + \boldsymbol{\Omega} \right)^\dagger \mathbf{T}. \end{aligned} \tag{3.21}$$

In order to obtain probability values or confidence scores for a classification output, we can use the inverse of sigmoid function to have the value between 0 and 1.

ELM implicitly allows training with multi-labels and multi-class classification. In our problem, we have datasets where an instance can be labeled with multiple actions. We can successfully use ELM without additional treatment for such cases.

4. PROPOSED METHODOLOGY

4.1. Motion Features

Our feature extraction, encoding and classification pipeline is illustrated in Figure 4.1. During training, we learn the coefficients of PCA, the parameters of GMM and the ELM model. Given a test video, we extract improved dense trajectory features and apply PCA projection on them. We employ Fisher vector encoding using the GMM for a global video description. Then, we concatenate some additional mid-level features. The final feature vector is given to ELM, and a confidence score for each action class is obtained.

The improved trajectory features are extracted using the software provided by Wang *et al.* [26]. These features include four aspects of video representation, namely motion boundary histograms (MBH), oriented histograms of flow (HOF), histogram of oriented gradients (HOG) and trajectory descriptors. With the default parameters, 192-dimensional MBH, 108-dimensional HOF and 96-dimensional HOG descriptors are extracted for each trajectory.

We use three local descriptors (MBH, HOF, HOG) and discard trajectory descriptors because of their inconsistent performance. Trajectory descriptors seem to help improving the performance in some datasets, but in our experiments they both lower the performance and their presence increases time complexity.

For training, we take a random subset of the local descriptors extracted from training videos. This procedure is a common practice because using all the descriptors for generating a codebook is computationally expensive and requires large amounts of memory. Moreover, most of these local descriptors are similar, especially the ones extracted from the same video. Random uniform sampling should not change the distribution we want to learn. Hence, training a GMM over a random subset of the training features is convenient.

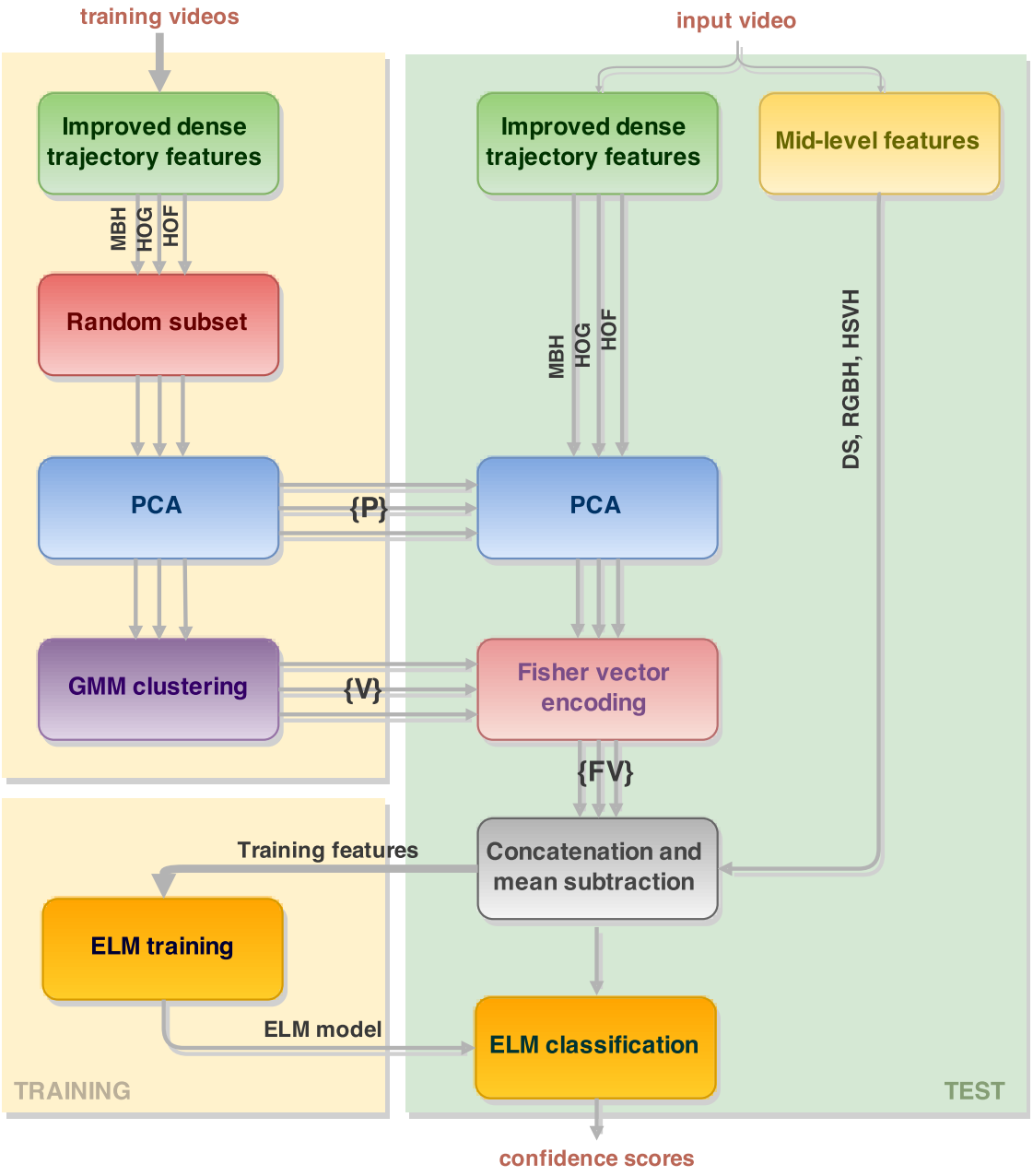


Figure 4.1. Proposed pipeline. Training part is ran once to get the PCA projection coefficients P= $\{P_{MBH}, P_{HOF}, P_{HOG}\}$, GMM vocabularies V= $\{V_{MBH}, V_{HOF}, V_{HOG}\}$ and ELM model.

Although MBH, HOF and HOG are all histogram based descriptors, they carry different information. Therefore; we learn a separate GMM for each aspect.

In order to both decrease FV size and to decorrelate the descriptors for diagonal covariance assumption, we apply PCA prior to building GMM vocabularies. Unless stated otherwise, we reduce each descriptor to 64 dimensions and learn a GMM with 64 components.

Given a video clip with a set of descriptors, we apply FV encoding for a global representation. Three different aspects yield three different Fisher vectors. We apply early fusion, but each FV is normalized separately before concatenation. The normalization is performed as in [38] by sign square-rooting (power normalization) followed by ℓ_2 normalization, so that linear classifiers can be used more successfully.

Besides local motion information, we extract some mid-level representations to obtain composite features to be used in classification. These mid-level features are explained in detail in the next section.

4.2. Mid-Level Features

We use semantic information about the actions in form of mid-level features, by considering several assumptions. The first assumption is that the area occupied by human body parts in a video frame provides complementary information for action recognition. For instance, UCF101 actions such as applying eye makeup, drying hair, brushing teeth and playing flute are much more likely to cover a close-up face compared to playing voleyball, rowing or skiing. Second, we assume that the color of the environment helps recognition of the action being performed. The dominant color in skiing and ice dancing videos is white, whereas soccer penalty videos show a lot of green pixels. These intermediate features can be used to bolster the classifiers based on low-level features. In Figure 4.2, the difference in color distributions between skiing and soccer penalty actions is illustrated. The results of several off-the-shelf body parts detectors are shown in Figure 4.3.

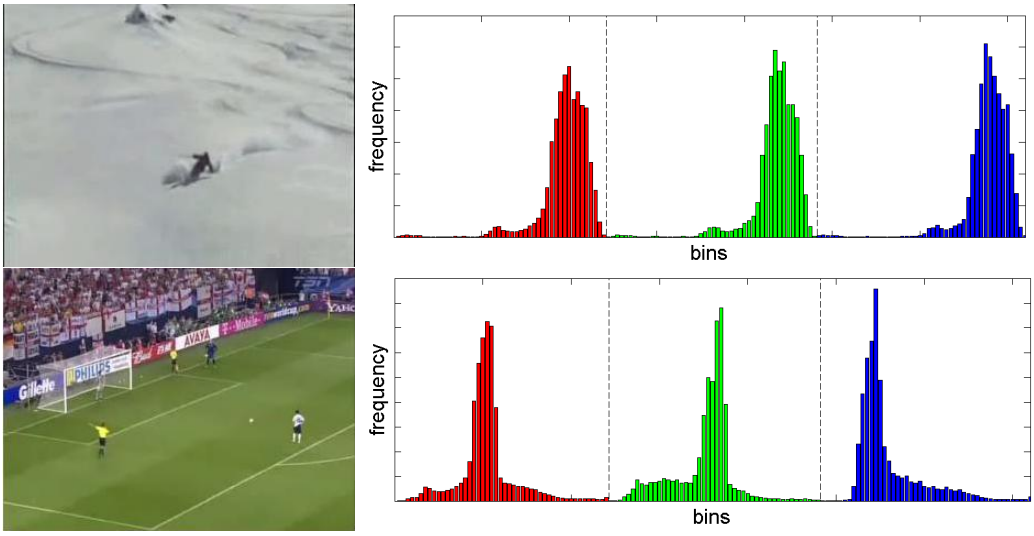


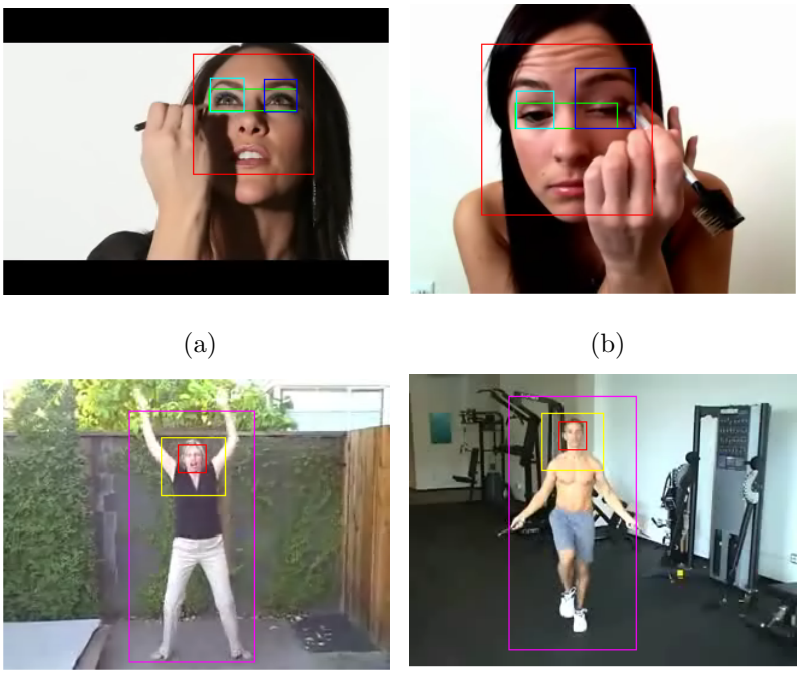
Figure 4.2. Sample frames from UCF101 dataset, and average RGB histograms for "skiing" and "soccer penalty" classes.

We apply seven object detectors on frames sampled from the videos. For each detector, ten equidistant frames of each video are used, and the detections are summarized with their statistics. Those detectors are the pre-trained face, upper body, eye pair, left eye, right eye and profile cascade object detectors [73], as well as a person detector [45]. We use minimum, maximum and mean area of the detected regions throughout the video. Additionally, we compute the frequency of the detections. We normalize each detection area by dividing it to the total frame area. Thus, we obtain four features per detector, which makes a mid-level feature vector of 28 dimensions.

We then compute normalized color histograms from ten frames on both RGB and HSV color spaces from each video. The histograms are then averaged over the frames to get a global feature vector of size 288 ($2 \times 48 \times 3$).

The final feature vector for a video becomes the concatenation of MBH, HOG and HOF Fisher vectors; the detector statistics (DS), and the two color histograms (RGBH, HSVH). Each feature dimension is centered on zero by subtracting the mean of the training features for data standardization purposes.

The mid-level features mentioned are not used in ChaLearn dataset, because



(a) (b)

(c) (d)

Figure 4.3. Body part detections on "apply eye makeup" (a)(b) and "jumping jack" (c)(d) classes of UCF101 dataset, with face (red), eye pair (green), left eye (blue), right eye (cyan), upper body (yellow), and people (magenta) detectors.

those videos are recorded in lab conditions, where the background is the same and stable. The context information provided with such mid-level features will not help discriminating actions.

4.3. Classification

Given the feature vector of a video clip, the multi-class ELM outputs the confidence scores for each action class depending on whether the video belongs to that category. The video is assigned to the action class with the highest confidence score.

Among other versions, we use ELM with linear kernel for its simplicity and high performance with Fisher vectors. The only hyperparameter is the regularization parameter C_{ELM}, which is optimized using a search over the validation set. The high speed of ELM training enables a quick search for a large number of possible C_{ELM} candidates.

It is possible to have multiple labels for an instance in ELM training. The label vector for each instance contains $+1$ for the presence and -1 for the absence of an action. In ChaLearn, two actions may occur at the same time such as "walk" and "clap". In these cases, we put $+1$ for both of these classes. Similarly, the validation set of THUMOS may have secondary action labels besides primary action labels. We observe that using secondary actions as positive actions helps increasing the performance.

In our experiments, we show why we have chosen kernel ELM over commonly used SVM. We also show empirically why we prefer linear kernels for Fisher vectors.

4.4. Localization

Temporal localization of actions, also referred as action detection or spotting, is the problem of predicting the boundaries of the action categories within a longer sequence of video. The task is more challenging than classifying a temporally trimmed video because it involves both action spotting and spotting.

The simplest possible approach for action localization is making frame-level decisions. In our case, that is to aggregate motion features into one FV for each frame. Frame-level decisions are usually noisy; therefore, the performance can easily be improved by post-processing such as smoothing, with the assumptions that neighbor frames have similar action categories and one cannot jump from an action to the other very rapidly. The problem with frame-level classification is that some actions can only be recognized when viewed within an interval.

A more clever alternative is to use window-level classification. Traditional detection methods use sliding window approach. In our work, we use fixed size sliding window for the action localization task. The window size is determined by computing the average duration of actions in the dataset. The stride of each window is one third of the window size. For instance in ChaLearn, we use a window size of 15 frames and a stride of 5 frames. Therefore; the windows have an overlap of 66%. We compute one FV for each window and pool their decisions for the final predictions.

The only dataset where we study localization is ChaLearn. In UCF101, the videos are already temporally trimmed. In THUMOS 2014, it is possible to trim the videos to better classify a video in an action category; however, it is computationally expensive and classifying the whole clip is already possible without localization.

In ChaLearn, the action performances contain pauses or irrelevant movements in between predefined action categories. Therefore; there is a need for an additional category, which we call "none". We train the classifier as if the problem was 12-class classification instead of 11 and sample from unannotated intervals to get training instances for this additional class.

5. EXPERIMENTS

5.1. Datasets

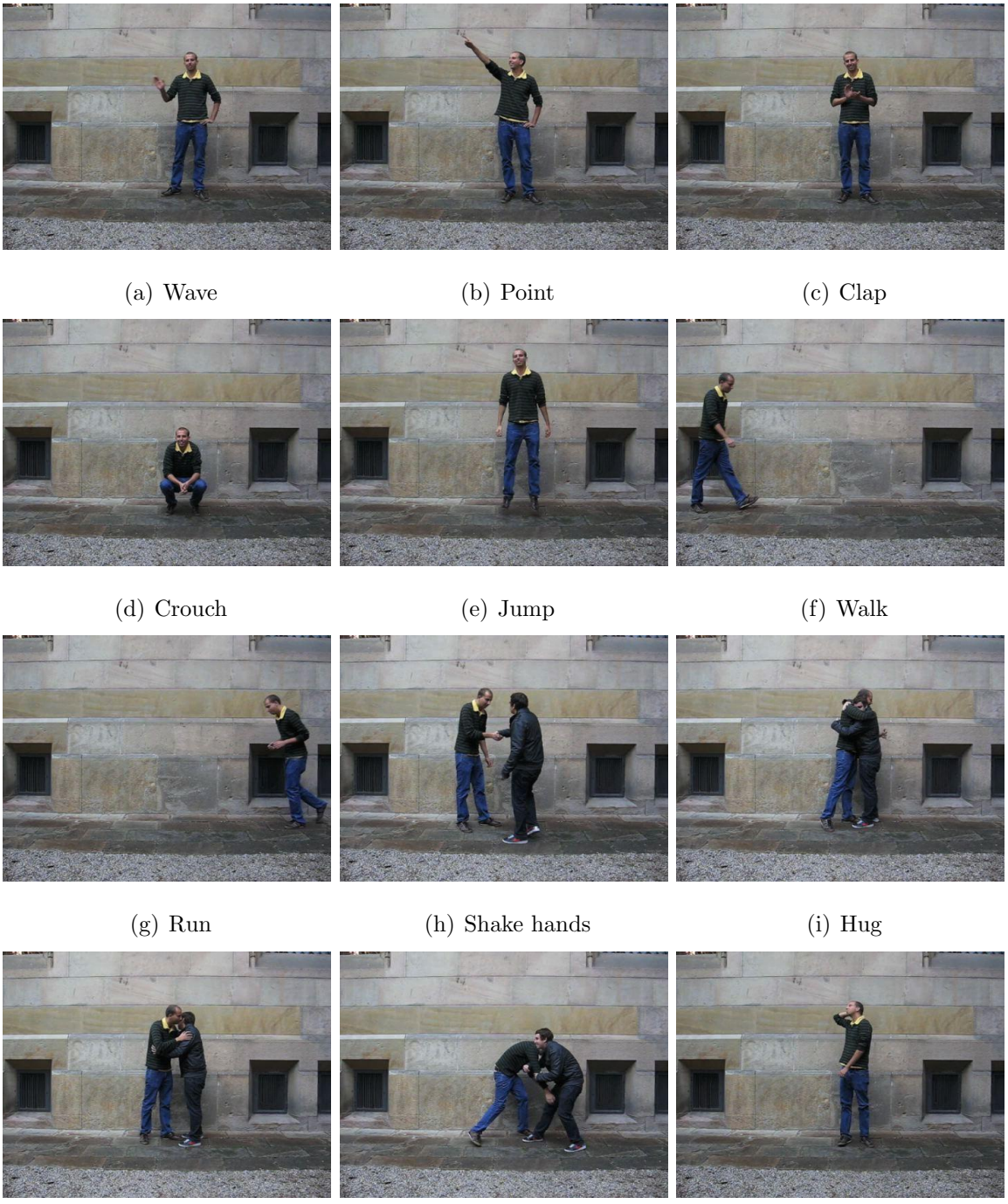
In this section, we present three datasets on which we carried experiments. These are ChaLearn LAP 2014 Track 2, UCF101 and THUMOS 2014 datasets. Action recognition challenges have been organized for all three datasets in ChaLearn LAP 2014, THUMOS 2013 and THUMOS 2014 workshops, respectively. The dataset for ChaLearn LAP 2015 Track 2 remained the same as in 2014, since the top performance is expected to improve.

5.1.1. ChaLearn Looking at People 2014 Track 2

ChaLearn Looking at People (LAP) 2014 Track 2 dataset [74,75] includes 9 RGB sequences divided into 5 training, 2 validation and 2 test videos. In total, there are 235 performances (135 in training, 44 in validation, 46 in test sets) of 11 action/interaction categories. Those are: wave, point, clap, crouch, jump, walk, run, shake hands, hug, kiss and fight. An overview of the dataset can be seen in Figure 5.1.

The resolution of the videos is 480×360 pixels and they are recorded at 15 frames per second. However, the dataset has a problem that one frame repeats itself once every five frames. We had to deal with fixing the videos and ground truth labels so that we don't have repeating frames. Each video lasts around 1 minute which makes a total of around 8,000 frames in the dataset.

The background of the recordings is not uniform but stable except a couple of times when the camera moves slightly. 17 subjects take part in the recordings of natural isolated and collaborative actions. The uncertainty of the number of performers in the scene makes the action recognition task more difficult. In order to introduce more challenge, a couple of occlusions are taking place while a subject is performing an action. A stranger may enter the scene and pass by in front of the performer.



(a) Wave (b) Point (c) Clap

(d) Crouch (e) Jump (f) Walk

(g) Run (h) Shake hands (i) Hug

(j) Kiss (k) Fight (l) NONE

Figure 5.1. Overview of the action classes in ChaLearn LAP 2014 Track 2 dataset.

The videos are recorded continuously while the actions are being performed one after the other without much pause in between. This requires a more complex action spotting system since there are not obvious boundaries between consecutive actions.

The standard evaluation protocol of the dataset is to report average Jaccard index (JI) over two test sequences: Sequence 5 and Sequence 7. Jaccard index is defined as,

$$JI_{s,c}(Y_{s,c}, \hat{Y}_{s,c}) = \frac{Y_{s,c} \cap \hat{Y}_{s,c}}{Y_{s,c} \cup \hat{Y}_{s,c}}, \tag{5.1}$$

where $Y_{s,c}$ is the ground truth label for sequence s and action class c. $\hat{Y}_{s,c}$ is the corresponding prediction labels. $Y_{s,c}$ and $\hat{Y}_{s,c}$ are binary vectors. The performance is reported as the mean over all Jaccard indices:

$$\frac{1}{sc} \sum_{s} \sum_{c} JI_{s,c}. \tag{5.2}$$

The best scoring methods in the challenge are summarized in [74]. The winner achieves 50.72% performance [76]. The first three ranking teams use improved trajectory features and support vector machines.

5.1.2. UCF101

UCF101 dataset is released in 2012 [20]. Since then, it is a widely adopted action and event recognition benchmark. In total, there are 13320 realistic action videos collected from Youtube, which correspond to 27 hours of video. The resolution of the videos is 25 frames per second and 320×240 pixels which can be considered as low resolution. The dataset consists of 101 action categories ranging from daily-life actions (e.g. "blow dry hair") to sports actions (e.g. "diving"). An overview of the large number of classes can be seen in Figure 5.2. The list of classes is given in Table 5.1. The 101 classes are divided into five types:

Table 5.1. List of classes in UCF101 and THUMOS 2014 datasets.

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

(i) Human-object interaction
(ii) Body-motion only
(iii) Human-human interaction
(iv) Playing musical instruments
(v) Sports

All the videos in the dataset are temporally trimmed (i.e. a video starts and ends with the same action without any irrelevant frames in between). For each category, there are 25 groups of video which may be similar in terms of background, viewpoint and performer. In each group, there are 4-7 videos.

The standard evaluation protocol on UCF101 is to report the mean accuracy over three different splits, which are provided by the dataset organizers.

Figure 5.2. Overview of UCF101 and THUMOS 2014 action classes. Figure adapted from [20].

The best score in the challenge was 85.9% [39]. They use improved trajectory features and support vector machines.

5.1.3. THUMOS 2014

The THUMOS 2014 dataset [22] is an extension to UCF101. The training set is based on the temporally trimmed videos from UCF101. Therefore, there are the same 101 action categories as in UCF101; plus a special "background" class that does not include any of the defined actions. In addition, new temporally untrimmed videos are collected again from Youtube. There are 1010 validation, 2500 background and 1574 test videos.

With a total of around 254 hours of video (25M frames), this dataset is currently one of the largest action recognition benchmarks. These open source videos typically involve low resolution samples with cluttered background and camera motion. Different from other benchmarks, the action performances are untrimmed, which makes the problem challenging.

In our study, we use the combination of training and validation sets as the training data and evaluate our performance on the test set, for which the labels have been sequestered during the challenge participation. In a given test video, our goal is to predict the presence of a target action class, together with a confidence score.

The standard evaluation protocol on THUMOS 2014 is to report the mean average

Table 5.2. Performances of descriptor types with mean average precision values and standard deviations among classes (THUMOS).

Feature set	mAP(%)	s.d.(%)	Worst and best recognized action classes(recognition rate)
HOG	47.78	23.88	Hammering(2.8) Archery(4.5) Haircut(4.7) Rowing(89.5) StillRings(90.1) MilitaryParade(96.5)
HOF	50.59	23.59	Haircut(4.2) SoccerPenalty(6.0) BaseballPitch(6.8) UnevenBars(88.1) MilitaryParade(96.5) StillRings(100)
MBH	53.17	22.54	SoccerPenalty(6.8) Haircut(8.4) BaseballPitch(8.5) HammerThrow(91.4) MilitaryParade(94.4) StillRings(100)
MBH+HOF	58.04	21.35	Haircut(6.3) SoccerPenalty(9.2) BaseballPitch(14.7) HammerThrow(93.2) MilitaryParade(95.9) StillRings(100)
HOG+HOF	60.04	21.37	Haircut(7.8) Archery(9.0) Hammering(12.7) BoxingSpeedBag(93.4) MilitaryParade(97.5) StillRings(100)
MBH+HOG	60.35	21.27	Haircut(9.5) SoccerPenalty(11.5) Archery(14.9) Skiing(91.6) MilitaryParade(98.42) StillRings(100)
MBH+HOG+HOF	**63.17**	20.02	Haircut(10.1) SoccerPenalty(12.8) Archery(14.9) HammerThrow(92.9) MilitaryParade(97.5) StillRings(100)

precision (mAP) over 102 classes.

$$mAP = \frac{1}{C}\sum_{c=1}^{C} AP(c), \tag{5.3}$$

where C is the number of classes. And average precision for each class is defined as:

$$AP(c) = \frac{\sum_{k=1}^{n}(P(k) \times rel(k))}{\sum_{k=1}^{n} rel(k)}. \tag{5.4}$$

Here, n test videos are sorted with descending scores and $P(k)$ is the precision at cut-off k of the list. $rel(k)$ equals to 1 if the video ranked k is a true positive and zero otherwise.

The best score in the challenge was 71% [77]. They use improved trajectory features, convolutional neural network features and support vector machines.

Table 5.3. Accuracy measures for descriptor types for each data split and their average (UCF101).

Feature set	Split 1 (%)	Split 2 (%)	Split 3 (%)	Overall (%)
Traj	51.92	53.05	54.22	53.06
HOG	70.08	69.63	69.97	69.89
HOF	71.35	73.78	74.70	73.28
MBH	76.16	77.00	76.22	76.46
MBH+HOF	79.17	80.88	81.01	80.35
HOG+HOF	81.13	82.54	82.25	81.97
MBH+HOG	82.34	82.91	81.98	82.41
MBH+HOG+HOF	83.45	84.79	84.47	84.24
MBH+HOG+HOF + Traj	83.61	85.59	84.28	**84.49**

5.2. Comparison of Descriptor Types

We performed several experiments to evaluate the effect of different steps of our pipeline. First, we examine the contribution of the individual aspects of the improved trajectory features MBH, HOG and HOF. The results on UCF101 and THUMOS 2014 datasets are presented in Tables 5.2 and 5.3, respectively. The best performances are obtained with the combination of the three aspects. As expected, the combination of a motion-based feature (MBH or HOF) and a spatio-based feature (HOG) yields higher performance than joining two motion-based features. Moreover, MBH is the most successful individual feature type in discriminating the action classes on these datasets due to its informativeness and robustness to camera motion.

In THUMOS 2014 dataset, among 102 categories, “haircut” appears to be the most difficult and “still rings” the easiest actions to recognize. The standard deviations are generally high, because of the difficult-to-distinguish classes (given in Table 5.2). These rates were among the lowest reported standard deviations in the challenge (See Figure 5.3). The two higher performances by Jain *et al.* [77] and by Oneata *et al.* [78] both used convolutional neural networks for additional features.

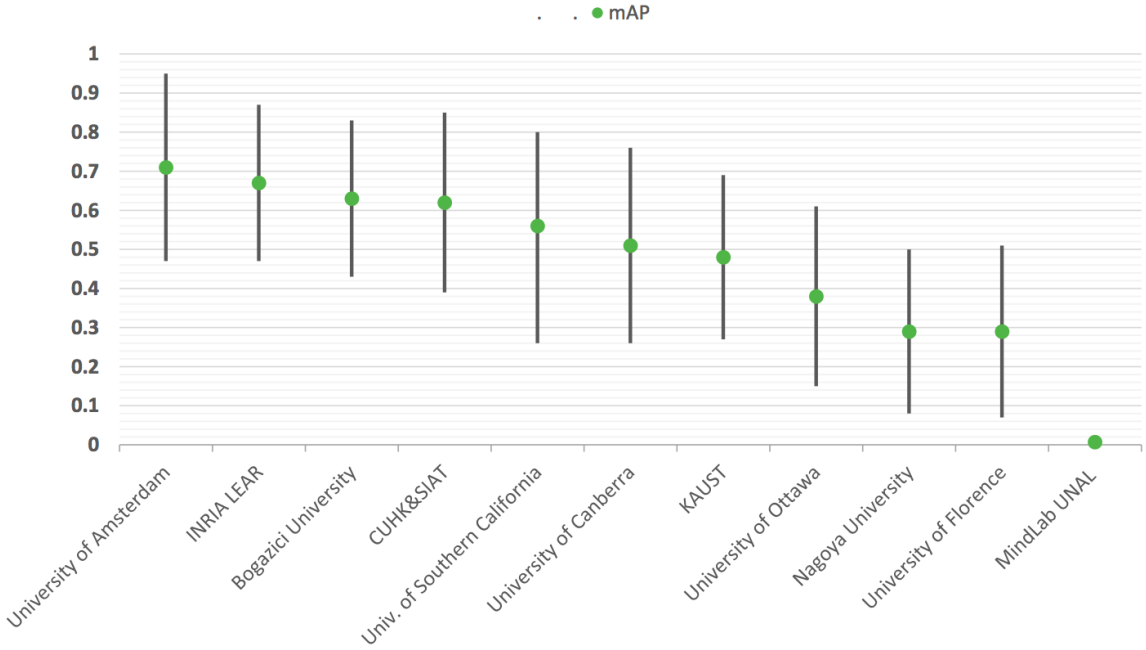


Figure 5.3. Challenge results (THUMOS).

In UCF101, the highest performance of THUMOS 2013 challenge was 85.9% by Wang and Schmid [39]. They used FV for MBH, HOG and HOF which were extracted with $K = 256$ number of GMM components. In addition, they divide the video into temporal and spatial blocks to later fuse the separate Fisher vectors from each block. In their paper [39], it is reported that if there is no division into blocks, the performance with SVM is 84.9% and the usage of trajectory descriptor even reduces the performance. In contrast, in our results the trajectory descriptor slightly improves the performance and it is very close to that of SVM with only $K = 64$.

5.3. Contribution of Mid-Level Features

In this section, we examine the use of the simple mid-level features described previously in Section 4.2, on THUMOS 2014 dataset. Obviously, these features are not sufficient alone to discriminate 102 classes, but they can provide complementary information, especially for some action classes. Table 5.4 summarizes the overall performance after the addition of several combinations from the mid-level features.

The concatenation of DS features to the Fisher vector did not result in a sig-

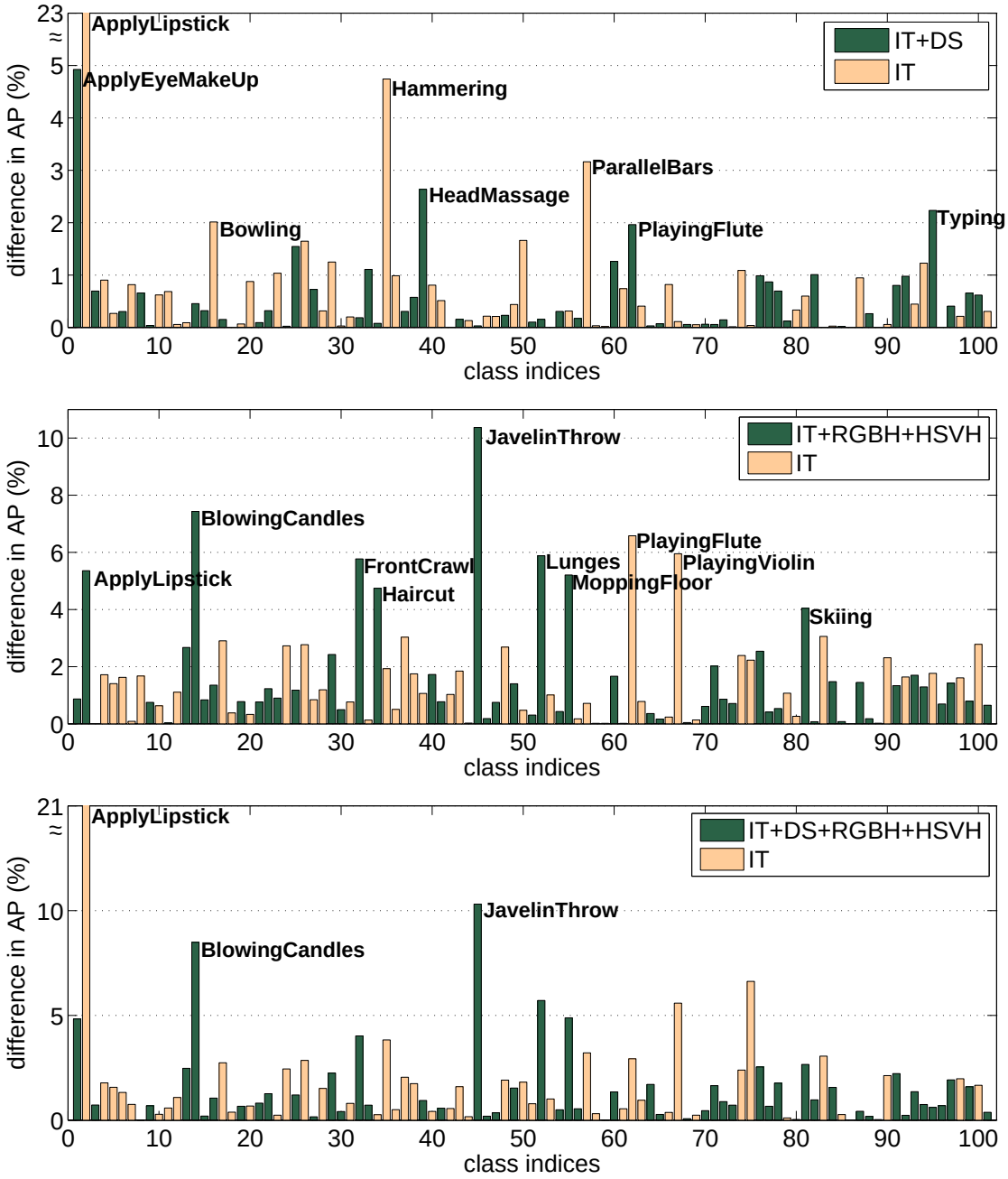


Figure 5.4. Contribution of mid-level features to improved trajectory features (MBH+HOG+HOF=IT). Green bars indicate improvement and yellow bars indicate decrease in performance after concatenation (THUMOS).

Table 5.4. Overall performances after mid-level feature concatenation (THUMOS).

Features	IT+DS	IT+RGB	IT+HSV	IT+HSVH+RGBH	IT+DS+RGBH+HSVH
mAP(%)	62.95	63.31	63.25	**63.37**	63.11

nificant increase or decrease in the overall performance; however, it can be seen from Figure 5.4 that the improvement on some classes are promising. We plot the difference in average precision for each class with and without concatenating mid-level feature sets to improved trajectory features. "Apply eye makeup" (index 1) is the action class where DS helped the most with a 5% increase, followed by "head massage" (index 39) with 2.6% and "playing flute" (index 62) with 2%. The mid-level attribute detects a face close-up, which helps identifying these classes.

One possible explanation of not having a significant difference in overall performance is that DS features have much lower dimensionality (28) than FV dimensionality ($2DK \times 3 = 2 \times 64 \times 64 \times 3 = 24576$). Moreover, these features are not guaranteed to be linearly separable. Separate kernels for each modalities can be summed up to solve this problem, but this remains as a future work.

The addition of RGBH and HSVH seems to consistently improve the performance. Although these two modalities are highly correlated, the combination is slightly higher than the invidual performances.

With the addition of color features, "skiing" (index 82) improves by 4%, the most difficult class "haircut" (index 34) by 5% and one of the two most confused classes "front crawl" (index 32) by 6%. All these improvements are absolute, for instance "javelin throw" (index 45) is improved from 32% to 42%.

5.4. Comparison of Different Encodings

We further made experiments with the BOF encoding and compared the results with FV representation. For BOF encoding, we use the codebook provided by the THUMOS 2014 challenge organizers. 4000 visual words are used for each modality

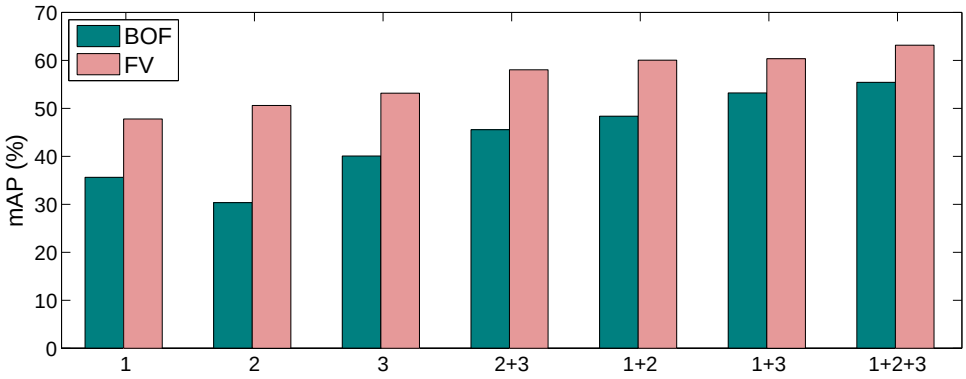


Figure 5.5. BOF and FV comparison on different combinations of descriptors from improved trajectory features (1: HOG, 2: HOF, 3: MBH), (THUMOS).

MBH, HOG, HOF separately. Each BOF vector is separately normalized first with ℓ_1, and then power normalization. The concatenation of the BOF vectors are used in ELM with RBF kernel. We use the RBF kernel since linear kernel results in much worse performance with histogram-based features.

With BOF representation, mean average precision of 55.41% is reached on THUMOS 2014 dataset. FV encoding with much fewer number of visual words (64) outperforms BOF encoding (4000 words) by 7.76% absolute. Figure 5.5 shows the consistent improvement of FV over BOF in different feature combinations.

Except for HOG descriptor, BOF encoding performances of different descriptor types and their combinations perform similarly as in FV encoding. The combination of MBH, HOF and HOG has the highest performance followed by the same order as in FV.

5.5. Comparison of ELM and SVM

We have compared ELM and SVM in terms of both training time and performance. It is clearly seen that using ELM is faster and more accurate.

The results on THUMOS 2014 are summarized in Table 5.5. These results are

Table 5.5. ELM and SVM comparison in terms of time and performance (THUMOS).

Algorithm	mAP(%)	Training time (sec) with 14330 samples	Testing time (sec) per sample
SVM (libsvm)	54.38	68064 (18.9 h)	0.917
SVM (liblinear)	54.02	4136 (1.2 h)	0.008
ELM	63.17	104	0.011

obtained by using the concatenation of Fisher vectors of MBH, HOF and HOG modalities. The experiments are carried out on a machine with two 2.26 GHz quad-core Intel Xeon processor and 32 GB of RAM. ELM yields a mean average precision of 63.17% with 104 seconds of training time whereas SVM performs 54.38% and 54.02% with 19 and 1.2 hours of training time when the algorithms of libsvm [69] and liblinear [68] are used, respectively. ELM outperforms SVM significantly in both aspects in this task.

The results on ChaLearn are similar. Since this is a smaller dataset, we are able to perform more extensive hyperparameter search with both SVM and ELM. As can be seen from Figure 5.6, although we span a large range for the cost parameter C_{SVM} of SVM, the highest performance of SVM (33.48%) remains below ELM's performance. ELM with linear kernel gives 35.18% Jaccard index at the highest point, but unless the regularization parameter C_{ELM} of ELM is too high, ELM is already above SVM's highest point for most of the points. It can be also observed that the variance of the ELM model is lower than that of SVM.

These performances on ChaLearn dataset are obtained when window-level classification is used with FV encoding of MBH, HOF and HOG. The number of GMM components was 128 and the dimensionality of each descriptor was reduced to 64 by PCA. In the next sections, we investigate the usage of frame-level and window-level classification and analyze the effects of FV parameters in more detail.

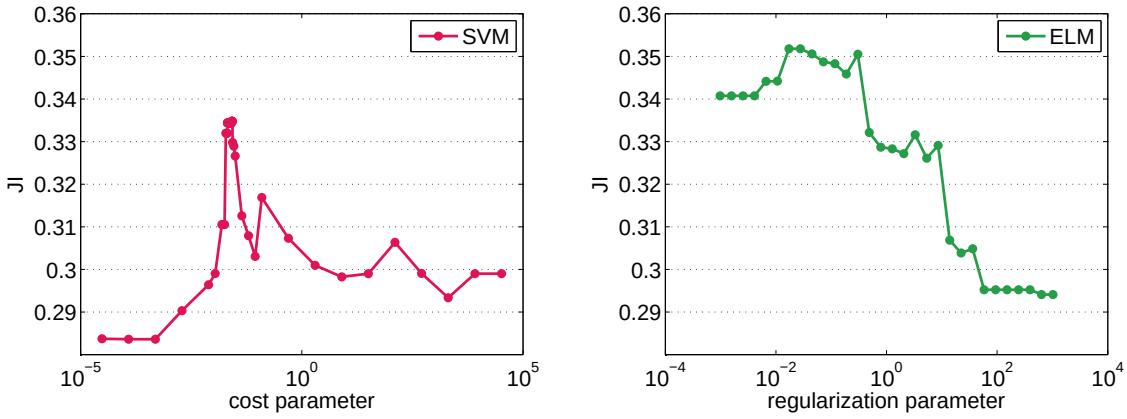


Figure 5.6. Left: The effect of cost parameter C_{SVM} in linear SVM. Right: The effect of regularization parameter C_{ELM} in linear ELM. Both x-axes are in logscale (ChaLearn).

5.6. Comparison of Frame-Level and Window-Level Classifications

We conducted some experiments to see the effect of frame-level and window-level classification for action spotting problem. The results reported in this section are on ChaLearn dataset.

Using improved trajectory features, we first apply frame-level aggregation to have one FV per frame. Given the action category for each frame, we train an ELM classifier using all the frames of the training and validation sets. Since there is a kernel operation involved, the memory requirements increase exponentially with the number of instances and are much higher than training a window-level classifier.

Figure 5.7 shows the classification outputs of the frame-level approach on the test set (Sequences 5 and 7) before and after post-processing. The classification outputs are very noisy and some filtering is required. The outputs are post-processed with median filtering operation of kernel size 11. The resulting classifications are much smoother, but the true positives of some rare classes are lost because of this operation.

The mean Jaccard index values for frame-level classification on both validation and test sets are examined with different FV parameters. Figure 5.8 presents the

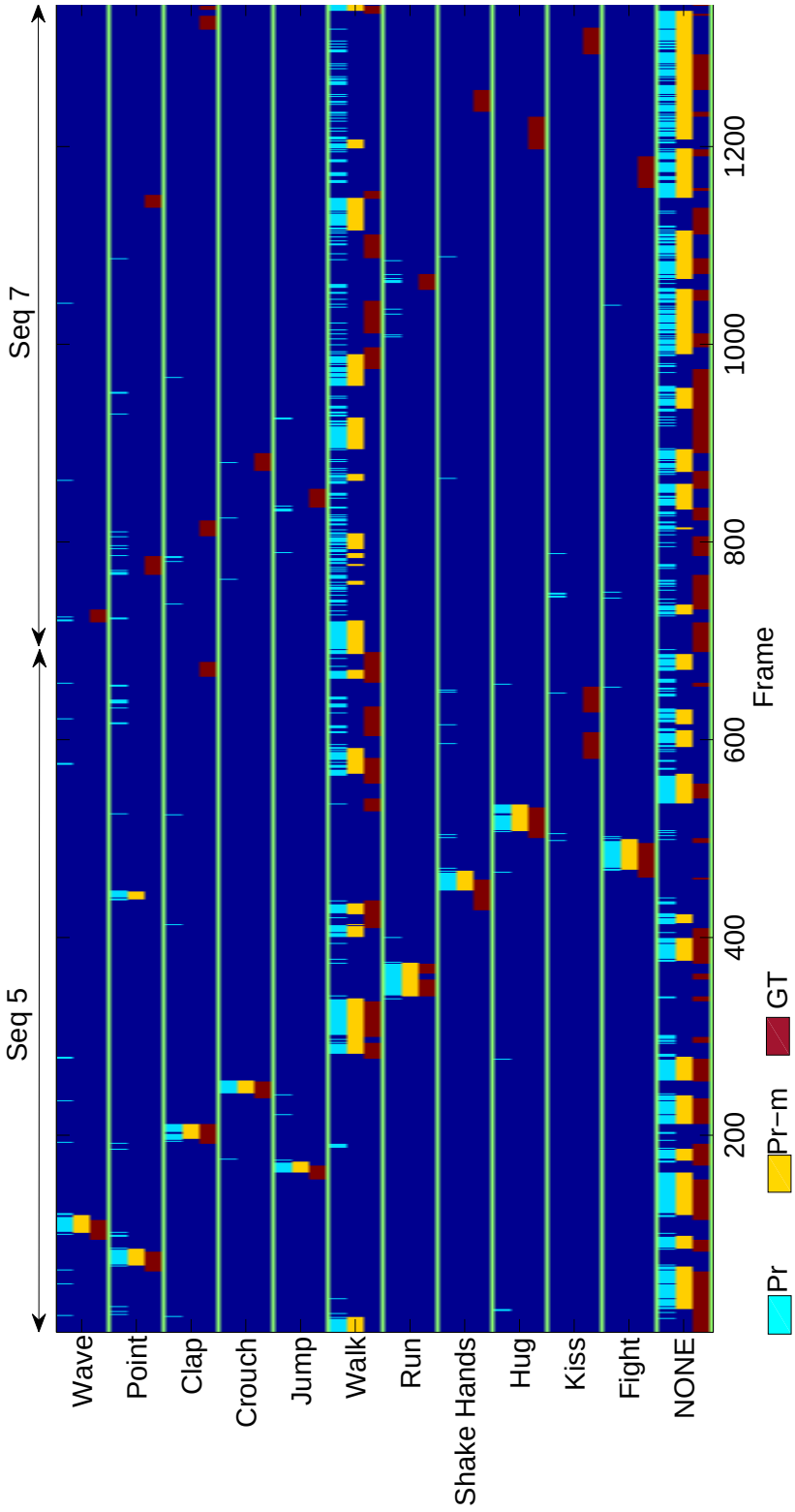


Figure 5.7. The effect of smoothing frame-level decisions. Predictions (Pr), median-filtered predictions (Pr-m), and ground truth (GT) are depicted for each frame in test set (ChaLearn).

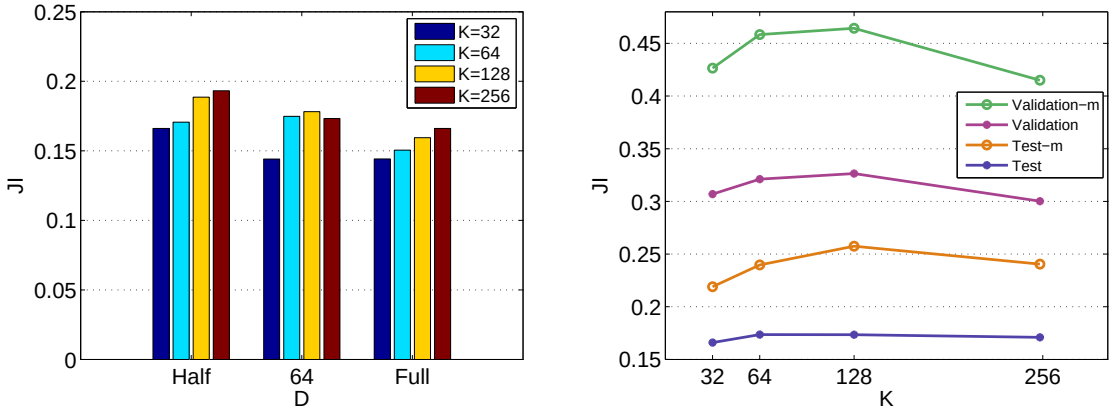


Figure 5.8. Left: The effect of dimensionality reduction by PCA. Right: The effect of number of Gaussian components K in FV encoding on validation and test sets, when median-filtered (denoted with -m) and not (ChaLearn).

results. Experiments on both sets show that post-processing provides a significant improvement for frame-level classifications. However; the performance on test set is nowhere close to the performance obtained by window-level classification. Validation performance is much higher than that of test set in general because the test set is more difficult. The effects of FV parameters will be discussed in the next section.

5.7. The Effect of Fisher Vector Parameters

We evaluated the FV representation by using different parameters for both D and K, which are descriptor dimensionality and the number of GMM components, respectively. Since UCF101 and THUMOS 2014 are large datasets, their feature extraction takes long time. Because of the limited resources, we conducted these experiments only on ChaLearn dataset.

We experimented with three different configurations for the dimensionality reduction. We either decrease each descriptor size to half (MBH=96, HOF=54, HOG=48), to 64 or keep the original dimensionality, but in all cases we project the descriptors in the decorrelated space computed by PCA.

Different than the experiments with K on the frame-level classification, we were

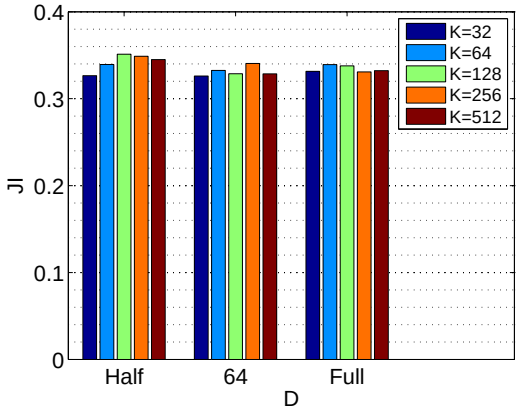


Figure 5.9. The effect of dimensionality reduction by PCA and the number of GMM components K (ChaLearn).

able to experiment with $K = 512$. The reason is that as the resulting dimensionality of FV rises, the memory requirements increase linearly. We were not able to perform frame-level classification on those Fisher vectors since the number of instances is already much higher than window-level approach. By going to $K = 512$, we expect to see a decrease on the performance due to high complexity.

Figure 5.9 summarizes the effect of D and K parameters in FV. Although there is no significant behavior in all values of D, there seems to be an increase in the performance while increasing K up to a point. The increase effect is more clear in Figure 5.8 (left) and the high complexity effect can be seen for $K = 512$ in Figure 5.9 as expected.

In terms of dimensionality reduction, reducing to half seems to perform slightly better than the other two. In total, reducing to half yields the lowest feature dimensionality among the three.

5.8. Model Selection in Extreme Learning Machine

In this section, we justify why we use linear ELM among other versions. We perform our experiments on the validation and test sets of THUMOS 2014. The effect of hyperparameters for ELM with linear kernel, RBF kernel and random projections

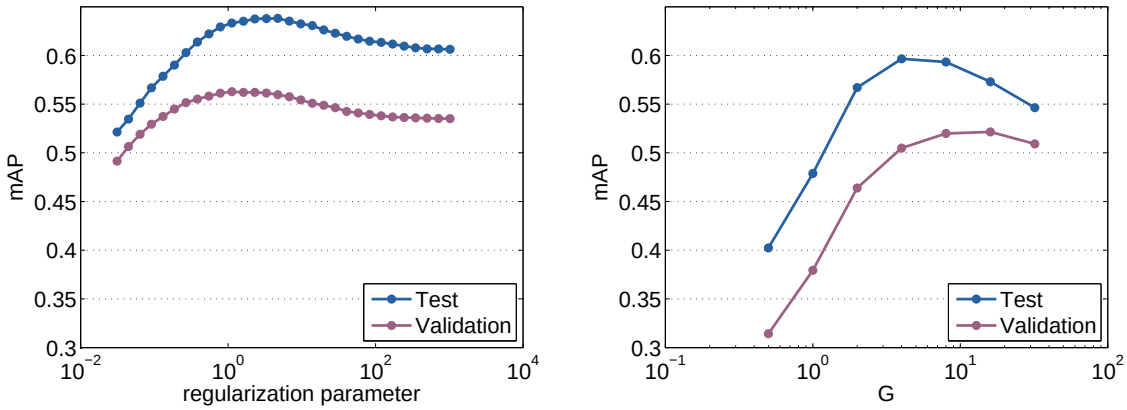


Figure 5.10. Kernel ELM. Left: The effect of regularization parameter C_{ELM} in linear ELM. Right: The effect of G parameter in RBF kernel ELM. Both x-axes are in logscale (THUMOS).

are examined.

Two kernel functions are selected for comparison. Those are linear kernel, which is expected to yield higher performance in combination with Fisher vector, and RBF kernel. The effect of regularization parameter C_{ELM} and the effect of RBF kernel parameter G can be seen in Figure 5.10.

The performances on validation and test sets move together, which shows the similarity between two sets. The mean average precision on the test set is higher because both training and validation videos are used for training and temporally untrimmed validation videos are more representative for the test set. However; training only on temporally trimmed videos is insufficient for validation performance.

The upper bound on RBF kernel ELM is still lower than mean average precision of linear ELM. Here, we see that linear ELM is both computationally efficient and more successful with FV features.

Besides kernels, we evaluated random projections, which were introduced with the initial version of ELM. The most crucial hyperparameter of random ELM is the number of hidden nodes L. This parameter determines the dimensionality of the space

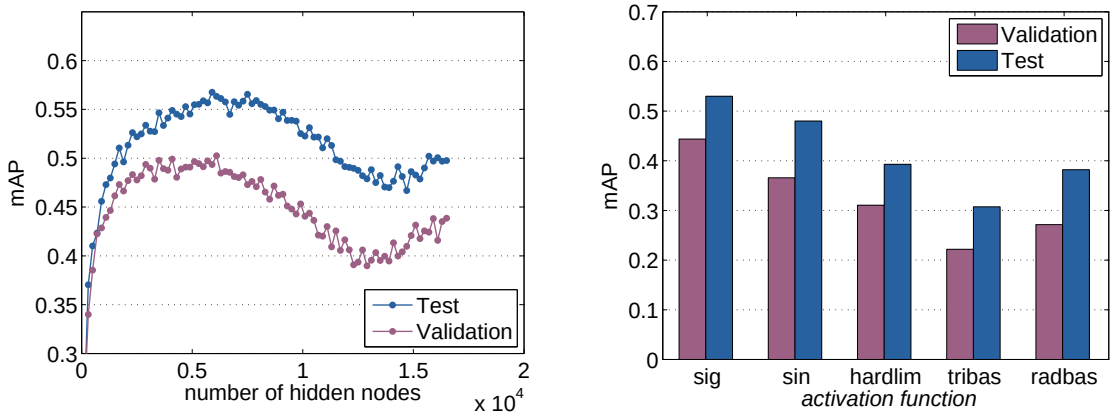


Figure 5.11. Random ELM. Left: The effect of number of hidden nodes L. Right: The effect of activation function (sigmoid, sine, hard-limit, triangular basis function, radial basis function) (THUMOS).

where the feature vectors are projected. We see from Figure 5.11 that the performance increases with the number of hidden nodes up to a point. It is surprising to see an increase after a certain point, but limitation of the memory resources made it impossible to test further points.

Another setting for the random ELM is the activation function option. We test with 5 different activation functions and see that sigmoid is suitable for our problem.

When Figures 5.10 and 5.11 are viewed, it can be seen that ELM with kernels outperform random projections consistently.

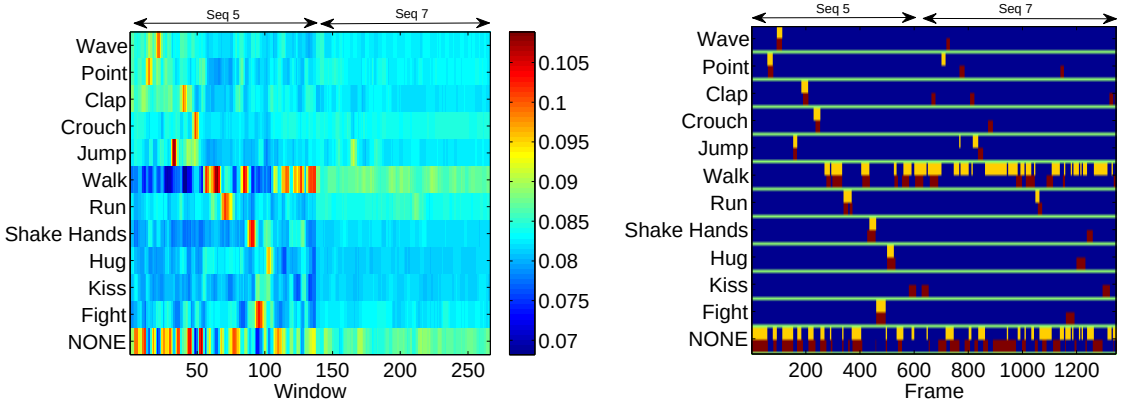


Figure 5.12. Left: Confidence scores. Right: Predictions (yellow) and ground truth (brown) (ChaLearn).

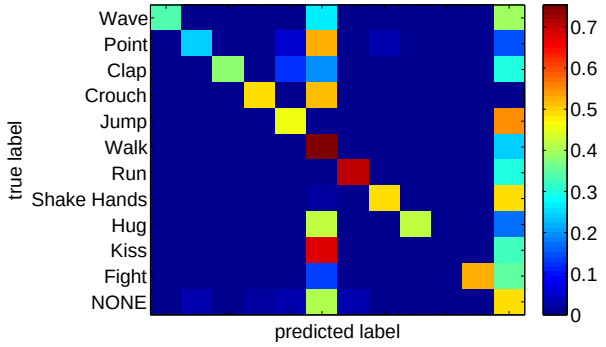


Figure 5.13. Left: Confusion matrix (ChaLearn).

5.9. Results

On ChaLearn LAP 2014 Track 2 dataset, we obtain 36.86% Jaccard index by using window-level classification and linear ELM with regularization parameter $C_{ELM} = 0.3$. We reduce each descriptor to half the size by PCA and use a GMM with 256 components. The normalized Fisher vectors for MBH, HOG and HOF are concatenated. The predictions and confidence scores are shown in Figure 5.12. Apparently, the second test video is much harder than the first.

The confusion matrix for ChaLearn dataset is presented in Figure 5.13. The null class is both hard to learn and it creates unbalance in the number of instances. Many false negatives are assigned to "walk" class and it is the other class which creates unbalance. Run class is relatively more successful.

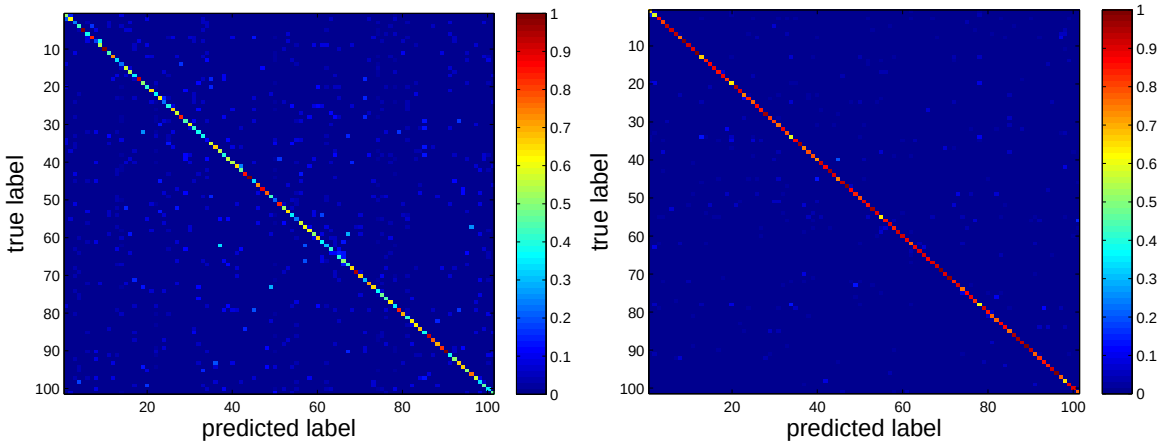


Figure 5.14. Confusion matrices for THUMOS 2014 (left) and UCF101 (right).

On THUMOS 2014 dataset, concatenating MBH, HOG, HOF, RGBH and HSVH features, we get our best result on the test set, which is 63.37% mean average precision. The confusion matrix for the classification with linear ELM is presented in Figure 5.14. Two most confused class pairs are "front crawling" and "breast stroke", which are two different swimming styles; and "drumming" and "playing tabla", which are two similar instruments.

On UCF101 dataset, we get an overall accuracy of 84.49% by using FV representation of MBH, HOG, HOF and trajectory descriptors. The confusion matrix for the classification with linear ELM is presented in Figure 5.14. This performance is much higher than that of THUMOS 2014 because the problem is easier. Training set is more representative of the test set since all the videos are pre-segmented in UCF. There are no irrelevant frames within a video.

Our results show that our pipeline yields competitive performances in recent challenging datasets. In the next chapter, we conclude our work and state possible improvements that would help increase the performance in action and event recognition from video.

6. CONCLUSION

In this study, we applied a novel methodology to recent and very challenging action recognition datasets. We extract improved trajectory features and MBH, HOG, HOF descriptors along trajectories. We apply separate Fisher vector encoding for each descriptor type and use extreme learning machines with linear kernel for multi-class classification.

We achieve 84.49% and 63.37% recognition performances with our approach on 101- and 102- class action classification problems, and 36.86% on an 11-class action detection problem on UCF101, THUMOS 2014 and ChaLearn LAP 2014 Track 2 datasets, respectively.

Our experiments involve analysis of various model parameters and methods, as well as comparative investigations with well-known approaches. Each building block of our pipeline is examined in detail with explanations to the results.

6.1. Remarks

Among descriptor types in improved trajectory features, motion boundary histograms are more successful in action representation, since they are more robust to camera motion. The combination of all descriptor types yields the best results, covering both motion and appearance information.

Mid-level features that we propose are not usable alone, but they carry complementary information. We apply some pre-trained detectors for face, upper body, eye pair, left eye, right eye and profile, as well as HOG person detector on several frames of each video. This feature extraction about body parts is simple and should be improved to contribute to the overall performance. Colors histograms averaged over a video can help action recognition by providing context-related information of the scene.

Fisher vector encoding consistently outperforms bag of features encoding for all three types of improved trajectory descriptors. Since FV uses Gaussian mixture model clustering, it encodes first and second order statistics of the set of descriptors. On the other hand, BOF uses k-means; therefore, considers only zero-order statistics. Fisher vector representation can be successfully used with extreme learning machines with linear kernels.

Extreme learning machines outperform support vector machines in our experiments. Our results show that ELM is a viable alternative to the commonly used SVM, especially with respect to the reduced training times. For limited training resources, ELM makes it possible to evaluate more feature combinations compared to SVM, which typically requires extensive parameter optimization [79].

In the context of action spotting, window-level action classification is more favorable than frame-level classification in terms of both reduced number of instances and performance. Window-level classification provides implicit incorporation of temporal information, whereas if frame-level classification is used, one should apply post-processing on the predictions to improve the results. A post-processing step, such as median filtering, takes into account the assumption that the actions in neighboring frames would not have abrupt changes.

In Fisher vector encoding, the reduced dimensionality D of local descriptors and the number of Gaussian mixture model components K both determine the complexity of the system. Increasing these parameters after a certain value results in suffering from curse of dimensionality. Hence, these parameters should be determined depending on the scale of the problem. However, our experiments show that the performance does not change significantly provided these parameters are set in a reasonable range, because Fisher vector encoding, the improvement of the estimation of the underlying dictionary is not crucial.

Extreme learning machines with kernels outperform random projections on our task. Sigmoid is a convenient choice for nonlinear activation function in ELM with

random projections. The regularization parameter in kernel ELM and the number of hidden nodes in random ELM are complexity parameters, which should be optimized upon validation.

6.2. Future Work

Possible improvements include a more sophisticated body parts tracker and additional treatment of the mid-level features before simple concatenation to the FV. These features are different in nature and may not be directly compatible with FV for feature-level fusion.

In a future work, recently popular convolutional neural network features can be used as appearance features and can be combined with the motion features. ImageNet [80] dataset has also some classes in common with UCF101. These data can be used for cross-dataset experiments.

The THUMOS 2014 challenge setup contains background videos, which are not from any of the 101 classes. In a later study, such videos can be used to train an improved garbage class detector, as well as for temporal segmentation of actions.

To deal with the rare classes, the number of training instances can be increased by sampling sub-windows from training intervals. ChaLearn is an unbalanced dataset in terms of class distributions; therefore, it is important to avoid one class dominating others. A way to achieve this is to increase the training samples of the rare classes.

One problem with ChaLearn dataset may be that some action classes may have a performance duration less than the tracking time for trajectories. In this case, actions represented by shorter trajectories cannot be learned properly. This is observed when predictions are visualized over the test videos. To solve this issue, the number of frames to track each point should be set upon optimization on validation videos.

Compared to UCF101 and THUMOS 2014, ChaLearn is simpler in terms of the

video content. We should be able to successfully detect and track humans in ChaLearn. A possible future direction would be modeling the background and capturing foreground objects. Hence, more specific features, which may not generalize for large-scale problems, can be extracted for this problem.

We conclude with the remark that ELM classifier can be improved by ensemble methods such as using multiple subsets of training in order to avoid over-fitting. Additionally, instead of minimizing the squared loss over the whole training set, a relaxation of the cost function that utilizes a soft-margin like support vector machines can be explored.

REFERENCES

1. Laptev, I., M. Marszalek, C. Schmid and B. Rozenfeld, "Learning Realistic Human Actions from Movies", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8, 2008.

2. Chaaraoui, A. A., P. Climent-Pérez and F. Flórez-Revuelta, "A Review on Vision Techniques Applied to Human Behaviour Analysis for Ambient-assisted Living", *Expert Systems with Applications*, Vol. 39, No. 12, pp. 10873–10888, 2012.

3. Khosla, A., R. Hamid, C.-J. Lin and N. Sundaresan, "Large-Scale Video Summarization Using Web-Image Priors", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

4. Shapovalova, N., C. Fernández, F. Roca and J. Gonzàlez, "Semantics of Human Behavior in Image Sequences", A. A. Salah and T. Gevers (Editors), *Computer Analysis of Human Behavior*, pp. 151–182, Springer London, 2011.

5. Delaitre, V., I. Laptev and J. Sivic, "Recognizing Human Actions in Still Images: A Study of Bag-of-Features and Part-based Representations", *British Machine Vision Conference (BMVC)*, 2010.

6. Yao, B., X. Jiang, A. Khosla, A. L. Lin, L. J. Guibas and L. Fei-Fei, "Action Recognition by Learning Bases of Action Attributes and Parts", *International Conference on Computer Vision (ICCV)*, 2011.

7. Raja, K., I. Laptev, P. Pérez and L. Oisel, "Joint Pose Estimation and Action Recognition in Image Graphs.", *ICIP*, pp. 25–28, IEEE, 2011.

8. Sadanand, S. and J. J. Corso, "Action Bank: A High-Level Representation of Activity in Video", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

9. Park, A.-Y. and S.-W. Lee, "Gesture Spotting in Continuous Whole Body Action Sequences Using Discrete Hidden Markov Models", *Gesture in Human-Computer Interaction and Simulation*, Vol. 3881 of *Lecture Notes in Computer Science*, pp. 100–111, Springer Berlin Heidelberg, 2006.

10. Derpanis, K., M. Sizintsev, K. Cannons and R. Wildes, "Efficient Action Spotting Based on a Spacetime Oriented Structure Representation", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1990–1997, 2010.

11. Yoon, S. M. and A. Kuijper, "Human Action Recognition Based on Skeleton Splitting", *Expert Systems with Applications*, Vol. 40, No. 17, pp. 6848–6855, 2013.

12. Schuldt, C., I. Laptev and B. Caputo, "Recognizing Human Actions: A Local SVM Approach", *International Conference on Pattern Recognition (ICPR)*, Vol. 3 of *ICPR '04*, pp. 32–36, IEEE Computer Society, 2004.

13. Blank, M., L. Gorelick, E. Shechtman, M. Irani and R. Basri, "Actions as Space-Time Shapes", *IEEE International Conference on Computer Vision (ICCV)*, pp. 1395–1402, 2005.

14. Rodriguez, M., J. Ahmed and M. Shah, "Action MACH A Spatio-Temporal Maximum Average Correlation Height Filter for Action Recognition", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8, 2008.

15. Liu, J., Y. Yang and M. Shah, "Learning Semantic Visual Vocabularies Using Diffusion Distance", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 461–468, 2009.

16. Marszalek, M., I. Laptev and C. Schmid, "Actions in Context", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2929–2936, 2009.

17. Reddy, K. and M. Shah, "Recognizing 50 Human Action Categories of Web Videos", *Machine Vision and Applications*, Vol. 24, No. 5, pp. 971–981, 2013.

18. Kuehne, H., H. Jhuang, E. Garrote, T. Poggio and T. Serre, "HMDB: A Large Video Database for Human Motion Recognition", *IEEE International Conference on Computer Vision (ICCV)*, 2011.

19. Kliper-Gross, O., T. Hassner and L. Wolf, "The Action Similarity Labeling Challenge", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 34, No. 3, pp. 615–621, 2012.

20. Soomro, K., A. Roshan Zamir and M. Shah, "UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild", *CRCV-TR-12-01*, 2012.

21. Strassel, S., A. Morris, J. Fiscus, C. Caruso, H. Lee, P. Over, J. Fiumara, B. Shaw, B. Antonishek and M. Michel, "Creating HAVIC: Heterogeneous Audio Visual Internet Collection", *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, European Language Resources Association (ELRA), 2012.

22. Jiang, Y.-G., J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah and R. Sukthankar, "THUMOS Challenge: Action Recognition with a Large Number of Classes", `http://crcv.ucf.edu/THUMOS14/`, 2014.

23. Chaquet, J. M., E. J. Carmona and A. Fernández-Caballero, "A Survey of Video Datasets for Human Action and Activity Recognition", *Computer Vision and Image Understanding*, Vol. 117, No. 6, pp. 633 – 659, 2013.

24. Laptev, I., "On Space-Time Interest Points", *International Journal of Computer Vision*, Vol. 64, No. 2-3, pp. 107–123, 2005.

25. Wang, H., A. Kläser, C. Schmid and C.-L. Liu, "Dense Trajectories and Motion Boundary Descriptors for Action Recognition", *International Journal of Computer Vision*, Vol. 103, No. 1, pp. 60–79, 2013.

26. Wang, H. and C. Schmid, "Action Recognition with Improved Trajectories", *IEEE*

International Conference on Computer Vision (ICCV), 2013.

27. Kläser, A., M. Marszałek and C. Schmid, "A Spatio-Temporal Descriptor Based on 3D-Gradients", *British Machine Vision Conference (BMVC)*, pp. 995–1004, 2008.

28. Scovanner, P., S. Ali and M. Shah, "A 3-Dimensional SIFT Descriptor and Its Application to Action Recognition", *ACM International Conference on Multimedia*, pp. 357–360, 2007.

29. Dalal, N., B. Triggs and C. Schmid, "Human Detection Using Oriented Histograms of Flow and Appearance", *European Conference on Computer Vision (ECCV)*, Vol. 3952 of *Lecture Notes in Computer Science*, pp. 428–441, Springer Berlin Heidelberg, 2006.

30. Liu, J., B. Kuipers and S. Savarese, "Recognizing Human Actions by Attributes", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3337–3344, 2011.

31. Zhu, J., B. Wang, X. Yang, W. Zhang and Z. Tu, "Action Recognition with Actons", *IEEE International Conference on Computer Vision (ICCV)*, 2013.

32. Wang, L., Y. Qiao and X. Tang, "Motionlets: Mid-level 3D Parts for Human Motion Recognition", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2674–2681, 2013.

33. Wang, H., M. M. Ullah, A. Kläser, I. Laptev and C. Schmid, "Evaluation of Local Spatio-Temporal Features for Action Recognition", *British Machine Vision Conference (BMVC)*, 2009.

34. Shugao Ma, N. I.-C., Jianming Zhang and S. Sclaroff, "Action Recognition and Localization by Hierarchical Space-Time Segments", *IEEE International Conference on Computer Vision (ICCV)*, 2013.

35. Oneata, D., J. Verbeek and C. Schmid, "Action and Event Recognition with Fisher Vectors on a Compact Feature Set", *IEEE International Conference on Computer Vision (ICCV)*, pp. 1817–1824, 2013.

36. Peng, X., C. Zou, Y. Qiao and Q. Peng, "Action Recognition with Stacked Fisher Vectors", *European Conference on Computer Vision (ECCV)*, Vol. 8693 of *Lecture Notes in Computer Science*, pp. 581–595, Springer International Publishing, 2014.

37. Yang, X. and Y. Tian, "Action Recognition Using Super Sparse Coding Vector with Spatio-Temporal Awareness", *Europen Conference on Computer Vision (ECCV)*, Vol. 8690 of *Lecture Notes in Computer Science*, pp. 727–741, Springer International Publishing, 2014.

38. Perronnin, F., J. Sánchez and T. Mensink, "Improving the Fisher Kernel for Large-Scale Image Classification", *European Conference on Computer Vision (ECCV)*, 2010.

39. Wang, H. and C. Schmid, "LEAR-INRIA Submission for the THUMOS Workshop", *ICCV Workshop on Action Recognition with a Large Number of Classes*, 2013.

40. Huang, G.-B., H. Zhou, X. Ding and R. Zhang, "Extreme Learning Machine for Regression and Multiclass Classification", *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, Vol. 42, No. 2, pp. 513–529, 2012.

41. Varol, G. and A. A. Salah, "Extreme Learning Machine for Large-Scale Action Recognition", *ECCV Workshop on Action Recognition with a Large Number of Classes*, 2014.

42. Varol, G. and A. A. Salah, "Extreme Learning Machine for Large-Scale Action Recognition on Temporally Untrimmed Videos", *submitted for publication*.

43. Shi, F., E. Petriu and R. Laganiere, "Sampling Strategies for Real-Time Action

Recognition", *IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 2595–2602, 2013.

44. Isard, M. and A. Blake, "CONDENSATION—Conditional Density Propagation for Visual Tracking", *International Journal of Computer Vision*, Vol. 29, No. 1, pp. 5–28, 1998.

45. Dalal, N. and B. Triggs, "Histograms of Oriented Gradients for Human Detection", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 886–893, 2005.

46. Lucas, B. D. and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision", *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, Vol. 2, pp. 674–679, Morgan Kaufmann Publishers Inc., 1981.

47. Shi, J. and C. Tomasi, "Good Features to Track", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 593–600, 1994.

48. Farnebäck, G., "Two-Frame Motion Estimation Based on Polynomial Expansion", *Image Analysis*, Vol. 2749 of *Lecture Notes in Computer Science*, pp. 363–370, Springer Berlin Heidelberg, 2003.

49. Bradski, G., "The OpenCV Library", *Dr. Dobb's Journal of Software Tools*, 2000.

50. Brox, T. and J. Malik, "Large Displacement Optical Flow: Descriptor Matching in Variational Motion Estimation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 33, No. 3, pp. 500–513, 2011.

51. Jiang, Y.-G., Q. Dai, X. Xue, W. Liu and C.-W. Ngo, "Trajectory-Based Modeling of Human Actions with Motion Reference Points", *European Conference on Computer Vision (ECCV)*, Vol. 7576 of *Lecture Notes in Computer Science*, pp. 425–438, 2012.

52. Tamrakar, A., S. Ali, Q. Yu, J. Liu, O. Javed, A. Divakaran, H. Cheng and H. Sawhney, "Evaluation of Low-Level Features and Their Combinations for Complex Event Detection in Open Source Videos", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3681–3688, 2012.

53. Bay, H., A. Ess, T. Tuytelaars and L. V. Gool, "Speeded-Up Robust Features (SURF)", *Computer Vision and Image Understanding*, Vol. 110, No. 3, pp. 346 – 359, 2008.

54. Fischler, M. A. and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", *Commun. ACM*, Vol. 24, No. 6, pp. 381–395, 1981.

55. Jégou, H., M. Douze, C. Schmid and P. Pérez, "Aggregating Local Descriptors into a Compact Image Representation", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3304–3311, 2010.

56. Perronnin, F. and C. Dance, "Fisher Kernels on Visual Vocabularies for Image Categorization", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8, 2007.

57. Jégou, H., F. Perronnin, M. Douze, J. Sánchez, P. Pérez and C. Schmid, "Aggregating Local Image Descriptors into Compact Codes", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 34, No. 9, pp. 1704–1716, 2012.

58. Macqueen, J., "Some Methods for Classification and Analysis of Multivariate Observations", *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297, 1967.

59. Dempster, A. P., N. M. Laird and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm", *Journal of the Royal Statistical Society, Series B*, Vol. 39, No. 1, pp. 1–38, 1977.

60. Verbeek, J. J., N. Vlassis and B. Kröse, "Efficient Greedy Learning of Gaussian Mixture Models", *Neural Computation*, Vol. 15, pp. 469–485, 2003.

61. Figueiredo, M. A. and A. Jain, "Unsupervised Learning of Finite Mixture Models", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 3, pp. 381–396, 2002.

62. Joachims, T., "Text Categorization with Suport Vector Machines: Learning with Many Relevant Features", *European Conference on Machine Learning (ECML)*, pp. 137–142, Springer-Verlag, 1998.

63. Csurka, G., C. R. Dance, L. Fan, J. Willamowski and C. Bray, "Visual Categorization with Bags of Keypoints", *Workshop on Statistical Learning in Computer Vision, ECCV*, pp. 1–22, 2004.

64. Jaakkola, T. and D. Haussler, "Exploiting Generative Models in Discriminative Classifiers", *In Advances in Neural Information Processing Systems 11*, pp. 487–493, MIT Press, 1998.

65. Cortes, C. and V. Vapnik, "Support-Vector Networks", *Machine Learning*, Vol. 20, No. 3, pp. 273–297, 1995.

66. Vapnik, V. N., *The Nature of Statistical Learning Theory*, Springer-Verlag New York, Inc., 1995.

67. Alpaydin, E., *Introduction to Machine Learning*, The MIT Press, 2014.

68. Fan, R.-E., K.-W. Chang, C.-J. Hsieh, X.-R. Wang and C.-J. Lin, "LIBLINEAR: A Library for Large Linear Classification", *Journal of Machine Learning Research*, Vol. 9, pp. 1871–1874, 2008.

69. Chang, C.-C. and C.-J. Lin, "LIBSVM: A Library for Support Vector Machines", *ACM Transactions on Intelligent Systems and Technology*, Vol. 2, pp. 27:1–27:27,

2011.

70. Huang, G.-B., Q.-Y. Zhu and C.-K. Siew, "Extreme Learning Machine: Theory and Applications", *Neurocomputing*, Vol. 70, No. 1–3, pp. 489 – 501, 2006.

71. Minhas, R., A. Baradarani, S. Seifzadeh and Q. J. Wu, "Human Action Recognition Using Extreme Learning Machine Based on Visual Vocabularies", *Neurocomputing*, Vol. 73, No. 10–12, pp. 1906 – 1917, 2010.

72. Iosifidis, A., A. Tefas and I. Pitas, "Regularized Extreme Learning Machine for Multi-View Semi-Supervised Action Recognition", *Neurocomputing*, Vol. 145, No. 0, pp. 250 – 262, 2014.

73. Viola, P. and M. Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. I–511–I–518, 2001.

74. Escalera, S., X. Baro, J. Gonzalez, M. A. Bautista, M. Madadi, M. Reyes, V. Ponce, H. J. Escalante, J. Shotton and I. Guyon, "ChaLearn Looking at People Challenge 2014: Dataset and Results", *ChaLearn Looking at People Workshop*, 2014.

75. Sanchez, D., M. A. Bautista and S. Escalera, "HuPBA 8k+: Dataset and ECOC-GraphCut Based Segmentation of Human Limbs", *Neurocomputing*, Vol. 150, No. A, pp. 173–188, 2015.

76. Peng, X., L. Wang, Z. Cai and Y. Qiao, "Action and Gesture Temporal Spotting with Super Vector Representation", *ChaLearn Looking at People Workshop, ECCV*, Vol. 8925 of *Lecture Notes in Computer Science*, pp. 518–527, 2014.

77. Jain, M., J. van Gemert and C. G. M. Snoek, "University of Amsterdam at THUMOS Challenge 2014", *ECCV Workshop on Action Recognition with a Large Number of Classes*, 2014.

78. Oneata, D., J. Verbeek and C. Schmid, "The LEAR Submission at THUMOS 2014", *ECCV Workshop on Action Recognition with a Large Number of Classes*, 2014.

79. Van de Sande, K., T. Gevers and C. Snoek, "Accelerating Visual Categorization with the GPU", K. Kutulakos (Editor), *Trends and Topics in Computer Vision*, Vol. 6554 of *Lecture Notes in Computer Science*, pp. 436–449, Springer Berlin Heidelberg, 2012.

80. Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge", `arXiv:1409.0575`, 2014.