THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

LOCAL VISUAL PATTERN MODELLING FOR IMAGE AND VIDEO CLASSIFICATION

Peng Wang

A thesis submitted for the degree of Doctor of Philosophy at

The University of Queensland in 2017

School of Information Technology & Electrical Engineering

Abstract

Local visual pattern modelling is one of the most important problems in the era of hand-crafted features where the aim is to bridge the gap between a set of local visual representations and some high-level vision tasks such as image and/or video classification. Recent years witnesses the overwhelming success of deep Convolutional Neural Networks (CNN), which poses new challenges and open problems for local pattern modelling.

The thesis focuses on the following problems. They are 1) How to leverage the CNN architecture to effectively model the local patterns? 2) How to combine the merits of supervised coding and Fisher vector coding, which are two state-of-the-art coding schemes, to model the high-dimensional local representations, e.g., CNN descriptors, to lead to better classification performance? 3) How to discard the strong assumption that local patterns are distributed i.i.d. and model the dependencies between region-level CNN features for some high-level tasks?

For the first question, we propose two CNN architectures to model the local features of videos for action recognition. While one piece of work focuses on how to design a network to effectively encode and aggregate multiple kinds of local features of a video, the other work proposes a convolutional pooling strategy to explore the temporal information hidden within the frame-level representations. These two works raise flexible CNN architectures that are compatible with video format and lead to promising action recognition performance.

Supervised encoding and Fisher vector encoding are two representative schemes to create image representations. Both of them can achieve state-of-the-art image classification performance but through different strategies: the former extracts discriminative patterns from local features at the encoding stage while the latter preserves rich information into high-dimensional signatures derived from a generative model of local features. For the second problem, we propose a hybrid Fisher vector encoding scheme for image classification which combines the strategies from both of the above two encoding methods. The key idea is to leverage supervised encoding to decompose local features into a discriminative part and a residual part and then build a generative model based on this decomposition.

For the third problem, we study a challenging problem of identifying unusual instances of known objects in images within an "open-world " setting. That is, we aim to find objects that are members of a known class, but which are not typical of that class. We propose to identify unusual objects by inspecting the distributions of local visual patterns at multiple image regions. Considering the promising performance of Region CNN [37], we represent an image by a set of local CNN features and then map them into scalar detection scores to get rid of the distraction influence of irrelevant content. To model the region-level score distribution we propose to use Gaussian Process (GP) to

construct two separate generative models, one for “regular object” and the other for “other objects”. We design a new covariance function to simultaneously model the detection score at a single location and the score dependencies between multiple regions. This treatment allows our method to capture the spatial dependencies between local regions, which turns out to be crucial for identifying unusual objects.

Declaration by Author

This thesis is composed of my original work, and contains no material previously published or written by another person except where due reference has been made in the text. I have clearly stated the contribution by others to jointly-authored works that I have included in my thesis.

I have clearly stated the contribution of others to my thesis as a whole, including statistical assistance, survey design, data analysis, significant technical procedures, professional editorial advice, and any other original research work used or reported in my thesis. The content of my thesis is the result of work I have carried out since the commencement of my research higher degree candidature and does not include a substantial part of work that has been submitted to qualify for the award of any other degree or diploma in any university or other tertiary institution. I have clearly stated which parts of my thesis, if any, have been submitted to qualify for another award.

I acknowledge that an electronic copy of my thesis must be lodged with the University Library and, subject to the policy and procedures of The University of Queensland, the thesis be made available for research and study in accordance with the Copyright Act 1968 unless a period of embargo has been approved by the Dean of the Graduate School.

I acknowledge that copyright of all material contained in my thesis resides with the copyright holder(s) of that material. Where appropriate I have obtained copyright permission from the copyright holder to reproduce material in this thesis.

Publications during candidature

- Lingqiao Liu*, Peng Wang*, Chunhua Shen, Lei Wang, Anton van den Hengel, Chao Wang, and Heng Tao Shen *Compositional Model based Fisher Vector Coding for Image Classification*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017 (accepted as full paper on 1 Dec. 2016, * indicates equal contribution). (incorporated as Chapter 5).

- Lianli Gao, Peng Wang, Jingkuan Song, Zi Huang, Jie Shao, and Heng Tao Shen *Event Video Mashup: from Hundreds of Videos to Minutes of Skeleton*. AAAI Conference on Artificial Intelligence (AAAI), 2017 (accepted as oral).

- Litao Yu, Yang Yang, Zi Huang, Peng Wang, Jingkuan Song, and Heng Tao Shen *Web Video Event Recognition by Semantic Analysis from Ubiquitous Documents*. IEEE Transactions on Image Processing, 2017 (accepted as full paper on 26 Sep. 2016).

- Jiewei Cao, Zi Huang, Peng Wang, Chao Li, Xiaoshuai Sun, and Heng Tao Shen *Quartet-net Learning for Visual Instance Retrieval*. In ACM International Conference on Multimedia (ACM MM), 2016, pages 456-460, DOI:10.1145/2964284.2967262, (short paper).

- Peng Wang, Lingqiao Liu, Chunhua Shen, Zi Huang, Anton van den Hengel, and Heng Tao Shen *Whats Wrong with that Object? Identifying Images of Unusual Objects by Modelling the Detection Score Distribution*. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pages 1573-1581, DOI:10.1109/CVPR.2016.174, (full paper and spotlight presentation). (incorporated as Chapter 6).

- Peng Wang, Yuanzhouhan Cao, Chunhua Shen, Lingqiao Liu, and Heng Tao Shen *Temporal Pyramid Pooling Based Convolutional Neural Networks for Action Recognition*. IEEE Transactions on Circuits and Systems for Video Technology, 2016, DOI: 10.1109/TCSVT.2016.2576761, (full paper). (incorporated as Chapter 3).

- Peng Wang, Yang Yang, Zi Huang, Jiewei Cao, and Heng Tao Shen *WeMash: An Online System for Web Video Mashup*. ACM International Conference on Multimedia (ACM MM), 2014, pages 753-754, DOI:10.1145/2647868.2654868, (demo).

Publications included in this thesis

Peng Wang, Yuanzhouhan Cao, Chunhua Shen, Lingqiao Liu, and Heng Tao Shen *Temporal Pyramid Pooling Based Convolutional Neural Networks for Action Recognition*. IEEE Transactions on Circuits and Systems for Video Technology, 2016, DOI: 10.1109/TCSVT.2016.2576761, -incorporated as Chapter 3.

Contributor	Statement of contribution
Peng Wang	Idea discussion and formulation (30%) Experiment design and conduction (60%) Paper writing (40%)
Yuanzhouhan Cao	Idea discussion and formulation (20%) Experiment design and conduction (40%) Paper writing (20%)
Chunhua Shen	Idea discussion and formulation (20%) Paper writing (10%)
Lingqiao Liu	Idea discussion and formulation (15%) Paper writing (20%)
Heng Tao Shen	Idea discussion and formulation (15%) Paper writing (10%)

Lingqiao Liu, Peng Wang, Chunhua Shen, Lei Wang, Anton van den Hengel, Chao Wang, and Heng Tao Shen *Compositional Model based Fisher Vector Coding for Image Classification*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017 (accepted as full paper on 01 Dec. 2016) -incorporated as Chapter 5.

Peng Wang, Lingqiao Liu, Chunhua Shen, Zi Huang, Anton van den Hengel, and Heng Tao Shen *Whats Wrong with that Object? Identifying Images of Unusual Objects by Modelling the Detection Score Distribution*. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2016, pages 1573-1581, -incorporated as Chapter 6.

Contributor	Statement of contribution
Lingqiao Liu	Idea discussion and formulation (40%) Experiment design and conduction (40%) Paper writing (50%)
Peng Wang	Idea discussion and formulation (30%) Experiment design and conduction (60%) Paper writing (30%)
Chunhua Shen	Idea discussion and formulation (10%) Paper writing (10%)
Lei Wang	Idea discussion and formulation (10%) Paper writing (5%)
Anton van den Hengel	Proofreading
Chao Wang	Proofreading
Heng Tao Shen	Idea discussion and formulation (10%) Paper writing (5%)

Contributor	Statement of contribution
Peng Wang	Idea discussion and formulation (35%) Experiment design and conduction (70%) Paper writing (30%)
Lingqiao Liu	Idea discussion and formulation (35%) Experiment design and conduction (30%) Paper writing (30%)
Chunhua Shen	Idea discussion and formulation (10%) Paper writing (10%)
Zi Huang	Idea discussion and formulation (10%) Paper writing (10%)
Anton van den Hengel	Paper writing (10%)
Heng Tao Shen	Idea discussion and formulation (10%) Paper writing (10%)

Contributions by others to the thesis

For all the published research work included in this thesis, Prof. Heng Tao Shen, as my principle advisor, has provided very helpful insight into the overall as well as the technical details and research problems; guidance for problem formulation as well as constructive comments and feedback. He also assisted with both the refinement of the idea and the pre-submission edition.

Statement of parts of the thesis submitted to qualify for the award of another degree

None.

Acknowledgments

I would like to express deep gratitude to my principle supervisor, Prof. Heng Tao Shen, for his in-depth guidance on my research and generous support for my life. It is with his help that I can solve one and another research problems and finish this thesis.

I would also like to thank the other members in my advisory team, Dr. Zi Huang and Prof. Chunhua Shen, for their constructive advice and insightful feedback.

I am very thankful to Dr. Lingqiao Liu. Whenever I come across any concrete research problem, he is always there to help me.

My thanks also go to the DKE group in the School of Information Technology and Electrical Engineering for providing me a world-class research environment. Specially, I thank the members Jiewei Cao, Litao Yu, Jianfeng Wang, Yang Yang, Jingkuan Song, Xiaofeng Zhu, Hongyun Cai, Weiqing Wang, Chao Li et al, for not only the research discussions but also the happy time we enjoyed together.

Last but not the least, I would like to thank my parents and my wife for their spiritual support throughout the tough times during the last three years.

Keywords

visual classification, local pattern modelling, deep learning, video analytics

Australian and New Zealand Standard Research Classifications (ANZSRC)

ANZSRC code: 080109, Pattern Recognition and Data Mining, 70%

ANZSRC code: 080201, Analysis of Algorithms and Complexity, 30%

Fields of Research (FoR) Classification

FoR code: 0801, Artificial Intelligence and Image Processing, 90%

FoR code: 0899, Other Information and Computing Sciences, 10%

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we give a brief introduction of the research in this thesis, including the background, problem formulation, contributions, and the organization of the thesis.

1.1 Background

Visual classification is a high-level task in the field of computer vision where the aim is to predict whether a visual category appears in an image or a video. With the variation of the visual category, visual classification has wide range of applications such as object classification, scene classification, action recognition, event detection and so on.

An image or a video can be represented by a set of local visual patterns, each of which captures the information contained within a small region, e.g., a patch in an image or a spatio-temporal region in a video. The objective of local visual pattern modelling is to bridge the gap between these local representations and some high-level vision tasks such as classification. Ever since bag-of-features (BoF) model [20], local visual pattern modelling has attracted a lot of research attention. The general idea is to first map the local feature into another space that is more suitable for high-level tasks and then aggregate these local representations into a global signature via various pooling methods. The key towards the success of the visual task depends both on the representation power of the local feature and the local feature modelling method built catering for the characteristics of the features.

Recently, benefiting from the colletion of large-scale training data and the computation accelaration, deep Convolutional Neural Networks (CNN) [54] shows promising performance in the field of visual classification. However, the requirement of a large-scale training data limits its applications especially when there is not sufficient data. Many approaches choose to use the CNN activations as

local features and directly apply existing local feature modelling methods on top of them [38][116]. However, the emergence of the CNN representations and CNN architectures do introduce some new open problems for local visual pattern modelling, e.g., can we design better modelling methods for CNN activations which have much higher dimensionalities comparing to traditional local features like SIFT [68], HOG [21], HOF [22] ? Can we resort to the CNN architecture to develop effective local feature modelling schemes?

1.2 Problem formulations

The thesis focuses on the following three topics: designing spatio-temporal CNN architecture to model the local features of videos for action recognition, high-dimensional local feature coding for image classification and unusual object identification from images via modelling the local visual pattern distribution.

1.2.1 Spatio-temporal CNN based Local Feature Modelling for Action Recognition

Encouraged by the success of CNN in image classification, recently much effort is spent on applying CNN to video based action recognition problems. One challenge is that video contains a varying number of frames which is incompatible to the standard input format of CNN. Existing methods handle this issue either by directly sampling a fixed number of frames or bypassing this issue by introducing a 3D convolutional layer which conducts convolution in spatial-temporal domain.

We address this problem from the perspective of local feature modelling, where the local feature can be either the frame-level representation or the local region of the temporal evolution of every dimension of such frame-level representation. For the first kind of local feature, we study how to effectively encode and aggregate multiple sources of frame-level representation in a CNN architecture. For the second type of local representation, we investigate how to use a CNN architecture to extract the dynamic information of a video while maintaining the number of parameters controllable.

1.2.2 Hybrid Generative Model based Fisher Vector Coding for Image Classification

In local feature modelling framework, local feature coding is an important step for classification performance. The objective of local feature coding is to map these local features into a space more suitable for pooling and consequently more suitable for high-level vision task.

In this part we propose a new Fisher vector coding scheme for local feature encoding. Following the general rules of Fisher vector coding, we first build a generative model for local features and then derive the coding vectors as the derivatives of the likelihood w.r.t the model parameters. One problem of existing methods is that they directly model the generation process of the local features. But we argue that any local feature should be composed of discriminative information and residual information and these two parts have different properties and should be modelled separately. This idea motives us to combine supervised coding and Fisher vector coding to form a hybrid generative model. Our key idea is to leverage supervised encoding to decompose local features into a discriminative part and a residual part and then build a generative model based on this decomposition.

1.2.3 Unusual Object Identification via Modelling the Detection Score Distribution

Humans have the ability to detect the irregular status of objects without seeing the irregular patterns beforehand. Mimicking this ability with computer vision technique can be practically useful for the applications such as surveillance or quality control. Existing studies towards this goal are usually conducted on small datasets and controlled scenarios or specific type of irregularity. To address this issue, in this work we present a large dataset which captures more general irregularities and has more complex background. Moreover, we adopt a more realistic open world evaluation protocol. That is, we need to distinguish the unusual version of object-of-interest not only from the regular object belonging to the same category but also from the other objects (objects from other categories) at the test stage.

Considering the promising performance of Region CNN [37] in image classification, we represent an image by a set of region-level CNN features. In order to get rid of the distraction influence of the irrelevant content, we learn an object detector to map the CNN features into detection scores. The key observation motivating our approach is that for regular object images as well as other objects images, the region-level detection scores follow their own essential patterns in terms of both the score values

and the spatial distributions while the detection scores obtained from an unusual object image tend to break these patterns. To model this distribution, we propose to use Gaussian Processes (GP) to construct two separate generative models for the case of the regular object and the other objects.

1.3 Main contributions

The main contributions of this thesis include a set of local pattern modelling algorithms that are applied to different visual classification scenarios introduced above. More specifically, they are:

- We propose a novel network structure which allows an arbitrary number of video frames as input. The key part of the network is a module which consists of an encoding layer and a temporal pyramid pooling layer. The encoding layer maps the frame-level representations to a feature vector suitable for pooling and the temporal pyramid pooling layer is proposed to explicitly consider the weak temporal structure of videos. In addition, we also introduce a feature concatenation layer into our network to combine motion and appearance information.

- We propose a convolutional pooling method that can effectively capture the dynamic information contained in a video sequence. Instead of implementing this idea on the original high-dimensional frame-level representations, we view the temporal evolution of the feature value at each feature dimension as a 1D signal and learn a unique convolutional filter bank for each of these 1D signals. And by this strategy, our network can maintain a small number of model parameters.

- We propose a hybrid coding method which combines the ideas of both supervised encoding and Fisher vector encoding. Our key insight is that a local feature can be decomposed into a discriminative part and a residual part with the guidance of a supervised encoding method. Based on this decomposition, we propose a generative model which further produces a novel Fisher vector encoding scheme that outperforms its unsupervised Fisher vector encoding and supervised encoding counterparts.

- We propose a novel approach for the task of unusual object identification in an open world setting via inspecting the region-level detection score distribution of an image. We have proposed to use Gaussian Processes to model the score value at a single region as well as the score dependences between multiple local regions. Our method shows superior performance against some compared methods on a large dataset presented in this work.

1.4 Thesis organization

The rest of this thesis is organized as follows: In Chapter 2, a detailed literate review on local pattern modelling and their applications to image and video classification is given. In Chapter 3, a hybrid CNN network for video based action recognition is introduced. The network is proposed to encode and aggregate multiple sources of frame-level representations. In Chapter 4, we propose a network based pooling method to effectively extract the dynamic information of videos for action recognition. The proposed pooling method is not limited to video analysis but can be generalized to other time-series tasks. In Chapter 5, we introduce a novel coding method for high-dimensional local feature modelling, which shares the merits of both supervised coding and Fisher vector coding. In Chapter 6, we study the problem of unusual object identification via modelling the local visual pattern distribution. Finally the conclusion and the potential research directions are discussed in Chapter 7.

Chapter 2

Literature Review

In this part, I will go through the related works in the literature. Since the topics of the thesis are 1) leveraging deep neural network to model local visual patterns for video based action recognition, 2) hybrid Fisher Vector coding for image classification, and 3) unusual object identification via detection score modelling. I will first review the traditional handcrafted-feature based action recognition followed by deep neural networks based methods. Then I will introduce the existing local feature coding methods for image classification. Finally, some existing works for unusual object identification will be introduced.

2.1 Modelling Handcrafted Feature for Action Recognition

How to design an effective spatial-temporal representation of a video is the focus of action recognition. The representations can be generally categorized into two classes that are spatial-temporal interest points based descriptors and trajectory based representations. While the former detects interest points in the 3D space and extracts corresponding descriptors, the latter captures the motion information via tracking a given point. Apart from these two kinds of low-level representations, there are efforts dedicated to extracting some mid-level semantic representations for action recognition.

2.1.1 Spatio-temporal Interest Points

Many different spatial-temporal detectors [56, 27, 78, 113, 114, 46] and descriptors [52, 57, 93, 113, 58, 92] are proposed in the literature. In [56], the authors extended the Harris detector from 2D to 3D to obtain spatial-temporal interest points. The interest points are local maxima of a cornerness criterion based on spatio-temporal second-moment matrix at each video point. The authors in [27]

proposed a cuboid interest points detector via combining 2D Guassian filter and temporal Gabor filter. The work in [8] extended this approach by using 2D Gabor filters of different orientations. Willems *et al.* [113] extended the Hessian saliency measure from image to video and proposed the Hessian detector. The detector measures the saliency of video points with the determinant of the 3D Hessian matrix. Wong *et al.* [114] applied non-negative matrix factorization to the video in order to add global information to the local interest points. Once the salient points are detected, different descriptors are utilized to describe the motion information. Schuldt *et al.* [92] used high-order derivatives to describe the interest points. Normalized brightness, gradient and optical flow are utilized to design the descriptors in [27]. In [93], SIFT [68] was extended to describe spatial-temporal information. Klser *et al.* introduced HOG3D descriptor and Willems *et al.* [113] proposed extended SURF which extends the image based SURF descriptors to videos. The authors in [58] combined HOG and histogram of optical flow (HOF) as local descriptors.

2.1.2 Trajectory based Representation

Different from interest points based representations that extract motion information at a given point, trajectory tracks a point over the temporal domain. Messing *et al.* [75] obtained the trajectories via tracking the Harris3D [56] points with KLT tracker [69]. Matikainen *et al.* [74] applied the bag of words framework to the quantized trajectory snippets. In [100], trajectories are formed by matching SIFT descriptors between consecutive frames. Intra-trajectory and inter-trajectory context are used to model the spatio-temporal context information. Raptis *et al.* [86] concatenated local features, i.e., HOG and HOF, along the trajectories to describe the motion information. Inspired by the success of dense sampling in image classification [34], Wang *et al.* [106] proposed dense trajectory. They densely sample points in each frame and track the points using dense optical flow. Along the trajectories, different local features are extracted. Apart from HOG and HOF, the MBH descriptor [23], computed as derivatives of optical flow, was employed to decrease the influence of camera motion. Then bag of words are applied to the descriptors to form the video-level representation. To explicitly address the problem of camera motion, Wang *et al.* proposed improved dense trajectory [107] which performed video stabilization by estimating homography between consecutive frames using human detectors. Their performance is further boosted by replacing traditional bag of words encoding with Fisher vectors encoding [84]. Ever since the introduction of dense trajectory, there emerged several methods [9, 81, 82] focusing on how to effectively make use of the trajectory information to obtain

better classification performance. In [9], Cai *et al.* used probabilistic CCA [4] to decompose descriptor pairs (e.g., MBHx and MBHy) into correlated part and independent part. Then they concatenated the correlated part and the Fisher vectors of the independent parts as the video representation. Peng *et al* [82] proposed stacked Fisher vector for action recognition. They performed a second-level Fisher vector encoding on local Fisher vectors of trajectory descriptors and found the stacked Fisher vector and traditional global Fisher vector can compensate well to each other. In [81], the authors conducted comprehensive analysis to different descriptor fusion and encoding mechanisms and compared their performances. In [111], Wang *et al.* replaced the HOG features of trajectories with more discriminative convolution features of a convolutional neural network and the performance was boosted significantly.

2.1.3 Discriminative Mid-Level Representation Mining

Apart from the low-level descriptors and/or encodings, there are efforts dedicated to mining discriminative mid-level representations for action recognition. Wang *et al.* [110] proposed to mine discriminative 3D parts, called motionlet, by clustering and ranking. In [43], the authors mined discriminative spatio-temporal patches in the video via exemplar-SVM and used SVM scores for action recognition. Zhu *et al.* [130] learned actons via a max-margin multi-channel multiple instance learning framework. Zheng *et al.* [127] formulated the attributes selection as a submodular optimization problem and used the attributes for action recognition. Instead of mining discriminative mid-level parts, [82] conducted Fisher vector encoding on densely sampled subvolumes followed by a second-level encoding on the local Fisher vectors.

2.1.4 Summary

This part introduced the hand-crafted descriptors designed to extract the spatio-temporal information of videos. Generally, the methods can be categorized into two classes, where one class is based on static spatio-temporal interest points and the other class represents the video by tracking the points over the temporal domain. For the former class, the interest points are first detected based on some criteria and then some features are extracted around these points. And these features are usually the 3D form of some standard spatial features such as SIFT [68] or SURF[5]. The trajectory-based descriptors were proposed to extract the long-range motion information, which significantly boosted the motion-relevant video classification performance. Among these methods, the dense trajectories [106, 107] have become the state-of-the-art features for videos. The general idea behind is to densely

sample and track the points and extract the hand-crafted features along the trajectories. It inspires many successive works on how to improve the dense trajectories, for example how to combine the advantage of deep learning and dense trajectory. Our work in Chapter 3 falls into this range.

2.2 Deep Neural Networks for Action Recognition

Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) have proven successful in different vision tasks. While the former achieves state-of-the-art performance at image classification and object detection, the latter shows promising results on time-series tasks like speech recognition and machine translation. In this part, we will overview their applications to video classification.

2.2.1 Deep Neural Networks for Action Recognition

Recently, CNN and RNN have been applied to action recognition and achieved impressive results. In [50], Karpathy *et al.* performed 3D convolution to the video to extract the spatio-temporal information. Since it works with video input, it sacrifice the possibility to leverage the powerful off-the-shelf image based CNN to initialize the model parameters. Thus they collected a huge dataset to avoid the problem of over-fitting. To speed up the network training, a multiresolution mechanism was proposed which simultaneously takes videos of different resolutions as input. In [96], Simonyan and Zisserman designed a two-stream network structure for action recognition. While one stream takes RGB frames as input, the other stream takes optical flow as input. The decision scores of these two streams are fused together as final classification results. Since the network structure samples a subset of frames as input, it may risk missing important content. In dense trajectory, while HOF and MBH provide motion information along the trajectories, HOG features describe the local appearance information. The authors in [111] replaced the HOG features with the convolutional features of a pre-trained deep network which significantly improved the classification results.

Inspired by the success obtained at speech analysis and natural language processing, there are attempts to apply LSTM to video classification. Donahue *et al.* [29] proposed a long-term recurrent convolutional networks for action recognition. The video frames are input into a convolutional network and the output of the first or second fully-connected layer of the CNN is fed into a stack of LSTM networks. The outputs of the LSTM at different timestamps are aggregated via average pooling to be the final decision score. [99] proposed an unsupervised LSTM learning mechanism to learn video

representations. The network is composed of two parts, an encoder to map an input sequence into a fixed-length representation and a decoder to perform sequence reconstruction or sequence prediction. And the output of the encoder is utilized as the video representation for classification.

2.2.2 Summary

This part introduced some standard works that build deep neural networks for video based action recognition. The works in [50] and [96] are two inspiring works. The former uses 3D convolutional kernel to directly extract spatio-temporal information and the latter decouples the appearance and motion clues into two separate networks and proposes to apply CNN on top of optical flow to capture motion information. Our work in Chapter 3 adopts the two-stream structure as well but we overcome the drawback of [96] that needs to sample fixed number of frames by proposing a temporal pyramid pooling module.

2.3 Local Feature Coding for Image Classification

Local feature coding is an important step in local feature modelling to improve classification performance. The mechanism of it is to encode the local feature into a space that is more suitable for high-level vision tasks. In this part, we will go through some standard local feature coding methods.

2.3.1 Hard and Soft Assignment Coding

Hard assignment coding, widely used in bag-of-features model, is a coding method that encodes a local feature by assigning it to the nearest neighbour in the codebook. Essentially, it works similar to K-means clustering, which assumes the feature space can be separated into k (k is size of codebook) regions and a local feature is assigned to one of these regions. It generates a binary coding vector and thus may result in large quantization error especially for the local features close to the boundary of different regions.

To address this problem, soft assignment [36] is proposed. It relaxes the assumption that a local feature is from its nearest region but encodes it using the distance between it and the centres in the codebook. This strategy significantly reduces the quantization error and improves the representation power of coding vector.

2.3.2 Sparse Coding

Another popular way of local feature coding is called sparse coding [117]. It assumes a local feature can be represented by the combination of some pre-learned elementary bases called "atoms" in a codebook and constrains the combination coefficients to be sparse. The sparsity constraint is mathematically formulated as minimizing a $L0$-norm term. For ease of optimization, $L0$ term is usually replaced with a $L1$-norm term which can enforce the reconstruction coefficient to have small number of non-zero elements.

$$\mathbf{u}_i = \arg\min_{\mathbf{u}\in \mathrm{R}^k} \|\mathbf{x}_i - \mathbf{B}\mathbf{u}\| + \lambda\|\mathbf{u}\|_1 \tag{2.1}$$

The difference between soft assignment and sparse coding is that the latter uses a small number of bases to reconstruct the local feature. Obtaining the coding vector by solving a $l1$ regularized regression, sparse coding can lead to lower reconstruction error and further improves the classification performance.

2.3.3 Fisher Vector Coding

Given a set of local features extracted from an image which we denote as $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T\}$, in the Fisher kernel [42] $\mathbf{X}$ can be described by the gradient vector of the likelihood function w.r.t. the model parameter vector λ:

$$\mathbf{G}_\lambda^{\mathbf{X}} = \nabla_\lambda \log P(\mathbf{X}|\lambda) = \sum_i \nabla_\lambda \log P(\mathbf{x}_i|\lambda). \tag{2.2}$$

From a bag-of-features model perspective, the above process can be seen as first calculating the gradient or Fisher vector of each local feature and then performing sum-pooling. In this sense, the Fisher vector of each local feature, $\nabla_\lambda \log P(\mathbf{x}_i|\lambda)$, can be seen as a coding vector and we call it Fisher vector coding in this thesis.

Most existing works model the generation process of $\mathbf{x}$ using a GMM. The general process is as follows:

- Draw a Gaussian model $\mathcal{N}(\mu_k, \Sigma_k)$ from the prior distribution $P(k),\ k = 1, 2, \cdots, m$.
- Draw a local feature $\mathbf{x}$ from $\mathcal{N}(\mu_k, \Sigma_k)$.

Later, several variants were proposed to improve the basic FVC. One of the first identified facts is that normalisation of Fisher vectors is essential to achieving good performance [85]. At the same

time, several similar variants were developed independently from different perspectives [45, 129, 120]. The improved Fisher vector and its variants showed state-of-the-art performance in image classification and quickly became one of the most popular visual representation methods in computer vision. Numerous approaches have been developed to further enhance performance. For example, The work in [2] closely analysed particular implementation details of VLAD, a famous variant of FVC. The work in [53] tried to incorporate spatial information from local features into the Fisher vector framework. In [18, 17], the authors revisited the basic i.i.d assumption of FVC and pointed out its limitation. They proposed a non-iid model and derived an approximated Fisher vector for image classification. Also, FVC has been widely applied to various applications and has demonstrated state-of-the-art performance in the related fields. For example, in combination with local trajectory features, FVC-based systems have achieved the state-of-the-art in video-based action recognition [108, 83].

2.3.4 FVC with CNN Local Features

Conventionally, most FVC implementations are applied to low-dimensional hand-crafted local features, such as SIFT [68]. With the recent development of deep learning, it has been observed that simply extracting neural activations from a pre-trained CNN model acheives significantly better performance [89]. However, it was soon discovered that directly using activations from a pre-trained CNN as global features is still not the optimal choice [39, 16, 66, 25], at least for a small/medium sized classification problems for which fine-tuning a CNN does not always improve performance significantly. Instead, it has been shown that it is beneficial to treat CNN activations as local features. In this case, the traditional local feature coding approaches, such as FVC, can be readily applied. The work in [39] points out that the fully-connected activation of a pre-trained CNN is not translation invariant. Thus, the authors propose to extract CNN activations from multiple regions of an image and use VLAD to encode these local features. In [16] and [65], the value of convolutional layer activations are analysed. They suggest that convolutional feature activations can be seen as a set of local features extracted at a dense grid. In particular, the work in [16] builds a texture classification system by applying FVC to the convolutional layer local features.

2.3.5 Supervised Coding and FVC

Here we briefly review the work of supervised coding and the attempts to combine it with FVC. Using supervised information to create an image representation is a popular idea in image classification. For example, supervised information has been utilized to learn discriminative codebooks for encoding

local features [59, 63, 48, 118, 112], either by using a separated codebook learning step [59, 48] or in an end-to-end fashion [63, 118]. Supervised information has also been applied to discover a set of middle-level discriminative patches [26, 49, 62] to train some patch detectors which are essentially local feature encoders. The CNN can also be seen as a special case of supervised coding methods if we view the responses of the filter bank in a convolutional layer as the coding vector of convolutional activations of the previous layer. From this perspective, the deep CNN can be seen as a hierarchical extension of the supervised coding method.

Generally speaking, the aforementioned supervised coding and FVC represent two major methodologies for creating discriminative image representations. For supervised coding, the supervised information is passed through the early stage of a classification system, i.e. by learning a dictionary or coding function. For FVC the information content of local features will be largely presevered in the corresponding high-dimensional signature. Then a simple classifier can be used to extract the discriminative patterns for classification. There have been several works trying to combine the idea of FVC and supervised coding. The work in [95] learns the model parameters of FVC in an end-to-end supervised training framework. In [102], multiple layers of Fisher vector coding modules are stacked into a deep architecture to form a deep network.

2.3.6 Summary

In this part we went through some classic local feature encoding strategies, the objective of which is to map the local features into a space more suitable for pooling. Most coding methods involve dictionary learning and local feature coding. Different coding methods can be distinguished by how to learn the dictionary and encode the local feature. Fisher vector coding (FVC) [42] represents the state-of-the-art local feature coding strategy. Its good performance is due to that it can preserve rich information into a high-dimensional signature. An alternative idea to encode the local features is by resorting to the supervised information which in the thesis we call supervised coding. In Chapter 5, we propose a compositional coding method that shares the advantages of both FVC and supervised coding.

2.4 Local Feature Modelling for Irregularity Identification

There exists a variety of work focusing on irregular image and/or video detection. While some approaches attempt to detect irregular image parts or video segments given a regular database [126, 7,

128, 40], other efforts are dedicated to addressing some specific types of irregularities [80, 15] such as out-of-context via building some corresponding models.

2.4.1 Modelling the Dissimilarity from Regular

Standard approaches for irregularity detection are based on the idea of evaluating the dissimilarity from regular. The authors of [128, 40] formulate the problem of unusual activity detection in video into a clustering problem where unusual activities are identified as the clusters with low inter-cluster similarity. The work [7] detects the irregularities in image or video by checking whether the image regions or video segments can be composed using large continuous chunks of data from the regular database. Despite the good performance in irregularity detection, this method severely suffers from the scalability issue, because it requires to traverse the database given any new query data. Sparse coding [60] is employed in [126] for unusual events detection. This work is based on the assumption that unusual events cannot be well reconstructed by a set of bases learned from usual events.

2.4.2 Specific Irregularity Identification

Another stream of work focus on addressing specific types of irregularities. The work of[14, 15] focus on exploiting contextual information for object recognition or out-of-context detection, like "car floating in the sky". In [14], they use a tree model to learn dependencies among object categories and in [15] they extend it by integrating different sources of contextual information into a graph model. The work [80] focuses on finding abnormal objects in given scenes. They consider wider range of irregular objects like those violate co-occurrence with surrounding objects or violate expected scale. However, the applications of these methods are very limited since they rely on pre-learned object detector to accurately localize the object-of-interest. Recently the work in [91] delves into various types of atypicalities and makes a more comprehensive study.

2.4.3 Summary

Most existing works, if not all, addresses the problem of irregularity identification in a restricted way, either using the scheme of comparing to the regular or focusing on specific irregularity type. In Chapter 6 we propose to address the problem in an open-world setting from the aspect of local visual pattern modelling. In our method, we represent each image by a set of local features and do irregularity identification via modelling the interactions between these local regions.

Chapter 3

Temporal Pyramid Pooling based CNN for Video Classification

3.1 Overview

Encouraged by the success of Convolutional Neural Networks (CNNs) in image classification, recently much effort is spent on applying CNNs to video classification. One challenge is that video contains a varying number of frames which is incompatible to the standard input format of CNNs. Existing methods handle this issue either by directly sampling a fixed number of frames or bypassing this issue by introducing a 3D convolutional layer which conducts convolution in spatial-temporal domain.

In this paper we address this problem from the perspective of local visual pattern modelling where the local feature corresponds to the frame-level representation. We propose a novel network structure which allows an arbitrary number of frames as the network input. The key of our solution is to introduce a module consisting of an encoding layer and a temporal pyramid pooling layer. The encoding layer maps the activation from previous layers to a feature vector suitable for pooling while the temporal pyramid pooling layer converts multiple frame-level activations into a fixed-length video-level representation. In addition, we adopt a feature concatenation layer which combines appearance information and motion information. Compared with the frame sampling strategy, our method avoids the risk of missing any important frames. Compared with the 3D convolutional method which requires a huge video dataset for network training, our model can be learned on a small target dataset because we can leverage the off-the-shelf image-level CNN for model parameter initialization. Experiments on three challenging datasets, Hollywood2, HMDB51 and UCF101 demonstrate the effectiveness of

the proposed network.

3.2 Introduction

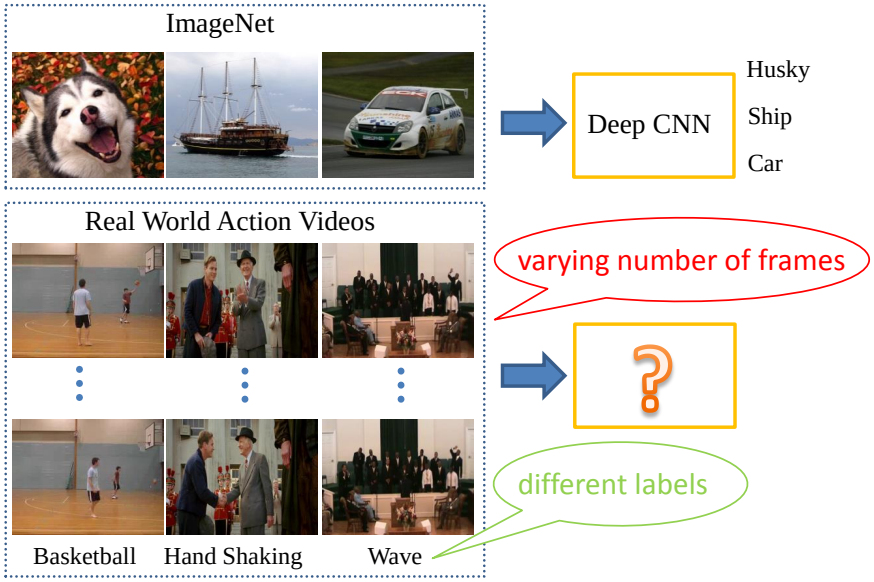


FIGURE 3.1: Incompatibility between videos and input format of CNN.

Recently, deep Convolutional Neural Networks has been established as the state-of-the-art method in image classification [54] and it has been demonstrated that a CNN pre-trained on a large image dataset, such as ImageNet [24], can be used to initialize networks built for other visual recognition tasks. Inspired by the success of CNN in image recognition, some studies attempt to apply CNN to video based action recognition. However, most existing deep models are designed to work with single image input. It is non-trivial to extend these models to videos since video clips often contain a varying number of frames. To handle this problem, the work in [96] samples a fixed number of frames and reshapes them into a compatible input format of an image-based CNN. However, sampling may risk missing important frames for action recognition, especially in videos with uncontrolled scene variation. Another strategy is to bypass this issue by directly using videos as input and replacing the 2D convolution with 3D convolution which is operated on the spatial-temporal domain. However, the above strategy sacrifices the possibility of leveraging the powerful off-the-shelf image-based CNN to initialize model parameters or extract mid-level features. Thus, it has to rely on a huge number of training videos to avoid the risk of over-fitting. For example, the authors in [50] collect a dataset of 1 million YouTube videos for network training which takes weeks to train with modern GPUs.

In this work, we propose a novel network structure which allows an arbitrary number of video frames as input. This is achieved by designing a module which consists of an encoding layer and a temporal pyramid pooling layer. The encoding layer maps the activations from previous layer to a feature vector suitable for pooling, which is akin to the encoding operation in the traditional bag-of-features model. The temporal pyramid pooling layer converts multiple frame-level activations into a fixed-length video-level representation. At the same time, the temporal pyramid pooling layer explicitly considers the weak temporal structure within videos. In addition, we also introduce a feature concatenation layer into our network to combine motion and appearance information.

3.3 Temporal pyramid pooling based CNN

3.3.1 Network overview

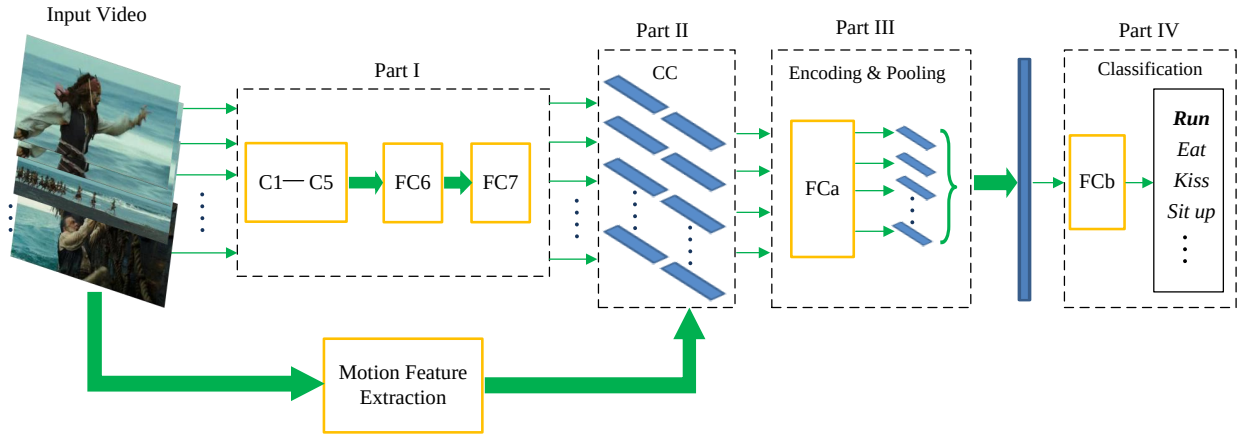


FIGURE 3.2: Overall structure of the proposed network.

The overall structure of our proposed network is shown in Fig. 4.3. It can be decomposed into four parts: the first part consists of five convolutional layers C1-C5 and two fully-connected layers FC6 and FC7. In the second part, the activation of FC7 is fed into the feature concatenation layer CC, which concatenates the output of FC7 and the frame-level motion feature. The concatenated feature then passes through a fully-connected layer FCa followed by a temporal pyramid pooling layer which converts frame-level features into the video-level feature. FCa together with the temporal pyramid pooling layer constitute the third part of our network, which is also the core part of our network. Finally, the classification layer, which is the fourth part of the network, is applied to the video-level feature to obtain the classification result. In the following sections, we discuss these four parts in

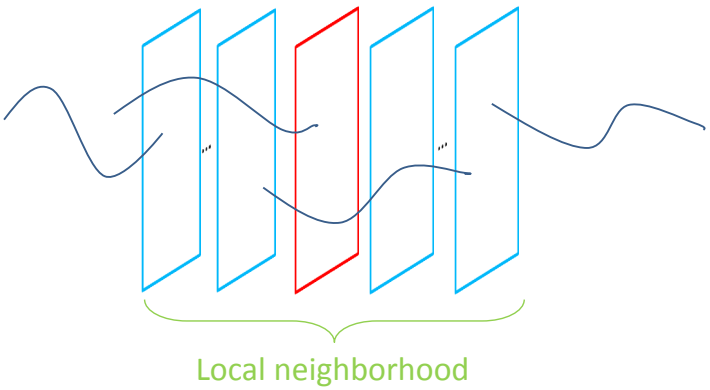


FIGURE 3.3: Illustration of frame-level motion feature generation.

detail.

3.3.2 Network architecture

Part I: C1 - FC7

The first part of our network is used to generate the frame-level appearance feature. We choose the structure of this part to be the same as an off-the-shelf CNN model. Thus, we can take advanatage of the model parameters pretrained on a large dataset, e.g., ImageNet [24] to initialize our network. More specifically, this part comprises 5 convolutional layers and 2 fully connected layers. The first convolutional layer has 64 kernels of size $11 \times 11 \times 3$ and a stride of 4 pixels which ensures fast processing speed. The second convolutional layer has 256 kernels of size $5 \times 5 \times 3$. The third, forth and fifth convolutional layers have 256 kernels of size $3 \times 3 \times 3$. Two fully connected layers both have 4096 neurons. Each frame in an input video is first resized to $224 \times 224 \times 3$ and then passes through the first part of our network, interleaved with ReLU non-linearties, max-pooling and normalization. The output of the seventh layer, a 4096 dimensional vector is then used as the static appearance feature of a video frame. At the training stage, we initialize the parameters of these seven layers with a pre-trained network in [12].

Part II: Feature concatenation layer and frame-level motion features

We introduce a feature concatenation layer to combine appearance and motion features since they have been shown to compensate each other in representing actions in videos. Our motion feature is built upon the dense trajectory descriptor because it achieves state-of-the-art performance. We only

Algorithm 1 Feature Merging Based Dimensionality Reduction

1: **procedure**
2: **Input**: A feature matrix $V \in \mathbb{R}^{m\times d}$, with m the number of training samples, d the dimensionality of the features.
3: For V, merging the features belonging to same class via average pooling, we get $\bar{\mathbf{S}} \in \mathbb{R}^{c\times d}$, with c the number of classes.;
4: Conduct k-means on columns of $\bar{\mathbf{S}}$, with number of clusters k being the target low dimension to which we want to reduce d.
5: Define $clique_j = \{i|cluster(i) = j\}$,$norm_j = \sqrt{|\{i|cluster(i) = j\}|}$, with i the feature indices, $i = 1, 2, \cdots, d$, and j cluster indices $\in \{1, 2, \cdots, k\}$. Given a new representation $h_i \in \mathbb{R}^d$, its low dimensional representation is $l_i \in \mathbb{R}^k$, and its jth element $l_{i,j} = \sum_{p\in clique_j} h_{i,p}/norm_j$.
6: **end procedure**

use HOF and MBH descriptors of improved dense trajectory [107] to describe motion information since we find that the 30-dimensional Trajectory descriptor in [107] does not contribute too much to the classification performance. Also, instead of utilizing trajectory features to describe motion information of the whole video, we extract motion features from a short temporal range, that is, within several consecutive frames. Fig. 3.3 illustrates this idea. For each frame, we extract the trajectories passing through a local neighbourhood and encode them using the Fisher vector. The motion feature of this frame is obtained by pooling all the Fisher vector encodings within this neighbourhood. Then this motion feature is concatenated with the appearance feature from FC7 to produce the frame-level representation. We refer to this fusion method as "early fusion".

In practice, however, the dimensionality of the Fisher vector encoding is too high to be handled by our network implemented on GPU. Thus, we employ a supervised feature merging algorithm variant in [67] (Eq. (7) in [67]) to perform dimensionality reduction. Compared with other methods, this method is very efficient in learning the dimensionality reduction function and performing dimensionality reduction especially in the scenario of reducing high-dimensional features to medium dimensions. More specifically, one only needs to calculate the mean of features in each class, which gives a data matrix $\bar{\mathbf{S}} \in \mathbb{R}^{c\times d}$, where d indicates the feature dimensionality and c indicates the total number of classes. Each column of $\bar{\mathbf{S}}$, denoted as $\mathbf{s}_i,\ i = 1, \cdots, d$, is treated as a c-dimensional 'signature' for the i-th feature. Then we perform k-means clustering on all d 'signatures' to group them into k clusters. Thus the d feature dimensions are partitioned into k groups and this grouping pattern is used to perform dimensionality reduction. The details are illustrated in Algorithm 1.

Part III: Encoding and temporal pyramid pooling layers

The encoding and temporal pyramid pooling layers constitute the most important part of our network. It transforms feature representations extracted from a varying number of frames into a fixed-length video-level representation. Note that these two layers are akin to the encoding and pooling operations in the traditional bag-of-features model. In the traditional bag-of-features model, an image contains a varying number of local features. In order to obtain a fixed-length image representation, one first applies an encoding operation to transform the local feature into a coding vector and then performs pooling to obtain the image level representation. The encoding step has been shown to be essential and pooling with the original local features without encoding usually leads to inferior performance. Similarly, the utilization of the encoding layer FCa in our network is of great importance as well. However, unlike most traditional bag-of-features models, in our work the encoding module FCa is embedded into a deep network structure and its parameters are adapted to the target classification task in an end-to-end fashion. Also, just like using spatial pyramid to incorporate the weak spatial information of local features, here we apply temporal pyramid pooling to better cater for the temporal structure of videos.

In our implementation, we calculate the output of FCa as $\mathbf{Y}_a = \sigma(\mathbf{X}\mathbf{W}_a + \mathbf{B}_a)$, where $\mathbf{W}_a \in \mathbb{R}^{d\times D}$ and $\mathbf{B}_a \in \mathbb{R}^{n\times D}$ are model parameters, $\mathbf{X} \in \mathbb{R}^{n\times d}$ and σ denote the input and "ReLU" activation function respectively. n indicates the number of frames in the input video, d and D are dimensionalities of the input frame-level representation and encoded representation respectively.

The temporal pyramid pooling strategy is illustrated in Fig. 3.4. The input video frames are partitioned in a coarse-to-fine fashion. Here we use two levels of partition. At the coarse level we treat the whole video as a pooling segment. At the fine level we evenly divided the video into multiple segments and perform pooling on each segment. The final video-level representation is obtained by concatenating pooling results from all the segments.

Part V: Classification layer

The final part of our network is a classification layer which classifies the video-level representation obtained from the temporal pyramid pooling layer. It calculates $\mathbf{Y}_b = \varphi(\mathrm{pool}\,(\mathbf{Y}_a)\,\mathbf{W}_b + \mathbf{B}_b)$ where $\mathbf{W}_b$ and $\mathbf{B}_b$ are model parameters, pool and φ are pooling and softmax [1] operation respectively. The output $\mathbf{Y}_b$ is a probability distribution indicating the likelihood of a video belonging to each class. In the training stage, we use the following loss function to measure the compatibility between this distribution and ground-truth class label:

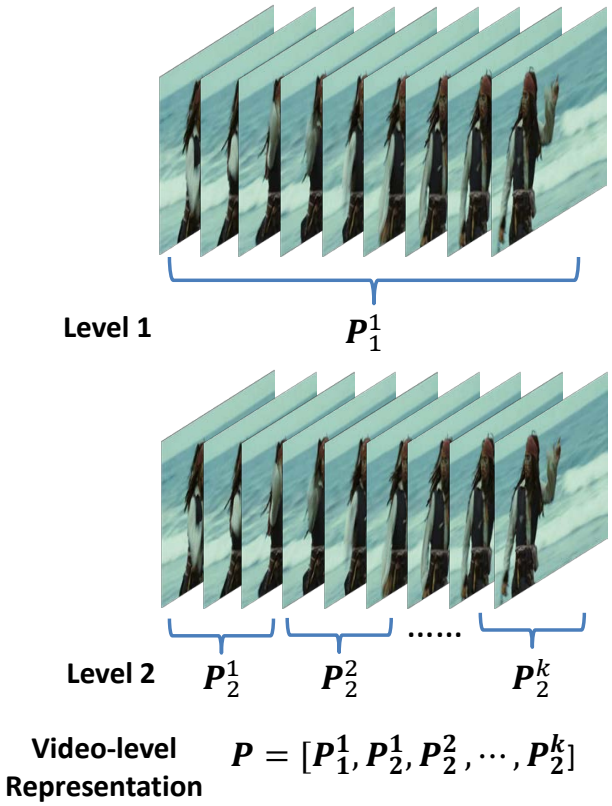


FIGURE 3.4: Illustration of temporal pyramid structure in the pooling layer.

$$L(\mathbf{W}, \mathbf{B}) = -\sum_{i=1}^{N} \log(\mathbf{Y}_b(c_i)), \tag{3.1}$$

where c_i denotes the class label of the ith video and N is the total number of training videos. Recall that $\mathbf{Y}_b$ is a c-dimensional vector. Here we use $\mathbf{Y}_b(c_i)$ denotes the value at c_ith dimension of $\mathbf{Y}_b$. Using Stochastic Gradient Descent (SGD), in each step we update model parameters by calculating the gradient of an instance-level loss $L_i(\mathbf{W}, \mathbf{B}) = \log(\mathbf{Y}_b(c_i))$.

3.3.3 Late fusion model

The aforementioned network structure combines motion and appearance information at the frame level. An alternative way is to fuse these two types of information after obtaining the output of the last layer of our network. We illustrate this scheme in Fig. 3.5. This scheme consists of two independent network streams. One stream uses appearance information and the other stream uses motion information. Each network in these two streams is very similar to that proposed in Fig. 4.3. The only difference is that the network in Fig. 3.5 does not have the feature concatenation layer.

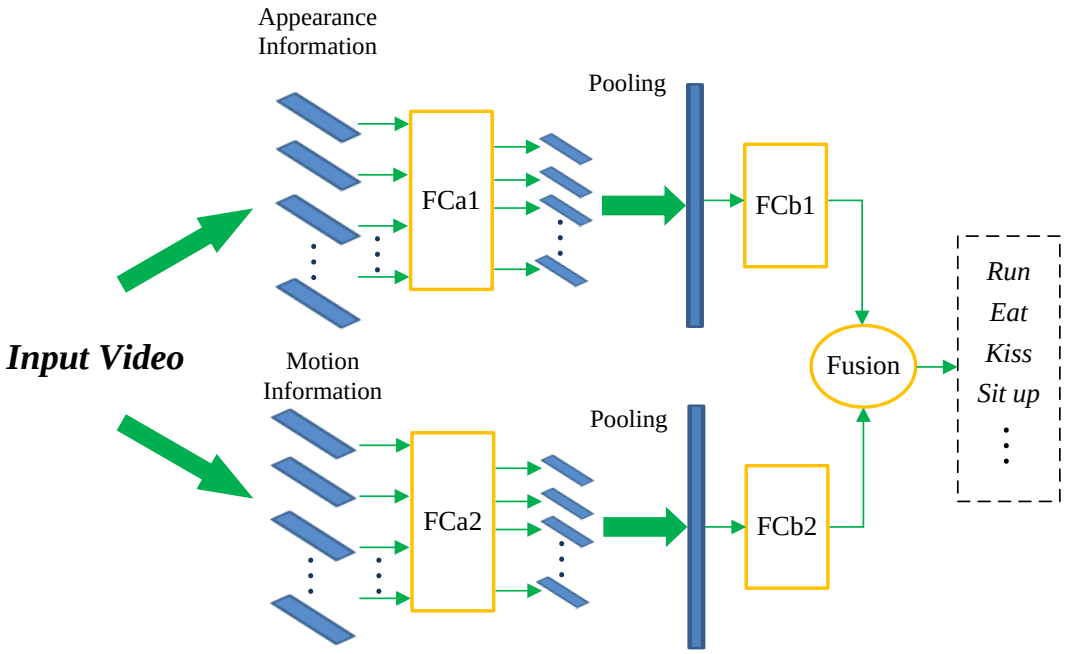


FIGURE 3.5: Illustration of late-fusion.

We independently train these two networks. At the testing stage, we obtain the final output of the fused network by calculating the weighted average of $\mathbf{Y}_{b1}$ and $\mathbf{Y}_{b2}$, the outputs from FCb1 and FCb2 respectively.

3.3.4 Implementation

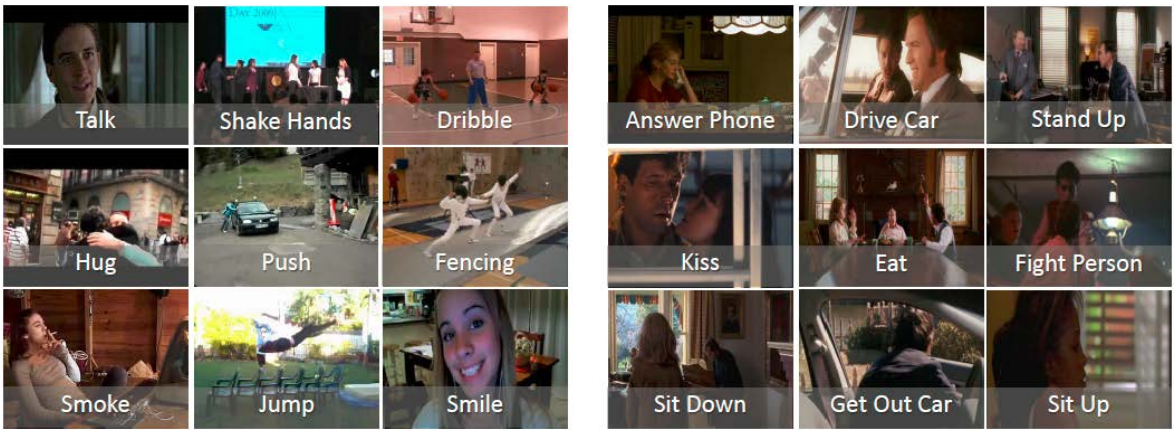
Motion feature

Our network utilizes both raw frame images and motion features as network input. To calculate the motion feature for a given frame, the Fisher vector encoding is applied to the trajectories falling into its neighbouring 11 frames (from -5 to 5). We test the neighbouring sizes from 1 frames and 17 frames and find 11 results in best performance. Following the setting of [107], we set the number of Gaussians to 256 for Fisher vector encoding. While in [107] each trajectory is composed of five descriptors, including HOG, Trajectory, HOF, MBHx and MBHy, we use only HOF and MBH due to their strong discriminative power. Since the Fisher vector is of high dimensionality the supervised feature merging strategy in Section 3.3.2 is applied to further reduce the dimensionality of the frame-level motion features.

Network training

In our work, we initialize the parameters of C1-FC7 using a pre-trained model "*vgg-fast*" [12] and keep them fixed during training. During the training procedure, the parameters of FCa and FCb are learned using stochastic gradient descent with momentum. We set momentum to be 0.9 and weight decay to be 5×10^{-4}. The training includes 25 epochs for all training sets.

3.4 Experimental evaluation



(a) HMDB51 (b) Hollywood2

FIGURE 3.6: Example frames from (a) HMDB51 and (b) Hollywood2.

We conduct a number of experiments on three challenging datasets, Holywood2, HMDB51 and UCF101 to evaluate the performance of the proposed method and analyze the effects of various components of our network structure.

3.4.1 Experimental setup

Datasets Hollywood2 and HMDB51 can be regarded as two challenging action recognition datasets because most existing methods achieve very low recognition accuracy on these two datasets. The difficulties lie in that they contain many complex actions and there are a lot of uncontrolled scene variations within the videos. Fig. 3.6 gives some example frames from HMDB51 and Hollywood2. UCF101 is another standard action recognition dataset which contains a larger number of videos.

The Hollywood2 dataset [72] is composed of video clips from 69 movies and includes 12 classes. It contains a total of 1,707 videos with 823 training videos and 884 testing videos. Training and

testing videos belong to different movies. The performance is measured by mean average precision (mAP).

The HMDB51 dataset [55] is collected from various sources, such as movies, Prelinger archive and YouTube. It contains 6,766 video clips belonging to 51 classes. According to the protocol in [55], three training-testing splits are provided for this dataset. For each class, there are 70 training videos and 30 testing videos. The average classification accuracy over all the classes and splits is reported. This dataset provides two versions, a stabilized one and an unstabilized one. In our experiments, we use the latter version.

The UCF101 dataset [51] is composed of realistic action videos collected from YouTube. It contains 13,320 videos belonging to 101 classes. We use three train-test splits as in the standard evaluation protocol recommended by the dataset provider. For each class, the videos are split into 25 groups in which 7 groups are used for test and the rest are treated as training data. The classification accuracy over all the classes and all the splits are reported as performance measurement.

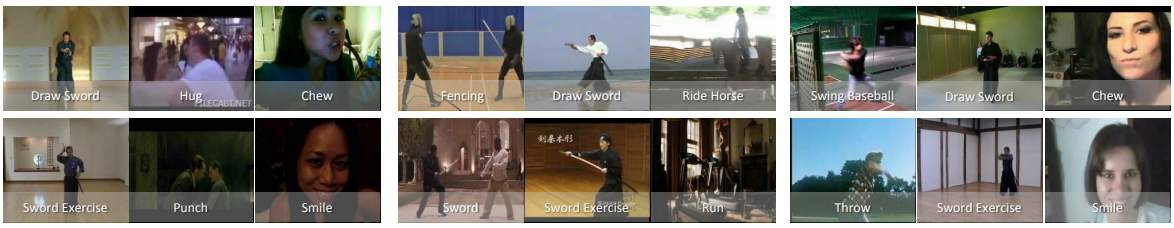
3.4.2 Performance evaluation

In this subsection, we first discuss four aspects related to our network structure, that is, the evaluation of the 'motion part' and the 'appearance part' of our network, the comparison of early fusion and late fusion, the effect of temporal pyramid in the pooling layer and the complementarity between our network and global Fisher vector of IDT. Then we compare the proposed method to the state-of-the-art methods .

3.4.3 Motion vs. appearance

Our network utilizes both appearance information and motion information. In this section, we test these two sources of information separately to demonstrate their impacts on action recognition. In other words, we discard the feature concatenation layer and only choose one type of feature, motion or appearance as our network input. To further demonstrate the effectiveness of our network structure on each type of feature, we also introduce two baselines which use either motion or appearance information.

Baseline 1: The first baseline applies the same CNN that is used for initializing our model parameters C1-FC7 to each frame. The FC7 layer activations are extracted as frame-level features. These features are aggregated through average pooling or temporal pyramid pooling. When temporal pyramid pooling is applied, we use the same pooling parameter as the temporal pyramid pooling layer of



(a) Appearance (b) Motion (c) Combined

FIGURE 3.7: Examples of misclassification for HMDB51 using (a) appearance features, (b) motion features and (c) combined features.

our network. After pooling, a linear SVM is applied to classify the pooled video representation. We denote this baseline as Appearance Average Pooling (AAP in short) and its temporal pyramid pooling variant as (ATP).

Baseline 2: The second baseline adopts frame-level motion feature and creates video-level representation through average pooling or temporal pyramid pooling. To ensure fair comparison, we use HOF and MBH of improved dense trajectory and their dimensionality reduced Fisher vectors as the motion descriptors and frame-level motion feature respectively. We denote this baseline as Trajectory Average Pooling (TAP in short) and its temporal pyramid pooling variant as (TTP).

Table 3.1, 3.2 and 3.3 show the results on Hollywood2, HMDB51 and UCF101. From these three tables we can have the following observations:

- The Motion feature contributes more than the appearance feature in achieving good classification performance. From all three datasets we can see that motion feature significantly outperforms the appearance feature.

- Our network structure works especially well for the appearance feature. In Table 3.1, our method outperforms ATP and AAP by 8% and 10% respectively; In Table 3.2, our method outperforms ATP and APP by 1.7% and 3.8% respectively. We have similar observations on UCF101 where our method outperforms AAP and ATP by 3.3% and 1.6%. Recall that the major difference between ATP and our network (with the output of FC7 as input) is the encoding layer FCa. The superior performance of our network demonstrates the necessity of applying the encoding layer before pooling.

- An interesting observation is that our network structure does not help too much for the motion feature. As can be seen in these three tables, our method achieves comparable performance to TTP, which means that the encoding layer does not lead to too much improvement. This

is probably because the frame-level motion feature we used is already a coding vector, Fisher vector namely, and it is ready for the pooling operation. Thus, adding another encoding layer will not bring too much improvement. In contrast, the output of FC7 is not well-tuned for pooling (recall that the layers before FC7 is pretrained on a CNN without pooling layer), thus adding the encoding layer is beneficial.

- Finally, we observe that adding temporal pyramid into feature pooling can obviously improve the classification performance since it can better describe the temporal structure of videos.

Fig. 3.7 gives some failed examples for HMDB51. As can be seen, these actions tended to be misclassified are very similar in terms of both appearance and motion patterns.

TABLE 3.1: Comparison of the network to baselines on Hollywood2 using appearance information or motion information.

Appearance	AAP	34.2%
	ATP	36.3%
	Ours	**44.4%**
Motion	TAP	60.3%
	TTP	62.7%
	Ours	**62.9%**

TABLE 3.2: Comparison of the network to baselines on HMDB51 using appearance information or motion information.

Appearance	AAP	37.5%
	ATP	39.6%
	Ours	**41.3%**
Motion	TAP	50.9%
	TTP	**53.5%**
	Ours	53.4%

TABLE 3.3: Comparison of the network to baselines on UCF101 using appearance information or motion information.

Appearance	AAP	66.3%
	ATP	68.0%
	Ours	**69.6%**
Motion	TAP	80.0%
	TTP	80.5%
	Ours	**80.9%**

3.4.4 Early fusion vs. late fusion

In this part, we compare two types of fusion methods, namely early fusion and late fusion. While early fusion concatenates motion features and appearance features together as input to train the network, late fusion, as in [96], trains two independent networks using appearance feature and motion feature separately and combines the softmax scores by simple weighted averaging. As can be seen from Table 3.4, both fusion methods boost the classification performance comparing to network with single input, which proves that appearance information and motion information are complementary. Further, early fusion obviously outperforms late fusion, improving the results by around 3% and 2% on Hollywood2 and HMDB51 respectively. Also, we show the average precision for each class of Hollywood2 in Table 3.5. We can see that apart from "Eat", early fusion performs better on all the other actions. The reason may lie in that we train the motion stream and appearance stream independently in the late fusion model without adapting model parameters towards optimal combination of two types of information.

TABLE 3.4: Comparison between early fusion and late fusion. For late fusion, we use $1/3$ appearance weight and $2/3$ motion weight.

Methods	Hollywood2	HMDB51
Late Fusion	64.7%	57.7%
Early Fusion	**67.5%**	**59.7%**

TABLE 3.5: Average Precisions (AP) for each class of Hollywood2. LF represents late-fusion and EF represents early-fusion.

	AnswerPhone	DriveCar	Eat	FightPerson	GetOutCar	HandShake
LF	49.1%	94.1%	58.7%	69.0%	81.8%	40.3%
EF	49.8%	95.3%	54.1%	69.4%	82.0%	51.4%
	HugPerson	Kiss	Run	SitDown	SitUp	StandUp
LF	45.8%	62.6%	86.1 %	76.2%	32.6%	79.8%
EF	55.1%	66.3%	88.2%	76.7%	36.9%	84.7%

TABLE 3.6: The impact of the value of b in the temporal pyramid pooling layer.

Hollywood2	$b = 0$	38.2%
	$b = 3$	44.1%
	$b = 5$	44.2%
	$b = 7$	43.9%
HMDB51	$b = 0$	38.5%
	$b = 3$	39.4%
	$b = 5$	41.3%
	$b = 7$	39.6 %

3.4.5 The impact of the temporal pyramid parameter

In this subsection, we evaluate the effects of temporal pyramid in the pooling layer. Intuitively, adding temporal-pyramid can better cater for the video structure. As in previous experiments, we choose a two-level temporal pyramid structure with one level covering all video frames and another level dividing a video into b segments. Here we evaluate the impact of b. We vary b from 0 to 7, where $b = 0$ means that no temporal pyramid is utilized. To simplify the experiment, we only conduct experiments on appearance features. As can be seen in Table 3.6, adding more segments significantly boosts the results initially and we achieve best performance at $b = 5$. After that peak point continuing to add segments will lead to worse results.

3.4.6 Fusion of temporal pyramid pooling net and global Fisher vector of IDT

In our network structure, we utilize frame-level motion features. To form a frame-level motion representation, we first extract local motion features along trajectories passing this frame and then encode them using Fisher vector. Different from global Fisher vector [107] that describes global motion via aggregating the trajectories over the entire video, frame-level motion representation abstracts local semantic information in temporal domain. In this part, we fuse the video-level representation generated by out network and the global Fisher vector of IDT together to fully exploit their discriminative power.

We adopt encoding level fusion. For the proposed network with both appearance features and motion features as input, we use the outputs of temporal pyramid pooling layer. The representation is then concatenated with the global Fisher vector of IDT. Note that these two sources of representations are $L2$ normalized respectively before concatenation. A linear SVM [32] is used for classification.

TABLE 3.7: Fusion of Temporal Pyramid Pooling Network and global Fisher vector for motion features.

Methods	Hollywood2	HMDB51	UCF101
Global IDT (HOF, MBH)	62.6%	54.7%	82.8%
Temporal Pyramid Net	68.4%	60.8%	87.6%
Fusion	**71.9%**	**63.1%**	**89.1%**

Table 3.7 gives the fusion results. Due to limitation of computational power, we reduce the dimension of frame-level motion representation from 76800 to 20000 to make the network training feasible. However, our network can still achieve better performance than high dimensional Fisher vectors. More importantly, we can see when combining these two kinds of representations together the recognition performance is boosted by a large margin which proves that these two kinds of representations can compensate each other in describing the actions in the videos.

3.4.7 Comparison to state-of-the-art

Hollywood2

Table 3.8 compares our method to several leading methods on this dataset. As can be seen, our method achieves better performance on this dataset. Compared to improved dense trajectory [107], we have

gained more than 7% improvement. The most competitive one to our method is the motion part regularization [77] which is outperformed by our method by about 5%. It can also be observed that motion features of high dimensional perform better than low dimensional features. And our network is strongly complementary to the gobal Fisher vector of IDT.

TABLE 3.8: Experimental results on Hollywood2. While LD means low dimension (4096) for frame-level motion features, HD indicates high dimension (20000) for frame-level motion features.

Dense Trajectory [106]	58.5%
Mathe *et al.* [73]	61.0%
Actons [130]	61.4%
Jain *et al.* [44]	62.5%
Improved Dense Trajectory [107]	64.3%
Motion Part Regularization [77]	66.7%
Ours (LD)	67.5%
Ours (HD)	68.4%
Ours best	**71.9%**

HMDB51

Table 4.5 compares our method to several state-of-the-art methods on HMDB51. Hybrid improved dense trajectories in [81], employs multiple unsupervised encoding methods i.e. Fisher vector [84], VLAD [45] and LLC [109]. Note that the best performed method, stacked Fisher vector [82] employs two-level Fisher vector encoding and concatenates them together as video representation.

We also compare our method to the work in [96] which is also a CNN based method and adopts frame sampling to handle the issue of video-length variation. Our method outperforms it. Note that in their experiment they combine HMDB51 and UCF101 [51] as the training set while our model is trained only on HMDB51. Besides better performance, we believe our network offers a more principled solution to handle the video-length variation issue.

UCF101

Table 4.6 demonstrates the results on UCF101. As can be seen, our method obtains better or comparable performance comparing to some other CNN based methods e.g., Spatio-temporal CNN [50],

TABLE 3.9: Experimental results on HMDB51. While LD means low dimension (4096) for frame-level motion features, HD indicates high dimension (20000) for frame-level motion features.

Spatio-temporal HMAX Network [55]	22.8%
Dense Trajectory [106]	47.2%
Jain *et al.* [44]	52.1%
Multi-view super vector [9]	55.9%
Improved Dense Trajectory [107]	57.2%
Hybrid Improved Dense Trajectory [81]	61.1%
Stacked Fisher vector [82]	**66.8%**
Trajectory Pooled Descriptors [111]	65.9%
Two-stream ConvNet (average fusion) [96]	58.0%
Two-stream ConvNet (SVM fusion) [96]	59.4%
Motion Part Regularization [77]	65.5%
Ours (LD)	59.7%
Ours (HD)	60.8%
Ours best	63.1%

Two-stream ConvNet [96] and Deep Net [121] which are most relevant to our method. Among the comparing methods, Trajectory Pooled Descriptors [111] performs best. The reason lies in that it aggregates the convolutional features via the guidance of the extracted trajectories. We may incorporate similar strategy to replace the handcrafted local appearance descriptor, e.g., HOG, which was used in improved dense trajectory with more discriminative convolutional features.

3.5 Summary

In this work, we build a CNN architecture upon the frame-level representations of a video for video classification. It is a hybrid network that works with both appearance feature and motion feature. The key of the network is a module composed of an encoding layer and a temporal pyramid pooling layer. While the encoding layer maps the frame-level representations into a space more suitable for pooling, the temporal pyramid pooling aggregates the local encoding vectors into a video signature.

TABLE 3.10: Experimental results on UCF101. HD indicates high dimension (20000) for frame-level motion features.

Spatio-temporal CNN [50]	63.3%
Multi-view super vector [9]	83.5%
Improved Dense Trajectory [107]	85.9%
Hybrid Improved Dense Trajectory [81]	87.9%
Two-stream ConvNet (average fusion) [96]	86.9%
Two-stream ConvNet (SVM fusion) [96]	88.0%
Trajectory Pooled Descriptors [111]	**91.5%**
Deep Net [121]	88.6%
Ours (HD)	87.6%
Ours best	89.1%

The network structure is flexible that can take a video of any length and represented by any frame-level form as input. But applying the encoding directly to the frame-level representation will result in a large-number of parameters especially when dealing with high-dimensional frame-level features. To address this problem we propose another network structure that extracts the dynamic information in a more economic way. The new network structure will be elaborated in the next Chapter.

Chapter 4

Order-aware Convolutional Pooling for Video Classification

4.1 Overview

In previous chapter, we propose a network structure to model the frame-level features of a video. The network performs encoding on these features and adopts a pooling operation to aggregate these coding vectors. The method suffers from the following two drawbacks: 1) encoding the high-dimensional frame-level features require a large number of parameters and 2) temporally pooling the features extracted at every frame completely or partially ignore the dynamic information contained in the temporal domain, which may undermine the discriminative power of the resulting video representation since the video sequence order could unveil the evolution of a specific event or action. To overcome this drawback and explore the importance of incorporating the temporal order information, in this work we propose a novel temporal pooling approach to aggregate the frame-level features. Specifically, we propose to apply the temporal convolution operation to the frame-level representations to extract the dynamic information. However, as discussed directly implementing this idea on the original high-dimensional feature will result in parameter explosion. To handle this issue, we propose to treat the temporal evolution of the feature value at each feature dimension as a 1D signal and learn a unique convolutional filter bank for each 1D signal. By conducting experiments on HMDB51 and UCF101, we demonstrate that the proposed method is superior to the conventional pooling methods.

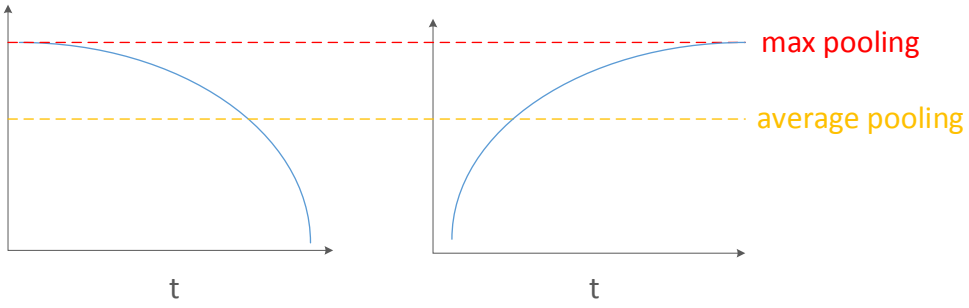


FIGURE 4.1: Conventional pooling for 1D signal.

4.2 Introduction

Video is composed of a sequence of frames and the frame sequence reflects the evolution of the video content. Thus, a video can be naturally represented by a sequence of frame-level features which may describe either the visual patterns or motion patterns at a specific time step. To generate a vectorized video representation, a common practice is to apply temporal pooling, e.g., average or max pooling, to the frame-level features. However, these temporal pooling methods are problematic because they completely ignore the frame order and consequently lose the dynamic information contained in the temporal domain. In other words, the same pooling result will be obtained after randomly shuffling the frames. The frame-order, however, plays an important role in identifying actions or events in the video because it unveils the evolution of the video content. Fig. 4.2 shows some sampled order-preserving frames of two videos describing "sit" and "stand up" respectively. As can be seen, the frame order reflects the trend of the actions and encodes crucial discriminative information for distinguishing these two actions. A remedy to direct temporal pooling is to adopt temporal pyramid pooling which coarsely considers the temporal structure of a video by partitioning a video into a set of segments and deriving the representation of the video by concatenating these segment-level representations. It, however, still undergoes the loss of local dynamic information within each segment.

To better capture the frame order information for action recognition, we propose a novel temporal pooling method to aggregate the frame-level representations. Our method is inspired by the use of the convolutional neural network (CNN) for image classification. In image classification, a CNN applies convolution operation to the local regions of the image to extract some discriminative visual patterns and uses pooling to make the representations invariant to some variations. Similarly, a video can be regarded as an image with the image height being one, the image width being the length of the video and the number of image channels being the dimensionality of the frame-level features. Then we can apply convolution to the videos in the temporal domain to extract some discriminative

FIGURE 4.2: Order-preserving sample frames from two actions. Upper: Sit. Lower: Stand up.

patterns contained within the local temporal interval. However, when the frames are represented by high-dimensional features, such as CNN features or high-dimensional encodings (e.g., Fisher vector) of motion features, directly implementing this idea will lead to parameter explosion. The reason is twofold: (1) The number of parameters per filter equals to the frame-feature dimensionality times the size of local temporal interval; (2) for high dimensional frame features, a large number of filters will be needed to capture the useful information. For example, when the frame-feature dimensionality is 10,000 and the interval size is 5 frames, 4,000 filters may be needed and this setting will result in about 200 million parameters in such a convolutional layer. Training a network with such a large number of parameters will incur overwhelming computational cost and increase the risk of over-fitting especially when a limited number of training videos are available.

To address this issue, we propose to inspect the video from an alternative way, that is, we treat the temporal evolution of the feature value at each dimension of the frame-level features as a 1D signal. And the key of our method is to learn a set of filter banks for the 1D temporal signals in a supervised fashion. The filter bank for each feature dimension is unique and it servers as detectors to identify the discriminative local temporal patterns along the 1D temporal signal. After applying the filter banks to all the 1D signals, their filter responses are aggregated via a conventional temporal pooling method, i.e. average-pooling, max-pooling or temporal pyramid pooling to obtain the video-level representation.

Our method is advanced to the conventional pooling methods like max or average pooling since the latter ones only capture some simple statistics e.g., max value or direct current (DC) component of the 1D signals. As illustrated in Fig. 4.1, these two 1D signals cover opposite temporal information

but conventional pooling methods will obtain the same pooling results because they have the same max value and DC value. In comparison, the proposed method can distinguish these two 1D signals by learning a filter to look into the local evolution trend. Also, compared with the straightforward implementation which learns a filter with all frame-feature dimensions involved, the proposed method strategy significantly reduces the number of model parameters while still being able to capture the frame order information. For example, when the feature dimensionality and interval size are 10,000 and 5 respectively and 3 filters are adopted for each dimension, the number of parameters reduces to about 150,000 which is far less than that required in the straightforward implementation. By conducting experiments on two challenging video-based action recognition datasets, HMDB51 and UCF101, we demonstrate that the proposed method achieves superior performance to the conventional pooling methods.

4.3 Proposed pooling method

The general idea of the proposed order-aware convolutional pooling is shown in Fig. 4.3. First, it extracts either appearance features or motion features from each video frame. Then a convolutional filter bank is learned and applied to each feature dimension over the temporal domain. The filter response signals of each dimension are aggregated as the dimension-level representation. Finally, these dimension-level representations are concatenated together as the video representation for classification.

4.3.1 Frame-level Representation Preparation

Appearance information and motion information embody different characteristics of the videos and they can compensate each other to describe a video. To take advantage of both information sources, we represent a video frame by concatenating both the appearance features and the motion features. We use CNN features of each video frame as the frame-level appearance features considering the proven success of CNN achieved in the field of image classification. Specifically, each frame is fed into a CNN model [54] pre-trained on ImageNet [24] and the activations of the second fully-connected layer are used as its appearance features. For motion features, we resort to improved dense trajectory (IDT) [107] considering its good performance for action recognition. Originally, IDT was proposed to generate a video representation which is obtained by aggregating the Fisher vectors of all the trajectories over the entire video by sum pooling. To create the IDT representation of a video frame,

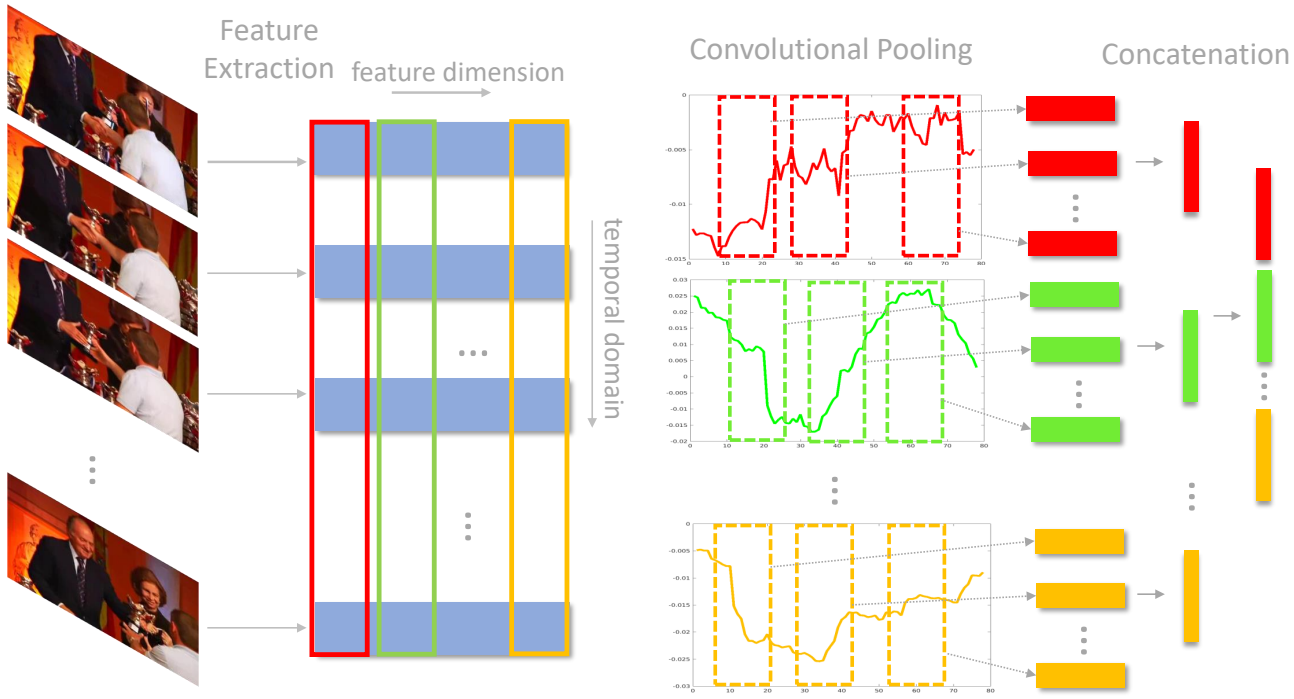


FIGURE 4.3: Illustration of order-aware pooling.

we first encode the trajectories passing this frame using Fisher vector encoding and then aggregate the coding vectors of these trajectories into a frame-level motion feature. The Fisher vector tends to be high dimensional which makes network training computational expensive. We use the dimensionality reduction method introduced in Chapter 3 to reduce the dimensionality of the frame-level features.

4.3.2 Order-aware Convolutional Pooling

After feature extraction, a video is represented by a sequence of frame-level features. The objective next is to design a pooling method that can benefit from the frame order of the videos. Recall that CNN makes use of the spatial structure of an image by applying convolution operation at different local regions of an image, our general idea is to apply convolution to the frame-level features over the temporal domain to make use of the 1D temporal structure (frame order) of the videos.

In image based CNN, a convolutional layer is composed of a filter bank and a nonlinear activation function. The filters in the filter bank can be regarded as some detectors which are applied to different local regions of an image to extract some discriminative patterns and non-linear function is applied to the responses of the filers to introduce nonlinearity to the neural network. Suppose the input of a convolutional layer are K feature maps with the size of $H \times W$ from the previous convolutional layer, where H, W denotes the height and width of the feature map respectively. A convolutional

filter is operated on a small spatial support of the input feature maps, say, a small local region with the size of $h \times w$. For each local region, the convolutional layer computes $f(\left(\sum_{k=1}^{K} \mathbf{W}_k^T \mathbf{r}_{ij}^k\right) + \mathbf{b})$, where $\mathbf{r}_{ij}^k$ denotes the flatten version of the activations within the $h \times w$ region at the kth feature map and its dimensionality will then be $h \times w$. The function $f(\cdot)$ is a non-linear function such as ReLU ($f(x) = max(0, x)$). The parameters $\{\mathbf{W}_k\}$ and $\mathbf{b}$ are to be learned during network training. Assuming the number of filters adopted in a convolutional layer is n, the total number of parameters of this layer is $w \times h \times K \times n + n$.

Inspired by the capability of CNN in making use of the spatial structure of the 2D image, we study applying convolution on action recognition via exploiting the temporal information of the videos. Recall that we represent a video as a sequence of frame-level feature vectors. For such a representation, we can treat our video representation as a special case of image representation with image height being 1, image width being the length of the video and the number of image channels being the dimensionality of the frame-level features. Then analogous to the convolutional operation adopted on images, we can learn some filters to extract the discriminative patterns within different temporal local regions. Suppose the dimensionality of the frame-level features is K, for the tth interval with the length being l, the convolutional operation computes $f(\left(\sum_{k=1}^{K} \mathbf{W}_k^T \mathbf{r}_t^k\right) + \mathbf{b})$ ($\mathbf{W}_k \in \mathbb{R}^{l \times n}$), where $\mathbf{r}_t^k$ is a l-dimensional vector with its dimensions corresponding to the feature values at the kth dimension within the interval. Similar to the way of calculating the number of parameters in 2D images, the number of model parameters of such a convolutional layer will be $l \times K \times n + n$. Since the video frames are usually represented by high-dimensional features e.g., fully-connected CNN features or high-dimensional coding vectors, a large number of filters will be needed to capture the useful information and this will result in parameter explosion. Assuming that the interval size here is 8, the number of filters adopted is 4,000 and the dimensionality of the frame-level features is 10,000, the total number of parameters involved is about **320,000,000**. Training a model with such a huge number of parameters will incur prohibitively high computational cost as well as increase the risk of over-fitting.

To address this problem, in this work, we inspect the video from an alternative way. That is we treat the feature value evolution of one feature dimension over the temporal domain as a 1D temporal signal as shown in Fig. 4.3 and represent a video as K independent such 1D signals. The rationality behind is that for many high-dimensional features such as Fisher vector, the correlation between different feature dimensions tend to be small [125]. For each of such 1D signals, we learn a unique filter bank and similar to the 2D convolution at each convolution step these filter banks operate on a local temporal interval, that is, the filter output at time t is calculated as $f(\mathbf{W}_k^T \mathbf{r}_t^k + \mathbf{b}_k)$. Similar to

the 2D case, the term $\mathbf{r}_t^k$ denotes the vectorized representation of the tth interval at the kth feature dimension and its dimensionality equals l, the size of the temporal interval. In this case, since the filter bank is applied only to very low dimensional interval vectors, the number of filters required will be dramatically reduced, e.g. reducing from 4000 to 3. Consequently, the number of model parameters will be far less than that involved in the aforementioned straightforward implementation. Let's assume that the number of filters for each dimension-wise filter bank is $\bar{n}$, then the total number of model parameters will be $l \times K \times \bar{n} + K \times \bar{n}$. Assuming again that the interval size is 8, the number of filters adopted for each 1D signal is 3 and the dimensionality of the frame-level feature is 10,000, the total number of parameters involved will become about **240,000**, only being $1/1000$ of that in the straightforward implementation.

The output of the convolution operation of each 1D signal is a set of filter response vectors at different temporal locations. Obviously, the number of responses varies with the length of the videos. Considering that a fixed-length video representation is required for video classification, the pooling operation is employed to aggregate the varying number of response vectors of each feature dimension into a fixed-length dimension-level representation.

To explicitly take into consideration the long-range temporal structure of the videos, we propose to use the temporal pyramid pooling to aggregate the local filter responses. Fig. 4.4 shows a three-level temporal pyramid pooling. The first level pools all the filter responses of a feature dimension directly over the temporal domain. For the ith level, the filter responses are partitioned into m_i segments temporally and within each segment we perform max pooling. Then the representations of each segment will be concatenated together to from the representation for this dimension. So if the dimensionality of each segment-level representation is d, the dimensionality of the ith level will be $m_i \times d$ and the dimensionality of the dimension-level representation will be $d\sum_{i=1}^{L} m_i$, where L is the number of levels used in the temporal pyramid pooling. After pooling the local responses, each dimension is represented by a fixed-length vector and the dimension-level representations are concatenated together to generate the representation of the video. Formally, the video representation can be expressed as follows:

$$\begin{aligned} \mathbf{P} &= [\mathbf{P}_1^T, \mathbf{P}_2^T, \cdots, \mathbf{P}_k^T, \cdots, \mathbf{P}_K^T]^T, \\ \text{where, } \mathbf{P}_k^T &= [\mathbf{P}_{k_1}^T, \mathbf{P}_{k_2}^T, \cdots, \mathbf{P}_{k_L}^T]^T, \end{aligned} \tag{4.1}$$

where $\mathbf{P}_{k_j}$ is the representation of the jth level of the kth dimension and K is the dimensionality of the frame-level representation.

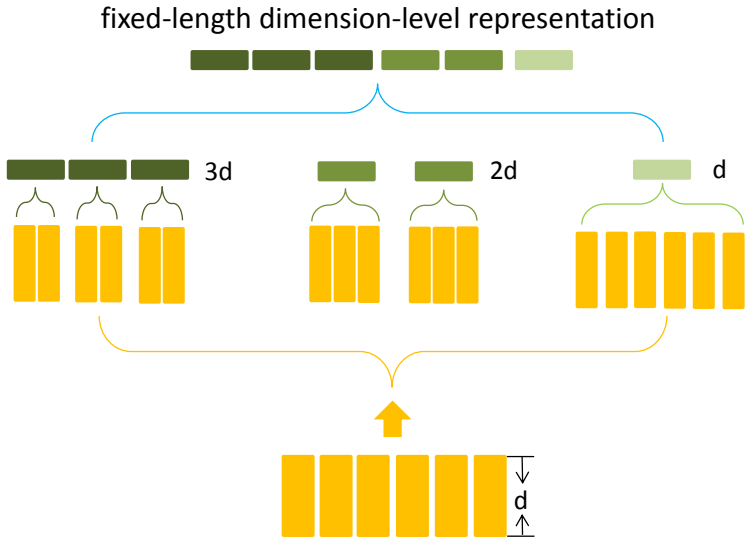


FIGURE 4.4: Illustration of temporal pyramid pooling.

4.3.3 Classification and Model Parameter Learning

We learn the model parameters in a supervised fashion, that is, we add a classification layer on top of the outputs of the proposed pooling layer. It calculates $\mathbf{Y} = \varphi\left(\mathbf{W}_c\mathbf{P} + \mathbf{b}_c\right)$ where $\mathbf{W}_c$ and $\mathbf{b}_c$ are model parameters that will be learned during network training and φ is the softmax [1] operation. The output $\mathbf{Y}$ is a probability distribution indicating the likelihood of a target label appearing in a video. In the training stage, we use the following loss function to measure the compatibility between this distribution and ground-truth class label:

$$L(\mathbf{W}, \mathbf{b}) = -\sum_{i=1}^{N} \log(\mathbf{Y}(c_i)), \tag{4.2}$$

where c_i denotes the class label of the ith video and N is the total number of training videos. Recall that $\mathbf{Y}$ is a c-dimensional vector and c equals to the number of classes. Here we use $\mathbf{Y}(c_i)$ to denote the value at c_ith dimension of $\mathbf{Y}$. Using Stochastic Gradient Descent (SGD), in each step we update model parameters by calculating the gradient of an instance-level loss $L_i(\mathbf{W}, \mathbf{B}) = -\log(\mathbf{Y}_b(c_i))$.

4.4 Experimental evaluation

The evaluation is performed on two datasets, HMDB51 [55] and UCF101 [51], which have been introduced in Chapter 3.4.

4.4.1 Experimental setup

Parameter setting

In the experiments, the parameters are set as follows unless otherwise stated. The interval size for the filters is set to be 8 and the interval stride is set to be 1. The number of filters adopted for each feature dimension is 3. The level of temporal pyramid is fixed to be 2 in the pooling layer when temporal pyramid pooling is employed. When SVM is applied in the experiments for classification, we fix $C = 100$. Due to the redundancy between consecutive frames of a video, we sample 1 frame from every 5 frames for our method. Thus a filter in our method actually covers a range of 40 frames in the original video since the interval size of our filter is 8.

Appearance feature

We utilize the 4,096-D activations of the second fully layer of AlexNet [54] (a deep CNN model pre-trained on ImageNet) provided in Caffe [47] as frame-level appearance features. Using better models such as "vgg-deep" [97] as feature extractors can further boost the classification performance. However, for fair comparison with some existing methods [96, 50] that are relevant to our work, we choose the shallower AlexNet model [54].

Motion feature

We adopt the improved dense trajectory (IDT) [107] as our frame-level motion features due to its good performance in action recognition. Different from [107] that aggregates all the trajectories of a video into video-level representation using Fisher vector encoding, we aim at obtaining frame-level representation to make it compatible with the proposed network. To obtain the motion feature of a frame, we consider the trajectories falling into a local neighbourhood of a frame with the size of the temporal neighbourhood being 10 (5 frames before and after a frame separately). We encode these trajectories using Fisher vector coding with 256 Gaussians and the coding vectors are aggregated by sum pooling to form the frame-level representation. In this work, we extract HOF (108D) and MBH (196D) descriptors only to describe the trajectories. Since the Fisher vector is of high dimensionality (76800-D here) and this will make computation expensive or even infeasible. We adopt two treatments to address this problem. First, we only keep the "mean" part of the Fisher vector and remove the "covariance" part to reduce the dimensionality into half of its original implementation. Second, we use the dimensionality reduction method introduced in Section 4.3.1 to reduce the dimensionality to 10,000.

4.4.2 Performance evaluation

In this subsection, the comparisons to the baselines on both appearance features and motion features will be given first to evaluate the effectiveness of the proposed pooling method. Then we investigate some other important aspects and/or properties related to our method, including the influence of the number of filters on classification performance and the complementarity between the proposed pooling method and the unsupervised global pooling used in IDT. Finally, we compare our method to the state-of-the-art.

Comparison to baselines

Both appearance feature and motion feature are employed to represent the video frames. In this subsection, we evaluate the efficacy of the proposed pooling method on these two types of features separately.

Baselines for the appearance feature: We now compare our method to the baselines using the frame-level appearance features. For the first two baselines, the frame-level CNN features are aggregated via average pooling (AP) or max pooling (MP) to obtain the video-level representations. After pooling, linear SVM [32] is employed for classification. For the third baseline, we adopt the pooling method proposed in [90] which combines several pooling strategies together to capture the dynamic information. For the fourth baseline, temporal pyramid pooling (TP) is used to form the representation of a video, where max pooling is chosen to aggregate the frame-level features within each temporal segment and the configuration of TP is the same as that used in our method.

Baselines for the motion feature: Similar to appearance features, we apply average pooling, max pooling and temporal pyramid pooling to the frame-level motion features to create the video-level representations. The frame-level motion features are obtained in the same way as our method.

Table 4.1 and 4.2 demonstrate the results on HMDB51 and UCF101. From the results, we have the following observations:

- Motion features can lead to better classification performance comparing to appearance features. On both datasets, a method using motion features can outperform its counterpart that uses appearance feature by more than 10%. This observation indicates that motion information plays a more important role in discriminating the actions in the videos.

- On appearance features, the proposed pooling method can consistently outperform the baselines. In table 4.1, the network adopting max pooling outperforms AP, MP and TP by 3.3%,

TABLE 4.1: Comparison of the proposed pooling method to the baselines on HMDB51 using appearance information or motion information.

Appearance	AP	37.5%
	MP	36.5%
	PoT (no TP) [90]	36.5%
	TP	39.2%
	Ours (MP)	**40.8%**
	Ours (TP)	**41.6%**
Motion	AP	50.9%
	MP	50.6%
	TP	54.7%
	Ours (MP)	52.8%
	Ours (TP)	**55.0%**

4.3% and 1.6% respectively. In Table 4.2, our method with max pooling outperforms AP, MP and TP by 3%, 1.9% and 0.8%. Note that the method in [90] does not gain any improvement to max pooling which indicates that its pooling strategies e.g., histogram of change and gradients' pooling, are suited to first-person videos only. These results justify the advantage of the proposed network over direct pooling in capturing the dynamic information contained in the appearance features of the videos. Another observation is that after introducing temporal pyramid pooling into our network, the performance can be further boosted. Specifically, the classification accuracy is improved by 0.8% on HMDB51 and 1% on UCF101, which reveals the benefit of using temporal pyramid pooling in our method to capture the long-range information.

- When working with motion features, our pooling method can obviously outperform AP and MP and achieve slightly better performance than TP. In Table 4.1, our method with max-pooling only gains improvement of 1.9% and 2.2% over AP and MP respectively. In Table 4.2, our method with max-pooling outperforms AP and MP by 1% and 0.8% respectively. Again, these

TABLE 4.2: Comparison of the proposed pooling method to the baselines on UCF101 using appearance information or motion information.

Appearance	AP	66.3%
	MP	67.4%
	PoT (no TP) [90]	67.5%
	TP	68.5%
	Ours (MP)	**69.3%**
	Ours (TP)	**70.4%**
Motion	AP	80.0%
	MP	80.2%
	TP	81.6%
	Ours (MP)	81.0%
	Ours (TP)	**82.1%**

observations prove the importance of incorporating the frame order information. When temporal pyramid pooling is applied, performance of both the baseline methods and the proposed method are improved and our method obtains slightly better performance than TP on the two datasets. The advantage is not as significant as that on appearance features. This is probably due to that the frame-level motion features have already encoded the dynamic information contained within adjacent frames, applying convolution on them cannot obtain significant improvement as on static appearance features.

The impact of the number of filters

The convolution operation constitutes the most important part of the proposed pooling method. In this part, we evaluate the impact of the number of filters. Specifically, we focus on investigating the change of classification performance w.r.t the number of filters adopted. We use frame-level appearance features and max pooling here. In our method, the interval size and interval stride are fixed to be 8 and 1 respectively and we choose three values 1, 3, 5 as the number of filters for each dimension. Table 4.3 shows the results.

As can be seen from the results on HMDB51, when only one filter is used the performance is unsatisfactory which means that one filter is insufficient to capture the dynamics contained in the given temporal interval of a feature dimension. Increasing n can improve the performance and the best performance is obtained when $n = 3$. After that, continuing to increase the number of filters leads to worse performance, which may be due to overfitting. On UCF101, using one filter produces worst performance again. However, unlike in HMDB51 the best performance is achieved when $n = 5$. The reason may be that UCF101 has much more training data which makes the model training less prone to overfitting.

TABLE 4.3: The impact of the number of filters n in the proposed pooling method (4096-D frame-level appearance features are used for this evaluation.)

HMDB51	$n = 1$	38.9%
	$n = 3$	**40.8%**
	$n = 5$	39.5%
UCF101	$n = 1$	67.8%
	$n = 3$	69.3%
	$n = 5$	**69.6%**

Performance on combined features

In this part, we evaluate the performance of the proposed method when using both the aforementioned appearance features and motion features. More specifically, for each video frame we perform $L2$ normalization to the appearance feature and the motion feature respectively and concatenate the normalized features together as the frame-level representation. Since our method uses both the CNN features and IDT based motion features, it is fair to compare another baseline which concatenates the max-pooled frame-level CNN features and the global motion features. Here the global motion feature is obtained via aggregating the Fisher vectors of the trajectories over the entire video as in IDT[107]. Similarly, the CNN features and global motion features are $L2$ normalized separately before concatenation. Linear SVM is used for classification. Table 4.4 shows the results. As can be seen from this table, our method can outperform the baseline by 1.6% and 0.9% on HMDB51 and UCF101 respectively. When combined with the unsupervisedly pooled motion features, our performance can be

significantly boosted further. This observation shows that the representations learned by our method is strongly complementary to the representation obtained via an unsupervised pooling method.

TABLE 4.4: Performance on combined features. For IDT, we use HOF and MBH only.

Methods	HMDB51	UCF101
CNN (max) + Global Motion Pooling	59.4%	86.9%
Ours	61.5%	87.8%
Ours + Global Motion Pooling	**64.1%**	**89.6%**

Comparison to State-of-the-art

In this part, we compare our method to the state-of-the-art methods on HMDB51 and UCF101. Note that, our performance can be further boosted by some strategies like employing a better CNN model [97] to extract CNN features or using higher dimensional motion features.

HMDB51 Table 4.5 compares our method to several state-of-the-art methods on HMDB51. As can be seen, our method achieves the second best performance. Hybrid improved dense trajectories in [81], employs multiple unsupervised encoding methods i.e. Fisher vector [84], VLAD [45] and LLC [109]. In comparison, our method is much more elegant in the sense that it relies on a single encoding module. Note that the best performed method, stacked Fisher vector, [82] employs two-level Fisher vector encoding and concatenates them together as video representation. The work [96] is a CNN based method and adopts frame sampling to handle the issue of video-length variation. The video evolution method [35] captures the evolution of the content within the video via learning a ranking functions to rank frames.

UCF101 Table 4.6 shows the results on UCF101. We first compare our method to the LRCN [28] which utilizes the LSTM to aggregate the frame-level CNN features for action recognition. Our method outperforms it by 2%. As can be seen in the lower part of Table 4.6, our method performs best among these methods. The spatio-temporal convolution based method [50] performs worse than dense trajectory based methods [9, 107, 81]. Our method can outperform two-stream CovNet [96] by 1.6%. The Deep net [76] stacks Long Short-Term Memory (LSTM) cells on top of a CNN for video classification. Still, we can achieve better performance than that.

TABLE 4.5: Experimental results on HMDB51.

Spatial-temporal HMAX network [55]	22.8%
DT [106]	47.2%
Jain *et al.* [44]	52.1%
DT+MVSV [9]	55.9%
IDT [107]	57.2%
Hybrid IDT [81]	61.1%
Stacked Fisher Vector [82]	**66.8%**
Two-stream ConvNet (average fusion) [96]	58.0%
Two-stream ConvNet (SVM fusion) [96]	59.4%
Video Evolution [35]	63.7%
Factorized Networks [101]	59.1%
Actionness [70]	60.4%
Ours	**64.1%**

4.5 Summary

In this chapter, we propose another CNN based method , called order-aware convolutional pooling, to model the local features of a video. Comparing to temporal pyramid pooling CNN in Chapter 3, it enjoys the following two advantages. First, it explicitly explores the dynamic information contained within a video by applying convolution operation over the temporal domain. Second, it can maintain a controllable amount of model parameters by treating the video in a novel way. Benefiting from this strategy, this method does not only improves the computational efficiency but may reduce the risk of overfitting.

The method is a general pooling method that can be applied to other time-series tasks like speech recognition and text classification where the speech signal or textual sentence can be represented by a sequence of feature vectors. We will evaluate our method in these tasks in the future.

TABLE 4.6: Experimental results on UCF101.

LRCN [28] (LSTM + CNN)	68.2%
Ours CNN	**70.4%**
Spatio-temporal CNN [50]	63.3%
DT+VLAD [9]	79.9%
DT+MVSV [9]	83.5%
IDT [107]	85.9%
Hybrid IDT [81]	87.9%
Two-stream ConvNet (average fusion) [96]	86.9%
Two-stream ConvNet (SVM fusion) [96]	88.0%
Deep Net [76]	88.6%
Factorized Networks [101]	88.1%
EMV+RBG-CNN [122]	86.4%
Dynamic Image Networks [6]	89.1%
Ours	**89.6%**

Chapter 5

Hybrid Generative Model Based Fisher Vector Coding for Image Classification

5.1 Overview

In last two chapters, we develop two supervised encoding and pooling methods for video classification. In this chapter, we present a novel encoding method which applies to, but is not limited to, image classification.

Supervised encoding and Fisher vector encoding are two representative schemes to create image representations. Both of them can achieve state-of-the-art image classification performance but through different strategies: the former extracts discriminative patterns from local features at the encoding stage while the latter preserves rich information into high-dimensional signatures derived from a generative model of local features. In this paper, we propose a hybrid encoding method which combines the strategies from both encoding methods. The key idea is to leverage supervised encoding to decompose local features into a discriminative part and a residual part and then build a generative model based on this decomposition. Realizing this model by following the recently proposed sparse coding based Fisher vector encoding (SCFV) framework, we derive the Fisher vector of the proposed generative model and produce a novel encoding method called hybrid generative model based Fisher vector encoding (HGMFV). We further apply this encoding method to the regional CNN features and conduct experiments on four popular datasets (UCSD-Birds-200, MIT-67, PASCAL07 and PASCAL12). Our experiment demonstrates that this hybrid encoding approach is superior over SCFV and conventional Fisher vector encoding as well as the supervised encoding method from which our hybrid encoding is derived.

5.2 Introduction

The performance of an image classification system largely depends on its image representation. Currently, there are two representative image representation creation schemes. One is supervised encoding, which includes supervised codebook learning [59, 63, 48, 118, 112], middle-level feature mining [26, 49, 62] and supervisedly trained convolutional neural networks (CNN) [54]. Those methods extract discriminative patterns from input images or input local features and create image representations by pooling the responses of the coding vectors or filters or detectors (the name varies in different contexts). Another one is Fisher vector encoding, which generates high-dimensional signatures from an unsupervised generative model learned for local features. The local features can be either traditional local features such as SIFT [68] or more advanced deep CNN activations [39, 16, 66, 25].

Both methods demonstrate state-of-the-art performance [39, 16, 66, 25, 89] on various visual classification problems. However, the underlying principles of those two schemes are very different. The former aims to identify the discriminative patterns at the early encoding stage of an image classification pipeline while the latter tries to encode as much information of local features as possible into the high-dimensional image representation and relies on classifier learning to identify the discriminative patterns. In general, supervised encoding can achieve good classification performance with low-dimensional image representations. Fisher vector, in contrast, relies on much higher dimensional representations but can achieve better performance on problems that do not have a large number of labelled training samples [39, 16, 66, 25].

In this chapter, we propose a hybrid method to combine the ideas from both methods. The key insight is that supervised encoding provides a mechanism to decompose a local feature into a discriminative part and a residual part. We can then build a generative model by taking this decomposition into account and derive its corresponding Fisher vector. This Fisher vector encodes both the discriminative information from supervised encoding and other potentially useful information that may not be captured by supervised encoding into the high-dimensional signature. Thus, better classification performance can be expected by using it to create image representations.

To build such a generative model, we draw inspiration from the recently proposed sparse coding based Fisher vector (SCFV) in [66]. In SCFV, a local feature is assumed to be generated from a Gaussian distribution which has a random mean vector. This mean vector is sampled from the linear combination of a set of bases and the combination coefficient is a latent random vector. Our model follows this framework but we step further to assume that a local feature can be decomposed into a discriminative part and a common part which is achieved by using a pre-trained supervised encoding

model. Then we utilize two sets of bases to model these two parts. This treatment produces a novel generative model and it allows us to derive a new Fisher vector encoding from it, which is called hybrid generative model based Fisher vector encoding (HGMFV). To speed up the calculation, we further leverage the matching pursuit algorithm to approximately solve the inference problem at both the Fisher vector computing stage and the bases learning stage.

To demonstrate the advantage of the HGMFV, we apply it to a bag-of-local-CNN-activations based classification system [39, 16, 66, 25] which has recently shown to be a promising strategy to create image representations. We further conduct experiments on MIT-67, PASCAL-07, PASCAL-12, and Birds-200 datasets and compare our method against SCFV and the supervised encoding approach. The experiment suggests that our method achieves superior performance over competitive methods.

5.3 Preliminary

5.3.1 Supervised encoding

In this work we assume that the coding vector generated from a supervised encoding method is sparse. Generally speaking, this is a reasonable assumption since many supervised encoding methods explicitly enforce the sparsity property [48, 118] and the coding vectors from many other methods can be sparsified by thresholding [26] or simply setting top-k largest coding values to be nonzero [62]. The presence of a nonzero coding value essentially indicates the occurrence of a discriminative pattern identified by the supervised encoding method. This insight motivates us to define the novel generative model.

Our method applies to a wide range of supervised encoding methods. However, in this paper, we only consider a particular one of them. Specifically, we encode a local feature $\mathbf{x}$ by using the following encoder:

$$\mathbf{c} = \mathrm{f}(\mathbf{P}^T\mathbf{x} + \mathbf{b}), \tag{5.1}$$

where $\mathbf{c}$ is the coding vector and f is a nonlinear function. Here we use the soft-threshold function $f(a) = \max(0, a)$ as suggested in [19]. The final image representation is obtained by performing max-pooling over the coding vectors of all local features. Fig. 5.1 shows the general idea of our supervised coding method. To learn the encoder parameter, we feed the image representation into a logistic regression module to calculate the posterior probability and employ negative log-likelihood as the loss function. Then $\mathbf{P}$ and $\mathbf{b}$ are jointly learned with the parameters in the logistic regressor in

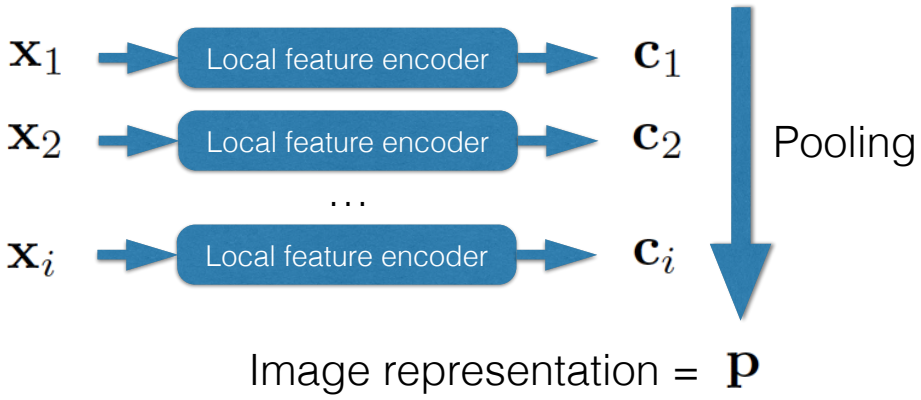


FIGURE 5.1: Illustration of the supervised coding method. It first applies a non-linear mapping to map each local feature $\mathbf{X}_i$ into a coding vector $\mathbf{C}_i$ and these coding vectors are pooled into an image representation $\mathbf{P}$. Classification is performed on top of these pooled image representations.

an end-to-end fashion through stochastic gradient descent. Note that this supervised encoder learning process is similar to performing fine-tuning on the last few layers of a convolutional neural network with $\mathbf{x}$ being the activations of a CNN [1].

5.3.2 Sparse coding based Fisher vector encoding

The recently proposed sparse coding based Fisher vector encoding (SCFV) is designed for encoding high-dimensional local features. Different from the traditional GMM based Fisher vector which essentially models local features by a finite set of templates (Gaussian distributions), it models local features as the linear combination of a set of templates (bases). Formally, the generative process of SCFV is described as follows:

- Draw a coding vector $\mathbf{u}$ from a zero mean Laplacian distribution $P_\lambda(\mathbf{u}) = \frac{1}{2\lambda}\exp(-\frac{\|\mathbf{u}\|_1}{\lambda})$.

- Draw a local feature $\mathbf{x}$ from the Gaussian distribution $\mathcal{N}(\mathbf{Bu}, \Sigma)$,

where $\mathbf{B}$ is the bases matrix and it can be learned by solving a dictionary learning problem in sparse coding and Σ is the covariance matrix which is usually fixed as a constant diagonal matrix.

[1] In fact, the performance of this approach is comparable with fine tuning a CNN network.

5.4 Proposed Method

5.4.1 Hybrid Generative Model

In this work, we combine the ideas of supervised encoding and Fisher vector encoding into a unified generative model. A local feature is assumed to be composed of a discriminative part and a residual part:

$$\mathbf{x} = \mathbf{x}_d + \mathbf{x}_r, \tag{5.2}$$

where $\mathbf{x}_d$ and $\mathbf{x}_r$ denote the discriminative part and the residual part respectively. The discriminative part indicates the visual pattern that is identified as informative for discrimination by an oracle method. The residual part in this decomposition can either correspond to the patterns shared by many classes, the irrelevant visual patterns or the remaining useful information which has not been successfully identified by the oracle method. The motivation for modeling these two components separately is that they are of differing value for the final application, and modeling them jointly thus undermines the discriminative power of the resulting Fisher vector.

The problem of how to achieve this decomposition remains, however. Clearly, there are infinitely possibilities to decompose $\mathbf{x}$ into $\mathbf{x}_d$ and $\mathbf{x}_r$. To solve this problem, we resort to the guidance of a pre-trained supervised coding method (we will discuss the specific choice in section 5.3.1). The idea of the supervised coding method is demonstrated in Fig. 5.1, the supervised coding method maps each local feature $\mathbf{x}$ to a coding vector $\mathbf{c}$ and pools coding vectors from all local features to obtain the image-level representation. It encompasses a wide range of feature coding methods, such as those discussed in section 2.3.5. In this paper we further assume that $\mathbf{c}$ is sparse. This is a reasonable assumption since many supervised encoding methods explicitly enforce the sparsity property [48, 118] and the coding vectors from many other methods can be sparsified by thresholding [26] or simply setting top-k largest coding values to be nonzero [62]. For those kinds of supervised coding methods, the presence of a nonzero coding value essentially indicates the occurrence of a discriminative elementary pattern identified by the supervised coding method. In other words, each active (non-zero) coding dimension corresponds to one discriminative elementary pattern and the discriminative part of the local feature is the combination of these patterns. Let $\mathbf{B}_d$ denote the collection of discriminative elementary patterns (bases) and $\mathbf{u}_d$ be their corresponding combination weight. The above insight motivates us to encourage $\mathbf{u}_d$ to share the similar nonzero dimensions with c , that is, to require $\|\mathbf{u}_d - \mathbf{c}\|_0$ to be small. However, the l_0 norm makes the Fisher vector derivation difficult. Thus we relax l_0 norm to l_2 norm in our approach.

To incorporate the above ideas into our two-stage feature generative process framework, we assume that $\mathbf{x}_d$ and $\mathbf{x}_r$ are drawn from Gaussian distributions whose mean vectors are the linear combination of two bases $\mathbf{B}_d$ and $\mathbf{B}_r$ respectively. For the combination weight of the residual part $\mathbf{u}_r$, we still assume that it is drawn from a Laplace distribution. The combination weight of the discriminative part $\mathbf{u}_d$, however is assumed drawn from a compound distribution which should encourage both sparsity and compatibility with the supervised coding $\mathbf{c}$. More specifically, we propose the following generative process of $\mathbf{x}$:

- Draw a coding vector $\mathbf{u}_d$ from the conditional distribution $P(\mathbf{u}_d|\mathbf{c})$.
- Draw a coding vector $\mathbf{u}_r$ from a zero mean Laplace distribution $P(\mathbf{u}_r) = \frac{1}{2\lambda}\exp(-\frac{\|\mathbf{u}_r\|_1}{\lambda_1})$.
- Draw a local feature $\mathbf{x}$ from the Gaussian distribution $\mathcal{N}(\mathbf{B}_d\mathbf{u}_d + \mathbf{B}_r\mathbf{u}_r, \Sigma)$, where $\mathbf{B}_d$ and $\mathbf{B}_r$ are model parameters. Here we define $\Sigma = diag(\sigma_1^2, \cdots, \sigma_m^2)$ and set $\sigma_1^2 = \cdots = \sigma_m^2 = \sigma^2$ as a constant.

In the above process, $P(\mathbf{u}_d|\mathbf{c})$ is defined as $\frac{1}{Z}\exp\left(-\frac{\|\mathbf{u}_d\|_1}{\lambda_2} - \frac{\|\mathbf{u}_d-\mathbf{c}\|_2}{\lambda_3}\right)$ to meet its two requirements as discussed above, where $Z = \int\limits_{\mathbf{u}_d} \exp\left(-\frac{\|\mathbf{u}_d\|_1}{\lambda_2} - \frac{\|\mathbf{u}_d-\mathbf{c}\|_2}{\lambda_3}\right) \mathrm{d}\mathbf{u}_d$ is a constant. Also note that we do not separately generate the discriminative and residual part of $\mathbf{x}$ in practise, i.e. $\mathbf{x}_d \sim \mathcal{N}(\mathbf{B}_d\mathbf{u}_d, \bar{\Sigma})$, $\mathbf{x}_r \sim \mathcal{N}(\mathbf{B}_r\mathbf{u}_r, \bar{\Sigma})$ and $\mathbf{x} = \mathbf{x}_d + \mathbf{x}_r$. This is because when both parts are generated from Gaussian distributions with the same covariance matrix, their summation is simply a Gaussian random variable with the mean vector being $\mathbf{B}_d\mathbf{u}_d + \mathbf{B}_r\mathbf{u}_r$ and covariance matrix being $\Sigma = 2\bar{\Sigma}$.

Similar to the approach I, we can derive the marginal probability of $\mathbf{x}$ from the above generative process as:

$$\begin{aligned} P(\mathbf{x}) &= \iint\limits_{\mathbf{u}_d,\mathbf{u}_r} P(\mathbf{x}, \mathbf{u}_d, \mathbf{u}_r|\mathbf{B}_d, \mathbf{B}_r, \mathbf{c})\mathrm{d}\mathbf{u}_d\mathrm{d}\mathbf{u}_r \\ &= \iint\limits_{\mathbf{u}_d,\mathbf{u}_r} P(\mathbf{x}|\mathbf{u}_d, \mathbf{u}_r, \mathbf{B}_d, \mathbf{B}_r, \mathbf{c})P(\mathbf{u}_r)P(\mathbf{u}_d|\mathbf{c})\mathrm{d}\mathbf{u}_d\mathrm{d}\mathbf{u}_r. \end{aligned} \tag{5.3}$$

This formulation involves an integral over latent variables $\mathbf{u}_d$ and $\mathbf{u}_r$, which makes the calculation difficult. Again, we follow the simplification to use the point-wise maximum within the integral term to approximate the likelihood:

$$\begin{aligned} P(\mathbf{x}) &\approx P(\mathbf{x}|\mathbf{u}_d^*, \mathbf{u}_r^*, \mathbf{B}_d, \mathbf{B}_r, \mathbf{c})P(\mathbf{u}_r^*)P(\mathbf{u}_\mathbf{d}^*|\mathbf{c}) \\ \mathbf{u}_d^*, \mathbf{u}_r^* &= \arg\max_{\mathbf{u}_d,\mathbf{u}_r} P(\mathbf{x}|\mathbf{u}_d, \mathbf{u}_r, \mathbf{B}_d, \mathbf{B}_r, \mathbf{c})P(\mathbf{u}_r)P(\mathbf{u}_\mathbf{d}|\mathbf{c}) \end{aligned} \tag{5.4}$$

The negative logarithm of the likelihood is then formulated as:

$$
\begin{aligned}
-\log P(\mathbf{x}|\mathbf{B}_d, \mathbf{B}_r, \mathbf{c}) = \min_{\mathbf{u}_d, \mathbf{u}_r} \|\mathbf{x} - \mathbf{B}_d\mathbf{u}_d - \mathbf{B}_r\mathbf{u}_r\|_2^2 + \\
\lambda_1\|\mathbf{u}_r\|_1 + \lambda_2\|\mathbf{u}_d\|_1 + \lambda_3\|\mathbf{u}_d - \mathbf{c}\|_2^2,
\end{aligned} \tag{5.5}
$$

where the model parameters $\mathbf{B}_d$ and $\mathbf{B}_r$ can be learned by minimizing the negative logarithm of the likelihood in Eq. (5.5).

5.4.2 Fisher vector derivation

We can derive the Fisher vector coding for our generative model:

$$
\begin{aligned}
\mathbf{G}^{\mathbf{x}}_{\mathbf{B_d}} &= \frac{\partial \log(P(\mathbf{x}|\mathbf{B}_d, \mathbf{B}_r, \mathbf{c}))}{\partial \mathbf{B}_d} \\
&= \frac{\partial \frac{1}{\sigma^2}\|\mathbf{x} - \mathbf{B}_d\mathbf{u}_d^* - \mathbf{B}_r\mathbf{u}_r^*\|_2^2 + \lambda_1\|\mathbf{u}_r^*\|_1 + \lambda_2\|\mathbf{u}_d^*\|_1 + \lambda_3\|\mathbf{u}_d^* - \mathbf{c}\|_2^2}{\partial \mathbf{B}_d} \\
&= \sum_{\mathbf{x}_i \in \mathbf{X}} (\mathbf{x}_i - \mathbf{B}_d\mathbf{u}_{di}^* - \mathbf{B}_r\mathbf{u}_{ri}^*)\mathbf{u}_{di}^{*\intercal}
\end{aligned} \tag{5.6}
$$

$$
\begin{aligned}
\mathbf{G}^{\mathbf{x}}_{\mathbf{B_r}} &= \frac{\partial \log(P(\mathbf{x}|\mathbf{B}_d, \mathbf{B}_r, \mathbf{c}))}{\partial \mathbf{B}_r} \\
&= \frac{\partial \frac{1}{\sigma^2}\|\mathbf{x} - \mathbf{B}_d\mathbf{u}_d^* - \mathbf{B}_r\mathbf{u}_r^*\|_2^2 + \lambda_1\|\mathbf{u}_r^*\|_1 + \lambda_2\|\mathbf{u}_d^*\|_1 + \lambda_3\|\mathbf{u}_d^* - \mathbf{c}\|_2^2}{\partial \mathbf{B}_r} \\
&= \sum_{\mathbf{x}_i \in \mathbf{X}} (\mathbf{x}_i - \mathbf{B}_d\mathbf{u}_{di}^* - \mathbf{B}_r\mathbf{u}_{ri}^*)\mathbf{u}_{ri}^{*\intercal}
\end{aligned} \tag{5.7}
$$

$$
\begin{aligned}
\mathbf{u}_d^*, \mathbf{u}_r^* = \arg\min_{\mathbf{u}_d, \mathbf{u}_r} \frac{1}{\sigma^2}\|\mathbf{x} - \mathbf{B}_d\mathbf{u}_d - \mathbf{B}_r\mathbf{u}_r\|_2^2 + \lambda_1\|\mathbf{u}_r\|_1 + \lambda_2\|\mathbf{u}_d\|_1 \\
+ \lambda_3\|\mathbf{u}_d - \mathbf{c}\|_2^2,
\end{aligned} \tag{5.8}
$$

where $\mathbf{u}_d, \mathbf{u}_r$ interact with $\mathbf{B}_d, \mathbf{B}_r$. According to the Lemma 2 in [11], we can calculate $\mathbf{G}^{\mathbf{x}}_{\mathbf{B_d}}$ and $\mathbf{G}^{\mathbf{x}}_{\mathbf{B_r}}$ as if $\mathbf{u}_d, \mathbf{u}_r$ did not depend on $\mathbf{B}_d, \mathbf{B}_r$. In other words, we can solve the inference problem in Eq. (5.8) to obtain $\mathbf{u}_d^*, \mathbf{u}_r^*$ first and then calculate $\mathbf{G}^{\mathbf{x}}_{\mathbf{B_d}}$ and $\mathbf{G}^{\mathbf{x}}_{\mathbf{B_r}}$. In the following sections, we call this Fisher vector encoding method Hybrid Sparse Coding based Fisher vector coding (HSCFVC in short) since the creation of its final image representation involves the components of both supervised coding and Fisher vector coding.

Note that HSCFVC essentially combines two ideas of building a good classification system: (1) identifying the discriminative pattern at the early coding stage of an image classification pipeline, i.e. supervised coding. (2) preserving as much information of local features as possible into the

high-dimensional image representation and relies on classifier learning to identify the discriminative pattern.

Algorithm 2 Matching Pursuit based algorithm for inferring $\mathbf{u}_d, \mathbf{u}_r$ in Eq. (5.9)

procedure MP (please see more details in the Appendix)
Input: $\mathbf{x}, \mathbf{B}_d, \mathbf{B}_r, k_1, k_2, \lambda, \mathbf{c}$
Output: $\mathbf{u}_d, \mathbf{u}_r$
Initialize residue $\mathbf{r} = \mathbf{x}, \mathbf{u}_d^1 = \mathbf{0}, \mathbf{u}_r^1 = \mathbf{0}$
Fixing $\mathbf{u}_r$, inferring $\mathbf{u}_d$
    for $t = 1 : k_1$ **do**
        Solve $\min_{\mathbf{e}_{d_j}, u_{d_j}} \|\mathbf{x} - \mathbf{B}_d\mathbf{u}_d^t - \mathbf{B}_r\mathbf{u}_r^t - \mathbf{B}_d\mathbf{e}_{d_j}u_{d_j}\|_2^2 + \lambda\|\mathbf{u}_d^t - \mathbf{c} + \mathbf{e}_{d_j}u_{d_j}\|_2^2$
        Update $\mathbf{r} \leftarrow \mathbf{r} - \mathbf{B}_d\mathbf{e}_{d_j}^* u_{d_j}^*$, $\mathbf{u}_d^{t+1} = \mathbf{u}_d^t + \mathbf{e}_{d_j}^* u_{d_j}^*$
    end for
Fixing $\mathbf{u}_d$, inferring $\mathbf{u}_r$
    for $t = 1 : k_2$ **do**
        Solve $\min_{\mathbf{e}_{r_j}, u_{r_j}} \|\mathbf{x} - \mathbf{B}_d\mathbf{u}_d^t - \mathbf{B}_r\mathbf{u}_r^t - \mathbf{B}_r\mathbf{e}_{r_j}u_{r_j}\|_2^2$
        Update $\mathbf{r} \leftarrow \mathbf{r} - \mathbf{B}_r\mathbf{e}_{r_j}^* u_{r_j}^*$, $\mathbf{u}_r^{t+1} = \mathbf{u}_r^t + \mathbf{e}_{r_j}^* u_{r_j}^*$
    end for
end procedure

5.4.3 Inference and learning

To learn the model parameters and calculate the Fisher vector, we need to solve the optimization problems in Eq. (5.5). This problem can be solved using existing sparse coding solvers. However, it can be still slow for high-dimensional local feature in practise. In [71], it has been suggested that a matching pursuit algorithm can be adopted as a substitute for an exact sparse coding problem for local feature encoding approach. Thus, in this work we use the method in [71] to approximately solve Eq. (5.5) which essentially solves the following variant problem of Eq. (5.5):

$$\begin{aligned} &\min_{\mathbf{u}_d, \mathbf{u}_r} \|\mathbf{x} - \mathbf{B}_d\mathbf{u}_d - \mathbf{B}_r\mathbf{u}_r\|_2^2 + \lambda\|\mathbf{u}_d - \mathbf{c}\|_2^2 \\ &s.t.\ \|\mathbf{u}_d\|_0 \leq k_1, \quad \|\mathbf{u}_r\|_0 \leq k_2. \end{aligned} \tag{5.9}$$

In the matching pursuit algorithm, the Eq. (5.9) is sequentially solved by updating one dimension of $\mathbf{u}_d$ and $\mathbf{u}_r$ at each iteration while keeping the values at other dimensions fixed. In our solution, we

first update each dimension of $\mathbf{u}_d$ and then update $\mathbf{u}_r$. The algorithm is described in Algorithm 2. For the derivation and more details of Algorithm 2, please refer to the Appendix section.

To learn the model parameters $\mathbf{B}_d$ and $\mathbf{B}_r$ in HSCFVC, we employ an alternating algorithm which iterates between the following two steps: (1) fixing $\mathbf{B}_d$ and $\mathbf{B}_r$ in HSCFVC, then solving $\mathbf{u}_d$ and $\mathbf{u}_r$; (2) fixing $\mathbf{u}_d$ and $\mathbf{u}_r$, then updating $\mathbf{B}_d$ and $\mathbf{B}_r$ through the solver proposed in [61].

5.4.4 Implementation details

Local features

Using the neuron activations of a pre-trained CNN model as local features has become popular recently [39, 16, 66, 25]. The local feature can be either extracted from the fully-connected layer or the convolutional layer. For the former case, a number of image regions are firstly sampled and each of them will pass through the deep CNN [2] to extract the fully-connected layer activations which will be used as a local feature. For the latter case, the whole image is directly fed into a pre-trained CNN and the activations at each spatial location of a convolutional layer are extracted as a local feature [65]. It has been observed that the fully-connected layer feature is useful for generic object classification and the convolutional layer feature is useful for texture and fine-grained image classification (the discriminative patterns are usually special types of textures). In this work, we use both kinds of local features in our experiment.

Pooling and normalization

From the i.i.d assumption, the Fisher vector of the whole image equals to

$$\frac{\partial \log(P(\mathbf{X}|\mathbf{B}))}{\partial \mathbf{B}} = \sum_i \frac{\partial \log(P(\mathbf{x}_i|\mathbf{B}))}{\partial \mathbf{B}} \tag{5.10}$$

This is equivalent to performing the sum-pooling for the extracted Fisher coding vectors. However, it has been observed [85, 2] that the image signature obtained by using sum-pooling tends to over-emphasize the information from the background [85] or bursting visual words [2]. It is important to apply some normalization operations when sum-pooling is used. In this paper, we apply the intra-normalization [2] to normalize the pooled Fisher vectors. Besides intra-normalization, we also utilize the power normalization as suggested in [85].

[2] a faster and equivalent implementation is to convert the fully-connected layer to the convolutional layer to perform the local feature extraction process [119].

5.5 Experiment

To evaluate the effectiveness of the two proposed compositional FVC approaches, we conduct experiments on three large datasets: Caltech-UCSD Birds-200-2011 (Birds-200 in short), MIT indoor scene-67 (MIT-67 in short) and Pascal VOC 2007 (Pascal-07 in short). These three datasets are commonly used evaluation benchmarks for fine-grained image classification, scene classification and object recognition. The focus of our experiments is to verify two aspects: (1) whether the proposed SCFVC outperforms the traditional GMM based FVC (GMM-FVC in short); (2) whether the proposed HSCFVC outperforms SCFVC and its guiding supervised coding method in section 5.3.1 (denoted as SupC in the following part) since HSCFVC is expected to enjoy the merits of both SupC and SCFVC.

TABLE 5.1: Comparison of results on Birds-200. The lower part of this table lists some results in the literature.

Methods	Accuracy	Comments
HSCFVC (proposed)	**80.8%**	
SCFVC	77.3%	
GMMFVC	70.1%	
SupC	69.5%	
CNN-Jitter	63.6%	
CrossLayer[65]	73.5%	
GlobalCNN-FT [3]	66.4 %	no parts, fine tunning
Parts-RCNN-FT [123]	76.37 %	use parts, fine tunning
Parts-RCNN [123]	68.7 %	use parts, no fine tunning
CNNaug-SVM [89]	61.8%	-
CNN-SVM [89]	53.3%	CNN global
DPD+CNN [30]	65.0%	use parts
DPD [124]	51.0%	-
Bilinear CNN [64]	**85.1%**	Two networks, fine-tuning
Two-Level Attention [115]	77.9%	
Unsupervised Part Model [94]	81.0%	

5.5.1 Experimental Setting

As mentioned above, we use the activations of a pre-trained CNN as the local features and activations from both the convolutional layer and the fully-connected layer are used. More specifically, we extract the fully-connected layer activations as the local feature for PASCAL-07 and MIT-67 because we empirically found that the fully connected layer activations work better for scene and generic object classification. For Birds-200, we use the convolutional activations as local features since it has been reported that convolutional layer activations lead to superior performance than the fully-connected layer activations when apply to the fine-grained image classification problem [65]. Throughout our experiments, we use the vgg-very-deep-19-layers CNN model [98] as the pre-trained CNN model. To extract the local features with the fully-connected layer activations, we first resize the input image into 512×512 pixels and 614×614 pixels. Then we extract regions of size 224x224 pixels at a dense spatial grid with the step size of 32 pixels. These local regions are fed into the deep CNN and the 4096-dimensional activations of the first fully-connected layer are extracted as local features. To extract the local features from the convolutional layer, we resize input images to 224×224 pixels and 448×448 pixels and then extract the convolutional feature activations from the "conv5-4" layer as local features (in such setting, there are $14 \times 14 + 28 \times 28$ local features per image). To decouple the correlations between dimensions of CNN features and avoid the dimensionality explosion of the Fisher vector representation, for fully-connected layer features we apply PCA to reduce its dimensionality to 2000. For convolutional layer features, we do not perform dimensionality reduction but only use PCA for decorrelation.

Five comparing methods are implemented. Besides the proposed SCFVC, HSCFVC and the traditional GMM based FVC, the supervised coding method which serves as the guiding coding method for HSCFVC is also compared to verify if additional performance improvement can be achieved via our HSCFVC. Also, we compare a baseline in [89, 3], denoted as CNN-Jitter, which averages the fully-connected layer activations from several transformed versions of an input image, i.e. cropping the four corners and middle region of an input image. We also quote results of other methods reported from the literature for reference. However, since they may adopt different implementation details, their performance may not be directly comparable to ours.

Both the proposed methods and baseline methods involve several hyper-parameters, their settings are described as follows. In SCFVC, the codebook size of $\mathbf{B}$ is set to be 200. In HSCFVC, the dimensionality of $\mathbf{c}$ and the codebook size of $\mathbf{B}_d, \mathbf{B}_c$ are set to be 100. Therefore, the dimensionality of the image representation created by SCFVC and HSCFVC are identical. For GMM-FVC, we also

set the number of Gaussian distributions to be 200 to make fair comparison. We employ the matching pursuit approximation to solve the inference problem in the SCFVC and HSCFVC. The sparsity of coding vector is controlled by the parameter k in Eq. (5.9). Both k_1 in HSCFVC and k in SCFVC have significant influences on performance. We select k_1 from $\{10, 20, 30\}$ and k from $\{10, 20, 30, 40\}$ via cross-validation. k_2 is fixed to 10 for simplicity. λ in Eq. (5.9) is fixed to be 0.5 unless otherwise stated. Throughout our experiments, we use the linear SVM [10] as the classifier.

5.5.2 Main results

Birds-200 Birds-200 is a commonly used benchmark for fine-grained image classification which contains 11788 images of 200 different bird species. The experimental results on this dataset are shown in Table 5.1. As can be seen, both the proposed SCFVC and HSCFVC outperform the traditional GMM-FVC by a large margin. The improvement can be as large as 10%. This observation clearly demonstrates the advantage of using compositional mechanism for modeling local features. Also, HSCFVC achieves better performance than SCFVC, which outperforms the latter by more than 3%. Recall that the difference between HSCFVC and SCFVC lies in that the former further decomposes a local feature into a discriminative part and a residual part, thus the superior performance of HSCFVC clearly verifies the benefit of adopting such modeling. To achieve this decomposition, HSCFVC uses a supervised coding method as guidance. Thus it is interesting to examine the performance relationship between HSCFVC and its guiding coding method. This comparison is also shown in Table 5.1. As can be seen, HSCFVC also outperforms its guiding supervised encoding by 11%. As discussed previously, this further performance boost is expected because the supervised coding method may not be able to extract all discriminative patterns from local features and the missing information can be re-gained from the high-dimensional image signature generated by HSCFVC. Also, it can be seen that the CNN-Jitter baseline performs worst in comparison with all other methods. This suggests that to build image-level representation with a pre-trained CNN model it is better to adopt the CNN to extract local features rather than global features as in the CNN-Jitter baseline. Finally, by cross-referencing the recently published performance on this dataset, we can conclude that the proposed method is on par with the state-of-the-art. Note that some methods achieve better performance by adopting strategies which have not been considered here but can be readily incorporated into our method. For example, in [64], the CNN model is fine-tuned. We can use the same technique to improve our performance.

MIT-67 MIT-67 contains 6700 images with 67 indoor scene categories. This dataset is very challenging because the differences between some categories are very subtle. The comparison of classification results is shown in Table 5.2. Again, we observe that the proposed HGMFVC and SCFVC significantly outperform traditional GMMFVC. The improvement from HSCFVC and SCFVC to GMM-FVC are around 7% and 5% respectively. In addition, the HSCFVC achieves superior performance than SCFVC and SupC. This again shows that HSCFVC is able to combine the benefit of both Fisher vector coding and supervised coding. By comparing our best performance against the results reported in the literature, we can see that our methods are comparable to the state-of-the-art results. The work in [16] also employs the traditional GMM-FVC but achieves higher classification performance than ours. By examining their experimental setting, we found that their method actually extracts convolutional activations of a pre-trained CNN at 7 different scales which uses far more local features than ours.

TABLE 5.2: Comparison of results on MIT-67. The lower part of this table lists some results in the literature.

Methods	Accuracy	Comments
HSCFVC (proposed)	**79.5%**	
SCFVC	77.6%	
GMMFVC	72.6%	
SupC	76.4%	
CNN-Jitter	70.2%	
MOP-CNN[39]	68.9%	with three scales
VLAD level2[39]	65.5%	with single best scale
CNN-SVM[89]	58.4%	use CNN on whole image
Mid-level Mining[62]	69.7%	use three scales
SemanticFV[25]	68.5%	best performance for single scale
SemanticFV[25]	72.9%	use four scales
CrossLayer[65]	71.5%	combine two resolutions and use global CNN features
DeepTexture [16]	**81.7%**	7 scales with convolutional layer activations
Bilinear CNN [64]	77.6%	without fine-tuning
Bilinear CNN [64]	71.1%	fine-tuning

Pascal-07 The Pascal VOC 2007 dataset is composed of 9963 images of 20 object categories. The task is to predict whether a target object is present in an image or not. Table 5.5 shows the results measured by mean average precision (mAP) of all 20 classes. Table 5.3 provides performance comparison on each one of the 20 categories. As seen, the HSCFVC and SCFVC again outperform the traditional GMM-FVC. Also, HSCFVC achieves the best classification performance. By cross-referencing Table 5.3, it is observed that the relative performance of HSCFVC, SCFVC and GMMFV is almost kept for all 20 classes, that is, HSCFVC always achieves the best performance and SCFVC is superior over GMMFVC. By taking a close examination on Table 5.3, we observe that the proposed method usually achieves the largest improvement on the difficult categories (those categories with less than 90% mAP), e.g. the categories "TV", "Sheep", "Bottle", "Chair". This can be understood by the fact that for difficult classes, many subtle class differences can only be captured by very discriminative image representations.

Note that the method in [98] directly applies average pooling on the fully-connected layer activations extracted from local image regions and it achieves 89.3% mAP. However, they choose a different scheme to crop image regions. Different from our setting, they maintain the aspect ratio of input images and sample across multiple scales (5 scales in total). We also experiment with a similar setting, that is, to retain the aspect ratio of input images and sample in 3 scales, and we can achieve comparable or even better performance (HGMFV achieves 89.8% mAP). Using this setting, we also compare GMMFVC, SCFVC, HSCFVC and average pooling (the method in [98] on PASCAL 2012. We train the model on the training set and evaluate the result on the validation set. The results are shown in Table 5.4. As can be seen, the performance relationship of different methods is consistent with that in PASCAL 2007.

5.5.3 Analysis of HSCFVC

The classification accuracy vs. the value of λ

In HSCFVC, the optimal coding vector is calculated by solving Eq. (5.9). The optimization in Eq. (5.9) involves a trade-off parameter λ which controls the fidelity of $\mathbf{u}_d$ to the supervised coding vector $\mathbf{c}$. Larger λ enforces the active elements of $\mathbf{u}_d$ to be consistent with those of $\mathbf{c}$. While on the contrary, smaller λ loses this consistency.

In this subsection, we evaluate its impact on the classification performance. We conduct our experiment on all three datasets and the results are shown in Fig. 5.2. As can be seen, for all datasets,

TABLE 5.3: Comparison of results on Pascal VOC 2007 for each of 20 classes.

	TV	train	sofa	sheep	plant	person	mbike	horse	dog	table
Global Jitter	81.9	96.3	72.9	86.1	61.6	95.2	89.3	91.5	90.9	79.7
GMMFVC	81.3	95.8	77.3	80.6	63.2	95.9	89.9	92.1	89.1	79.1
SCFVC	84.1	96.4	79.7	84.2	64.2	96.2	90.4	93.8	90.9	83.1
HSCFVC	**87.4**	96.7	80.0	**88.6**	**65.9**	**97.1**	**92.5**	94.6	**93.9**	84.2
	cow	chair	cat	car	bus	bottle	boat	bird	bike	areo
Global Jitter	77.8	67.4	92.1	91.2	85.2	56.6	92.8	92.9	90.4	97.1
GMMFVC	81.1	66.6	93.3	92.3	86.4	58.9	89.2	93.5	92.2	94.9
SCFVC	81.7	68.2	92.6	91.9	88.4	61.8	90.6	93.6	92.6	97.3
HSCFVC	**83.4**	**72.2**	**95.2**	**93.9**	**90.3**	**65.0**	**92.5**	**95.3**	**94.0**	97.6

TABLE 5.4: Comparison of results on Pascal VOC 2012 for each of 20 classes.

	TV	train	sofa	sheep	plant	person	mbike	horse	dog	table	-
Average Pooling	83.4	96.8	64.8	89.6	54.0	95.7	91.2	91.4	96.1	73.8	
GMMFVC	87.2	97.5	71.5	89.1	56.0	95.8	91.3	90.8	94.7	76.2	
SCFVC	87.5	97.2	76.0	91.8	62.4	96.7	93.0	92.7	96.3	79.5	
HSCFVC	87.9	97.7	75.8	92.2	62.6	96.7	93.4	**93.6**	96.6	**81.1**	
	cow	chair	cat	car	bus	bottle	boat	bird	bike	areo	mAP
Average Pooling	83.2	70.5	96.9	78.5	92.9	64.8	89.1	94.4	86.2	98.5	84.6
GMMFVC	84.3	75.4	95.9	82.2	93.2	64.5	88.6	94.4	88.5	97.6	85.8
SCFVC	86.3	77.7	97.3	83.7	94.4	70.2	90.3	95.9	90.4	98.5	87.9
HSCFVC	**88.6**	**78.8**	97.4	84.3	94.8	**71.2**	**90.5**	96.0	**91.0**	**99.8**	**88.5**

poor performance will be obtained if λ is set to 0. This is not surprising because in this case the guidance signal of the supervised coding method is completely disabled. As expected, when λ increases, the classification accuracy rises accordingly. The performance becomes steady when λ is reasonably large. This suggests the necessity of introducing the fidelity term $\|\mathbf{u_d} - \mathbf{c}\|_2^2$ in Eq. (5.9). Also, as can be seen, as long as λ is sufficient large, the classification performance does not vary too much with the choice of λ and that is why we simply set λ to 0.5 throughout our experiments.

TABLE 5.5: Comparison of results on Pascal VOC 2007. The lower part of this table lists some results in the literature.

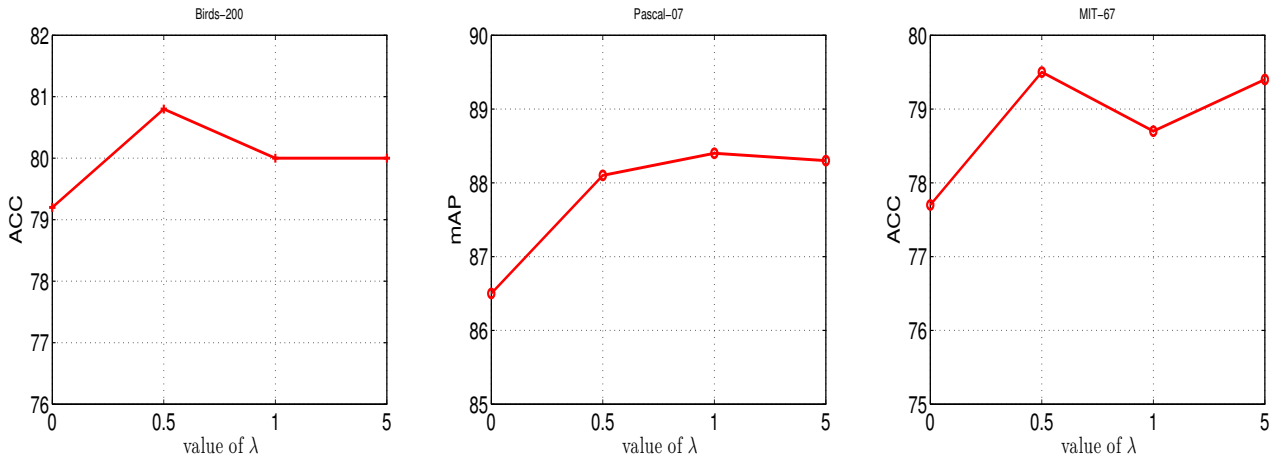
Methods	MAP	Comments
HSCFVC (proposed)	**88.1% (89.8%)**	
SCFVC	86.6%	
GMMFVC	84.6%	
SupC	84.2%	
CNNaug-SVM[89]	77.2%	with augmented data, use CNN on whole image
CNN-SVM[89]	73.9%	no augmented data, use CNN on whole image
Deep Fisher[102]	56.3%	training GMM parameters in an end-to-end fashion
Mid-level Mining[62]	75.2%	use three scales
CrossLayer[65]	77.8%	combine two resolutions and global CNN features
DeepTexture [16]	84.9%	
SPP-Net [41]	82.9%	-
CNN S TUNE-RNK [13]	82.4%	-
CNN aggregation [98]	**89.3%**	using a different cropping scheme

TABLE 5.6: The impact of the number of bases on our methods.

Methods	the number of bases	Accuracy
SCFVC	200	77.6%
SCFVC	400	77.3%
SCFVC	600	77.2%
HSCFVC	200	79.5%
HSCFVC	400	79.4%
HSCFVC	600	78.9%

The impact of the residual part Fisher vector $\mathbf{G}^{\mathbf{X}}_{\mathbf{B_c}}$ on classification performance

Recall that the Fisher vector in HSCFVC can be decomposed into two parts: the discriminative part $\mathbf{G}^{\mathbf{X}}_{\mathbf{B_d}}$ and the residual part $\mathbf{G}^{\mathbf{X}}_{\mathbf{B_c}}$. By default we use both parts for classification because we postulate that these two parts can compensate each other. In this subsection, we verify this point by comparing



(a) (b) (c)

FIGURE 5.2: The impact of the parameter λ in Eq. (5.9) on the classification performance. (a) Results on Birds-200; (b) results on Pascal-07; (c) results on MIT-67.

the performance of merely using $\mathbf{G}^{\mathbf{X}}_{\mathbf{B_d}}$, the discriminative part Fisher vector and using both parts. Note that we do not compare the scheme of only using the residual part here because only using the residual part is equivalent to SCFVC which has already been compared in the previous sections. The comparison results are shown in Fig. 5.3. As can be seen, for all three datasets, combining $\mathbf{G}^{\mathbf{X}}_{\mathbf{B_c}}$ and $\mathbf{G}^{\mathbf{X}}_{\mathbf{B_d}}$ can lead to better performance, which supports our assumption.

5.6 Summary

In this work, we propose a novel Fisher vector coding method for image classification. Different from exiting Fisher vector coding schemes that try to model the generation process of local features directly, we decompose the local feature into a discriminative part and a residual part and model them separately. The decomposition can be achieved via the guidance of any supervised coding methods. By this way, we can essentially benefit from the merits of both unsupervised Fisher vector coding and supervised coding.

By evaluating our method in various standard image classification datasets, our method does not only achieve superior performance comparing to the baseline coding methods but observes comparable or even better performance than state-of-the-art methods. Although our coding method is applied to image classification in the thesis, it can be generalized to other vision tasks where the visual representation plays important role.

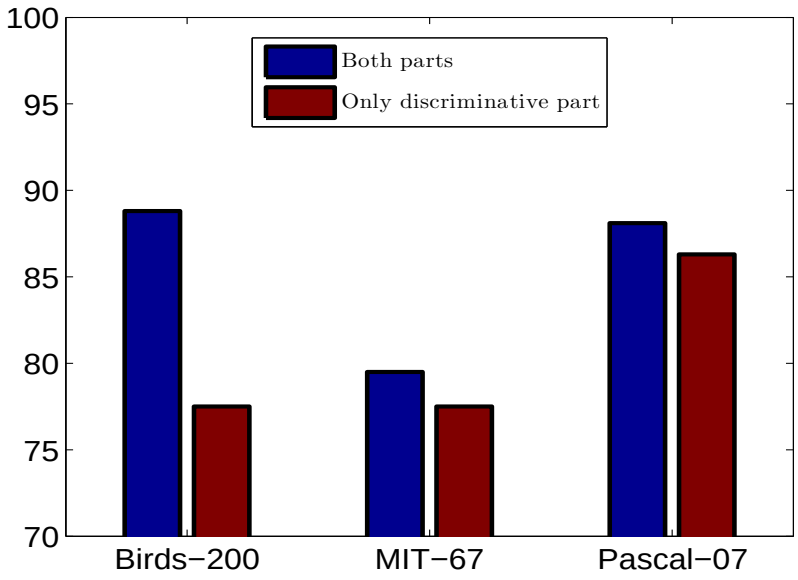


FIGURE 5.3: Comparison of with and without the common part Fisher vector.

Chapter 6

Identifying Unusual Objects From Images by Modelling the Detection Score Distribution

6.1 Overview

In many local visual pattern modelling methods, including the ones presented above, the local patterns are assumed to be independent and identically distributed (i.i.d.). But in fact, these local features tend to interact with each other to describe an image or a video.

In this chapter, we study the challenging problem of identifying unusual instances of known objects within an "open world" setting via modelling the local visual pattern distribution. That is, we aim to find objects that are members of a known class, but which are not typical of that class. Thus the unusual objects should be distinguished from not only "regular objects" but "other objects" as well.

Considering the promising performance of Region CNN [37], we represent an image by a set of region-level CNN features and learn an object detector to map these local features into scalar detection scores in order to get rid of the distraction of noisy content. We propose a novel approach to identify unusual objects by inspecting the distribution of the detection scores at multiple image regions. The key observation motivating our approach is that for "regular object" images as well as "other objects" images, the region-level scores follow their own essential patterns in terms of both the score values and the spatial distributions while the detection scores obtained from an "unusual object" image tend to break these patterns. To model this distribution, we propose to use Gaussian Processes (GP) to construct two separate generative models for the case of the "regular object" and the "other objects". More specifically, we design a new covariance function to simultaneously model the detection score at a single region and the score dependencies at multiple regions. We finally demonstrate the superior

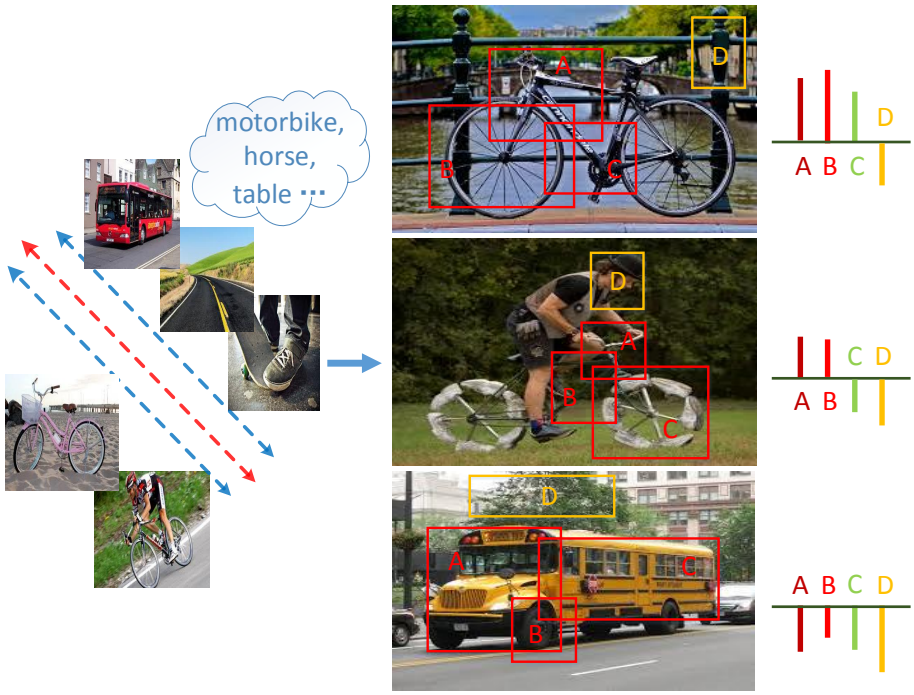


FIGURE 6.1: Illustration of our idea for detecting the unusual "bicycle". The region-level detection scores for regular "bicycle" and "other objects" have their own patterns in terms of score value and spatial distribution. The detection scores of "unusual bicycle" will break these patterns.

performance of our method on a large dataset newly constructed for this task.

6.2 Introduction

Humans have the ability to detect the unusual status of objects without seeing the unusual patterns beforehand. Mimicking this ability with computer vision technique can be practically useful for the applications such as surveillance or quality control. Existing studies towards this goal are usually conducted on small datasets and controlled scenarios i.e., with relatively simple background [7] or specific type of unusuality [15, 80]. To address this issue, in this work we present a large dataset which captures more general unusualities and has more complex background. Moreover, we adopt a more realistic "open world" evaluation protocol. That is, we need to distinguish the "unusual version of object-of-interest" not only from the "regular object" belonging to the same category but also from the "other objects" (objects from other categories) at the test stage.

The reason why people can recognize an image to be an unusual example of a certain object is because it shares some common patterns of this object but deviates from the regular examples of the

object. In other words, the "unusual object" images are supposed to be more similar to the "regular object" images comparing to the images of other objects. If we apply a detector learned using "regular object" images as positive training data and images of other objects as negative data to the regions of an image, the score values of the "unusual object" images are expected to be larger than the scores of the "other object" images and smaller than those of the "regular object" images. Apart from the values of the region-level detection scores, the spatial distributions of the detection scores may encode some discriminative information as well. As illustrated in Fig. 6.1, positive detection scores should be densely overlapped in regular images while in unusual images the score distribution may break this pattern due to the existence of the unusual parts. To model these two factors, we propose to use Gaussian Processes (GP) [88] to construct two separate generative models for the detection scores of "regular object" image regions and "other objects" image regions. The mean function is defined to depict the prior information of the score values of either "regular object" images or "other object" images and a new covariance function is designed to simultaneously model the detection score at a single region non-parametrically and capture the inter-dependency of scores at multiple regions. Note that unlike the conventional use of GP in computer vision, our model does not assume that the region scores of an image are i.i.d. This treatment allows our method to capture the spatial dependency of detection scores, which turns out to be crucial for identifying unusual objects. By comparing with several alternative solutions on the proposed dataset, we experimentally demonstrate the effectiveness of the proposed method. To summarize, the main contributions of this paper are:

- We propose a large dataset and present a more realistic "open world" evaluation protocol for the task of unusual object identification from images.
- We propose a novel approach for unusuality detection by looking into the detection score values as well as the spatial distributions of the detection scores of the image regions. We propose to use Gaussian Processes (GP) to simultaneously model the detection score at a single region and the score dependencies at multiple regions.

6.3 A New Dataset

6.3.1 Dataset Introduction

In this paper, we propose a new dataset for the task of unusual image detection. The data is collected from *Google Images* and *Bing Images* which is composed of 20,420 images belonging to 20 classes.

FIGURE 6.2: Examples of unusual images. Left column: aeroplane, apple, bus. Right column: horse, dining table, road.

We choose the 20 classes referring to the PASCAL VOC dataset [31] but replace some classes that are not suitable for the task. The images of each class are composed of both regular images and unusual images. For regular images, we try different feasible queries to collect sufficient data. Taking "apple" for example, we try "fuji apple", "pink lady", "golden delicious", etc. To collect unusual images, we use keywords like "unusual", "unusual", "abnormal", "weird", "broken", "decayed", "rare", etc. After the images are returned, we manually remove the unrelated and low-quality data. Also, we perform near-duplicate detection to remove some duplicate images. In general, the number of unusual images per class is comparable to the sum of regular and "other class" images. Fig. 6.2 shows some examples of unusual images.

TABLE 6.1: Comparison of the proposed dataset to existing datasets. [15] addresses the unusual type of "out of context". [80] deals with violations of "co-occurrence", "positional relationship" and "expected scale".

dataset	♯ images	unusual category	accurate detector
[80]	150	specific	yes
[15]	218	specific	yes
ours	20,420	general	no

There exist some other datasets [15, 80] for unusual image detection. A comparison between our dataset and the existing datasets is summarized in Table 6.1. The main difference is twofold.

- Our dataset is large-scale comparing to the existing datasets, increasing the number of images from several hundred to more than twenty thousand.
- While the existing datasets are proposed for specific unusual category such as "out-of-context", "relative position violation" and "relative scale violation", our dataset is for general unusual cases.

Besides the above differences, we adopt a more practical evaluation protocol compared with [15, 80]. That is, we evaluate the unusual object detection with the presence of irrelevant objects. This is different from [7] where unusuality detection is performed in controlled environment with relatively simple background.

6.3.2 Problem Definition

For a given object category $\mathcal{C}$, we divide it into two disjoint subcategories, a regular sub-class $\mathcal{C}^r$ and an unusual sub-class $\mathcal{C}^u$, with $\mathcal{C} = \mathcal{C}^r \cup \mathcal{C}^u$ and $\mathcal{C}^r \cap \mathcal{C}^u = \emptyset$. We call an image I a regular image if $I \in \mathcal{C}^r$ and an unusual image if $I \in \mathcal{C}^u$. If an image I does not contain the given object, we label it as belonging to the "other class" set $\mathcal{C}^o$. The task is to determine if a test image $I \in \mathcal{C}^u$. Note that for $\mathcal{C}$, only the regular and "other class" images are available for training.

6.4 Key Motivation

Regular object images of the same class are alike; each unusual object image, however, is unusual in its own way. Thus, it is somehow impossible to collect a dataset to cover the space of the unusual images and one common idea to handle this difficulty is to build a "regular object" model to identify the "unusual objects" as outliers. While most traditional methods [126, 7] build this model based on the visual features extracted from images, our approach takes an alternative methodology by firstly training a detector from the "regular object" images and "other objects" images and then discovering the unusuality based on the detection score patterns. The merit of using detection scores for unusuality detection are as follows. (1) It is more computationally efficient since the appearance information has been compressed to a single scalar of detection values. This enables us to explore complex interaction of multiple regions within an image while maintaining reasonable computational cost. (2) It naturally handles the background and "other class" distraction since our detector is trained by using the "regular object" and "other objects". More specifically, our method is inspired by two intuitive postulates of how humans recognize an "unusual object", which are elaborated as follows.

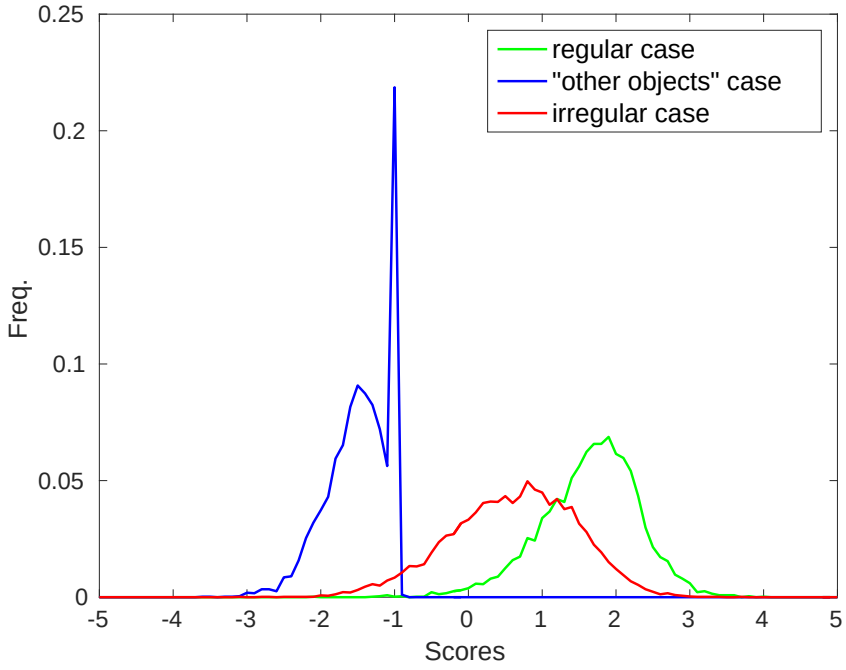


FIGURE 6.3: Histograms of decision scores for regular images, unusual images and "other class" images in the testing data. The decision scores are obtained by applying the classifiers learned from global images.

Postulate I: discrimination in detection score values. From the perspective of human vision, an unusual object is something "looks like an object-of-interest, but is still different from its common appearance". If we view the object detection score as a measure of the likelihood of an image containing the object, then the above postulate could correspond to a relationship in detection scores $f(I^o) < f(I^u) < f(I^r)$, where $f(I^o)$, $f(I^u)$ and $f(I^r)$ denote the detection score of the "other object", "unusual object" and "regular object" respectively. To verify this relationship, we train an image-level object classifier and plot the accumulated histograms of the scores of regular, unusual and other-class images of each class in Fig. 6.3. It can be seen from this figure that the distribution of the score values is generally consistent with our assumption. However, there are still overlaps especially between regular and unusual images, which means that using this criterion alone cannot perfectly distinguish the unusual images.

Postulate II: discrimination in the spatial dependency of detection scores. When exposed to part of the regular object, human can predict what the neighbouring parts of the object should look like without any difficulty. But unusual object may break this smoothness. This suggests that if we apply an object detector to the object proposals of an image, the region-level detection scores of the three different types of images may exhibit different dependency patterns. Fig. 6.4 shows the top 20

regions of some example images of *car* class according to the values of the detection scores. As seen, for regular car the positive bounding boxes are densely overlapped and images from other classes such as *motorbike* are supposed to have no positively scored proposals. Detection scores of unusual images may disobey both of these two distribution patterns. For example, two strongly overlapped regions may have opposite detection scores.

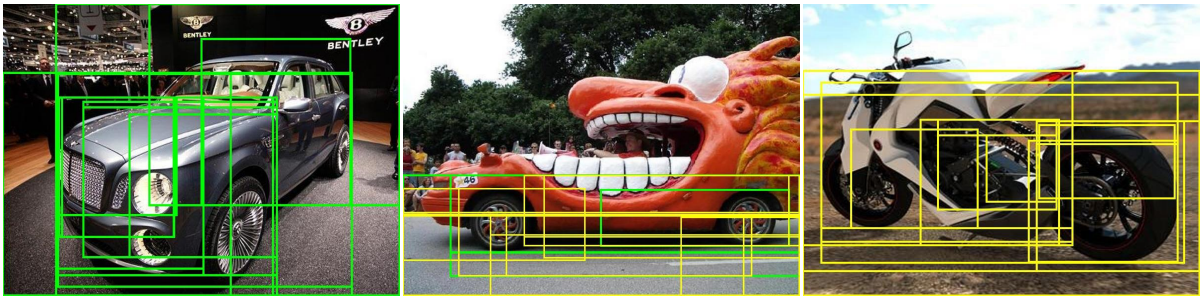


FIGURE 6.4: Visualization of spatial distribution of detection scores for test images of *car* class. Top-20 scored bounding boxes of an image are visualized. Positive proposals are visualized in green box and negative are visualized in yellow. From left to right: regular car, unusual car and other object (motorbike).

6.5 Proposed Approach

Motivated by the above analysis, we propose a two-step approach to the task of unusual image detection. We first apply a Multi-Instance Learning (MIL) approach to learn a region-level object detector and then design Gaussian Processes (GP) based generative models to model the detection score distributions of the "regular object" and the "other objects". Once the model parameters are learned, we can readily determine whether a test image is unusual by evaluating its fitting possibilities to these two generative models.

6.5.1 Object Detector Learning

Taking the region proposals of images as instances, we represent each image as a bag of instances. Since we only have the image-level label indicating the presence or absence of the object, the learning of region-level detector is essentially a weakly supervised object localization problem. Considering both the localization accuracy and the scalability, we follow the MIL method in [79] to learn an object detector for each class. For a class C, we have a set of regular images containing the object as

positive training data and a set of images belonging to other classes where the object concerned does not appear as negative training data.

We use Selective Search [103] to extract a set of object proposals for each image and from the perspective of MIL, each proposal is regarded as an instance. Then each image I^i is represented by a $N_i \times D$ matrix $\mathbf{X}^i$ where N_i denotes the number of proposals and D represents the dimensionality of the proposal representations. Inspired by [79], we optimize the following objective function to learn the detector,

$$\mathcal{J} = \sum_i \log(1 + e^{-y^i \max_j \{\mathbf{w^T x}_j^i + b\}}), \tag{6.1}$$

where $\mathbf{w} \in \mathbb{R}^{D\times 1}$ serves as an object detector, $\mathbf{x}_j^i$ indicates the jth instance of the ith image and $\mathbf{w}^T\mathbf{x}_j^i + b$ is its detection score. The single image-level score is aggregated via the max-pooling operator $\max\{\cdot\}$ and it should be consistent with the image-level class label $y^i \in \{1, -1\}$. The parameters $\mathbf{w}$ and b can be learned via back-propagation using stochastic gradient descent (SGD).

6.5.2 Gaussian Processes based Generative Models

In this section, we elaborate how to use GP to model the distribution of the region-level detection scores. Unlike traditional GP based regression [105] which takes a single feature vector as input, we treat multiple proposals within an image as the input and our model will return a probability to indicate the fitting likelihood of the proposal set.

GP assumes that any finite number of random variables drawn from the GP follow a joint Gaussian distribution and this distribution is fully characterized by a mean function $m(x)$ and a covariance function $k(x, x')$ [88]. In our case, we treat the detection score of each proposal as a random variable. The mean function depicts the prior information of the score values, e.g. the value tends to be a positive scalar for the "regular object" images. The covariance function plays two roles. (1) As in standard GP regression, it serves as a non-parametric estimator of the score value. More specifically, if a proposal is similar (in terms of a defined proposal representation) to a proposal in the training set, it encourages them to share similar scores. (2) As one of our contributions, we also add a term in the covariance function to encourage the overlapped object proposals within the same test image to share similar detection scores. In the following subsections, we introduce the details of the design of the mean function and covariance function.

GP Construction

For each class $\mathcal{C}$, we will construct two GP based generative models for regular images and "other objects" images separately. Without losing generality, we will focus on regular images in the following part.

Suppose we have $N^{\mathcal{C}}$ positive training images for class $\mathcal{C}$. For each image I^i ($i \in \{1, 2, \cdots, N^{\mathcal{C}}\}$), we use the top-$n$ scored proposals s^i_j ($j \in \{1, 2, \cdots, n\}$) only in order to reduce the distraction impact of the background. Their associated detection scores can be obtained via the function $f(s^i_j)$. In our model we assume that f is distributed as a GP with a mean function $m(\cdot)$ and a covariance function $k(\cdot, \cdot)$

$$f \sim \mathcal{GP}(m, k). \tag{6.2}$$

Mean function: We define the mean function $m(s) = \mu$, where μ is a scalar constant learned through parameter estimation. It can be intuitively understood as the bias of the detection score in the regular object or other object cases. For example, it tends to be a positive (negative) value for the "regular (other) object" case.

Covariance function: As aforementioned analysis, the covariance function is decomposed into two parts, an inter-image part and an inner-image part. While the inter-image part is employed to regress the proposal-level detection score in the light of the proposals in the training set, the inner-image part is used to model the dependencies of the scores within one test image. To define the inter-image covariance function for a proposal pair belonging to different images, it needs to design a representation for each proposal so that their similarity can be readily measured. We leverage the spatial relationship between a proposal and the proposal with the maximum detection score within the same image as this representation. More specifically, assuming the maximum-scored proposal in an image I^i is s^i_{max}, the representation of a proposal s in I^i is defined as,

$$\phi(s) = [\mathrm{IoU}(s, s^i_{max}), c(s, s^i_{max})], \tag{6.3}$$

where $\mathrm{IoU}(s, s^i_{max})$ denotes the intersection-over-union between s and s^i_{max} and $c(s, s^i_{max})$ denotes the normalized distances between the centers of s and s^i_{max}. Note that these two measurements reflect a proposal's overlapping degree, distance to the maximum-scored proposal and indirectly the size of the proposal. Intuitively, these factors could be used to predict the detection score value of a proposal.

With this representation, we can define the inter-image covariance function $k_{inter}(s, s')$ of s and

s' as,

$$\exp\Big(-\frac{1}{2}\big(\phi(s)-\phi(s')\big)^T \mathrm{diag}(\gamma)\big(\phi(s)-\phi(s')\big)\Big), \tag{6.4}$$

where $\mathrm{diag}(\gamma)$ is a diagonal weighting matrix to be learned.

The inner-image covariance function serves as one of the key contributions of this work, which poses a smoothness constraint over the scores of the overlapped object proposals in an image. For a pair of inner-image proposals s and s', we define the inner-image covariance function as follows [1],

$$k_{inner}(s,s') = \frac{2S(s\cap s')}{S(s\cap s')+S(s\cup s')}, \tag{6.5}$$

where S stands for the area. Note that the formula is variant to standard intersection-over-union [31] commonly used as detection metric. The reason why we define it like this is because it is exactly χ^2 kernel and can guarantee the covariance matrix to be positive definite [104].

With both the inter-image and inner-image covariance function, we can obtain the overall covariance function of any proposal pair s and s' as,

$$k(s,s') = a\cdot k_{inner}(s,s') + b\cdot k_{inter}(s,s'), \tag{6.6}$$

where a, b are hyper-parameters regulating the weights of these two kernel functions.

Hyper-parameter Estimation

In this part, we introduce the hyper-parameter learning for the GPs. Still, we use regular images for description. In the definition of the mean and covariance functions of the GP, we introduce the hyper-parameters $\theta = \{\mu, \gamma, a, b\}$. We estimate the hyper-parameters by minimizing the negative logarithm of the marginal likelihood of all the detection scores of the training proposals given the hyper-parameters,

$$-L = -\log p(f(\mathcal{S})|\mathcal{S},\theta), \tag{6.7}$$

where $\mathcal{S}$ denotes the training proposals and $f(\mathcal{S})$ denotes their detection scores. We use the toolbox introduced in [87] for hyper-parameter optimization.

Test Image Evaluation

For class $\mathcal{C}$, let $\mathbf{s_r}$ be a set of proposals of regular training images and $\mathbf{f_r}$ be their detection scores. We can establish the covariance matrix K for the training data. Given a target set of proposals $\mathbf{s_t}$ from a test image and their detection scores $\mathbf{f_t}$, the joint distribution of $\mathbf{f_r}, \mathbf{f_t}$ can be written as,

[1] If two proposals s and s' are from different images, $k_{inner}(s,s') = 0$

$$\begin{bmatrix} \mathbf{f_r} \\ \mathbf{f_t} \end{bmatrix} \sim \mathcal{N}\Big(\begin{bmatrix} \mu \\ \mu \end{bmatrix}, \begin{bmatrix} K & k(\mathbf{s_r}, \mathbf{s_t}) \\ k(\mathbf{s_r}, \mathbf{s_t})^T & k(\mathbf{s_t}, \mathbf{s_t}) \end{bmatrix} \Big), \tag{6.8}$$

where μ is the mean vector, $k(\mathbf{s_r}, \mathbf{s_t})$ calculates the inter-image covariance matrix between training set and testing set and $k(\mathbf{s_t}, \mathbf{s_t})$ calculates the inner-image covariance of the test data. The fitting likelihood of the testing set to the generative model of the regular images can be expressed as,

$$\begin{aligned} \mathbf{f_t}|\mathbf{f_r} \sim \mathcal{N}\Big(& \mu + k(\mathbf{s_r}, \mathbf{s_t})^T K^{-1}(\mathbf{f_r} - \mu), \\ & k(\mathbf{s_t}, \mathbf{s_t}) - k(\mathbf{s_r}, \mathbf{s_t})^T K^{-1} k(\mathbf{s_r}, \mathbf{s_t}) \Big). \end{aligned} \tag{6.9}$$

Similarly, we can obtain the likelihood of the testing set given the "other class" training set. After obtaining the likelihood of the testing set given both regular training data and "other class" training data, we can compute the logarithm of the overall fitting likelihood of $\mathbf{f_t}$ as

$$\max\big(\log p(\mathbf{f_t}|\mathbf{f_r}), \log p(\mathbf{f_t}|\mathbf{f_o})\big), \tag{6.10}$$

where $\mathbf{f_o}$ represents the scores of "other class" training set. For either regular or "other class" test images, they could fit one of the generative models better than the unusual images. In other words, unusual images are supposed to obtain lower values in Eq. (6.10). Since the score obtained from Eq. (6.10) is negative (logarithm of a probability), we use the negative value of the score as the unusuality measurement.

6.6 Experiments

6.6.1 Experimental Settings

In this paper, we use the pre-trained CNN model [97] as feature extractors for object detector learning. Specifically, we use the activations of both the second fully-connected layer and the last convolutional layer as the representation of the object proposal or the whole image. Feeding an image into the CNN model, the activations of a convolutional layer are $n \times m \times d$ (e.g., $14 \times 14 \times 512$ for the last convolutional layer) with n, m corresponding to different spatial locations and d the number of feature maps. Given a proposal, we aggregate the convolutional features covered by it via max pooling to obtain the proposal-level convolutional features. We perform $L2$ normalization to these two types of features separately and concatenate them as the final representation. The dimensionality of the features is 4,608.

For each class, we construct GP based generative models for regular images and "other class" images separately. For regular images, we initialize the value of the mean function as 3 and for "other class" images we set the initial value to be -3. The hyper-parameters a, b in Eq. (6.6) are both initialized to be 0.5 and γ is initialized randomly. We use the top-20 scored proposals of each image for both generative model construction and test image evaluation. The test data of each class is divided into three parts including regular images, unusual images and images belonging to other classes. We label unusual images as 1 and label regular and "other class" images as -1. Mean Average Precision (mAP) is employed to evaluate the performances of the approaches.

6.6.2 Experimental Results

Alternative Solutions

TABLE 6.2: Experimental results. Average precision for each class and mAP are reported.

Methods	aeroplane	apple	bicycle	boat	building	bus	car	chair	cow	dinging table	
Positive-negative Ratio	58.0	26.6	50.4	52.4	60.0	37.8	55.4	48.7	31.6	28.8	
Global SVM	88.8	70.8	81.3	82.9	85.5	76.4	87.6	69.7	61.7	79.8	
MIL + Max	86.9	70.0	85.0	78.8	81.7	77.6	87.8	70.5	63.9	76.4	
MIL + Max + Gaussian	86.0	72.1	83.1	78.5	74.5	76.3	83.2	59.3	56.7	68.4	
MIL + Top 20	86.7	78.3	86.6	86.9	79.6	75.2	86.5	64.0	63.8	56.8	
Sparse coding (200)	86.9	48.6	80.6	81.0	82.8	57.4	82.8	71.7	56.1	72.2	
Sparse coding (4,000)	93.6	74.5	89.8	86.7	94.5	86.1	92.8	78.7	76.8	**86.0**	
Ours	**95.4**	**82.2**	**91.2**	**93.0**	**94.6**	**92.8**	**95.1**	**92.8**	**92.0**	74.8	
Methods	horse	house	motorbike	road	shoes	sofa	street	table lamp	train	tree	**mAP**
Positive-negative Ratio	23.9	47.4	30.9	48.2	56.4	39.7	42.7	16.9	28.6	44.7	41.4
Global SVM	73.3	82.0	75.6	81.3	88.2	77.7	73.8	66.5	69.2	73.9	77.3
MIL + Max	70.3	80.0	74.8	78.1	87.7	76.4	69.1	65.1	67.3	77.0	76.3
MIL + Max + Gaussian	63.1	74.6	65.9	66.1	85.8	69.7	55.5	60.5	64.1	69.8	70.7
MIL + Top 20	63.7	76.4	76.9	73.6	90.3	69.7	63.7	52.3	67.2	75.2	73.7
Sparse coding (200)	61.5	71.3	61.0	80.1	82.3	80.2	84.1	52.3	65.5	57.6	70.8
Sparse coding (4,000)	80.0	89.3	75.5	89.9	87.2	87.7	91.1	67.9	81.9	78.9	84.4
Ours	**85.4**	**94.4**	**85.0**	**90.8**	**95.3**	**88.9**	**94.8**	**78.3**	**91.3**	**85.0**	**89.7**

We compare our method to the following methods.

Positive-negative Ratio If we apply an object detector to the image regions, considerable portion of the regions of a regular image should be positively scored. While on the contrary, images of other classes are supposed to have negatively-scored proposals only. Based on this intuitive assumption, we use the ratio of positive proposal number to the number of negative proposals within one image

as its representation to construct two Gaussian models for regular images and "other class" images separately. Given a test image, we determine whether it is unusual via evaluating its fitting degree to these two Gaussians.

Global SVM According to the analysis in **Postulate I** in Section 6.4, the classification score of an image reflects the degree of containing the regular object-of-interest and the scores of the three types of images (regular, unusual, other class) should form the relationship of $f(I^o) < f(I^u) < f(I^r)$. For this method, we train a classifier for each class based on the global features of the images using linear SVM [33] where regular images are used as positive data and "other class" images are treated as negative data. Assuming the mean of the decision scores of unusual images is 0, we use negative absolute value of the decision score $-|f(I^t)|$ as the unusuality measurement for a test image I^t.

MIL + Max The global representation of an image is a mixture of the patterns of both the object-of-interest and the background. To avoid the distraction influence of the background, for the second solution we use the maximum proposal-level score $f_{max}(I^t)$ as the decision score of each image based on the object detector learned from MIL. Similarly we use $-|f_{max}(I^t)|$ as the unusuality measurement.

MIL + Max + Gaussian Different from above **MIL + Max** strategy, we take into consideration the uncertainty of the distribution of the maximum detection scores via modelling the maximum scores of regular images $\mathbf{I^r}$ and "other class" images $\mathbf{I^o}$ using two Gaussian distributions separately. We use maximum likelihood to estimate the parameters of these two Gaussians (means and variances). Given a test image I^t, we can calculate the likelihood of the image belonging to regular images as $p(I^t|\mathbf{I^r})$ and similarly the possibility of belonging to other classes as $p(I^t|\mathbf{I^u})$. Since an unusual image is expected to be able to fit neither of these two models, we set the final score of a test image as $-\max(p(I^t|\mathbf{I^r}), p(I^t|\mathbf{I^u}))$.

MIL + Top k Instead of using the maximum score only, for this method, we obtain the image-level score $f_{topk}(I^t)$ of a test image I^t by averaging the top k scores of its proposals. And the final score for an image is $-|f_{topk}(I^t)|$.

Sparse coding Similar to [126], we use sparse coding based reconstruction error as the criterion for unusual image detection. The assumption is that both regular images and "other class" images can be well reconstructed by their corresponding dictionaries. For each class, we learn dictionaries for regular images and "other class" images separately. We try dictionary size 200, 4,000 and 5,000. Given a test image I^t, we infer the coding vectors of its proposals and calculate the reconstruction residues of the proposals. Let r_r^t be the mean residue for this image calculated based on the dictionary learned from regular images and r_o^t be the mean residue based on the dictionary learned from "other

class" images. For an unusual image, the errors of both models will be large. Thus the unusuality measurement can be calculated as $\min(r_r^t, r_o^t)$.

Quantitative Results

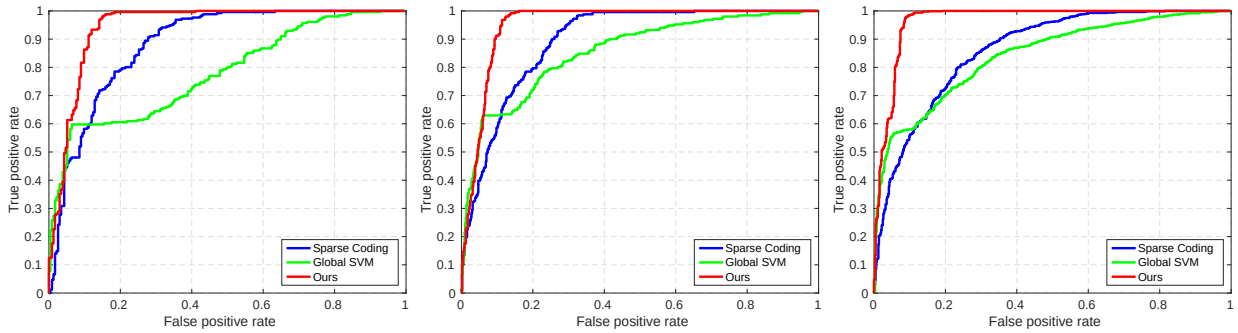


FIGURE 6.5: ROC curve for Sparse coding, Global SVM and our method on three categories. From left to right: boat, motorbike, shoes.

Table 6.2 shows the quantitative results. As can be seen, our method outperforms other compared methods. Also we show the ROC performances of our method and two most competitive methods on some example categories in Fig. 6.5. Both these two measurements demonstrate the effectiveness of the proposed method.

The proposal ratio based method performs worst among these methods which indicates that the unusuality detection cannot be achieved by simply counting the number of positive and/or negative proposals. There are two reasons. The first is that the number of proposals varies between different images and the second reason is that for some unusual object images e.g., images of severely damaged cars, there may be no positively scored proposals detected.

The next four methods are classification-based methods. While the first three use single score per image from either the global image or the region with maximum detection score, **MIL+Top k** utilizes multiple region scores but treat them as i.i.d. **Global SVM** achieves a mAP of 77.3% (when using fully-connected features only, we obtain 75.4%) which to some extent justifies **Postulate I**. However, as illustrated in Fig. 6.3, this strategy fails to distinguish some unusual images that obtain extreme high or low decision scores. A drawback of using image-level representation is that the background can influence the decision score especially when the background dominates the image. Multi-instance learning is supposed to be a remedy because it makes it possible to focus on the object-of-interest via considering the proposal with maximum detection score. But using maximum detection score alone may risk missing the unusual part of the object. From Table 6.2, we can see **MIL+Max**

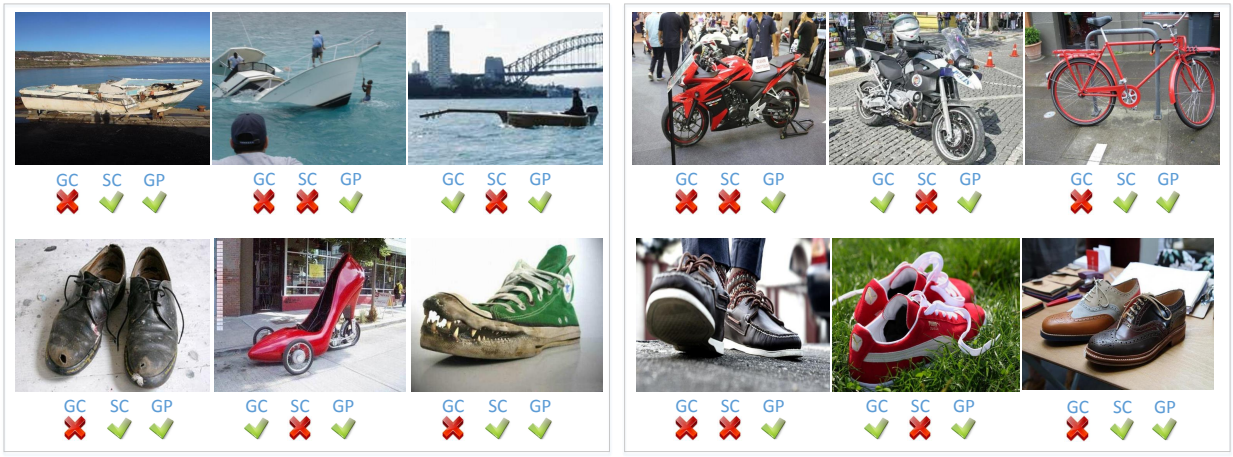


FIGURE 6.6: Qualitative performance comparison between our method (GP) and two alternative solutions, Global SVM (GC) and Sparse coding (SC). Left column displays the false negative examples and right column displays the false positive examples. Three categories are boat, shoes and motorbike.

obtains comparable results to **Global SVM**. To take into consideration the uncertainty of the detection scores, rather than directly using the maximum detection scores, we construct Guassian models for the maximum scores of regular images and "other class" images separately and determine whether an image is unusual via evaluating its fitting likelihood to these two Gaussian models. However, the performance degrades to 70.7%. The reason may be that the distribution of the maximum detection scores is not strictly Gaussian. Instead of using the maximum detection score of each image, in **MIL+Top20**, we aggregate the top 20 scores of each image via average pooling. Benefiting from this strategy, the performances on some classes like *apple, boat* are obviously boosted. However, on some other classes such as *horse, table lamp* it shows inferior performance to **Global SVM** and **MIL+Max**. As can be seen, our method significantly outperforms this strategy on all the classes. This big gap may to a large extent result from our capabilities of modelling the inter-dependencies of the proposal-level scores within one image.

For sparse coding, we first test the performance using dictionaries of size 200 as [126] and the result is unsatisfactory which means 200 bases are not sufficient to cover the feature spaces of regular images or "other class" images. When the dictionary size is increased to 4,000, the performance is significantly improved. But after that continuing to increase the dictionary size (we test **5,000**) can lead to no improvement any more. Our method outperforms sparse coding by 5.3%. Apart from effectiveness, our method is also more efficient than sparse coding. Given a test image, while sparse coding needs to infer the coding vector for the high-dimensional appearance features our method

works on quite low-dimensional space as defined in Eq. (6.3).

Qualitative Results

Fig. 6.6 demonstrates the qualitative comparison between our method and two compared methods **Global SVM (GC)** and **Sparse coding (SC)** on three object categories that are boat, motorbike and shoes. Comparing to our method, GC suffers from two drawbacks: 1) it subjects to the distraction influence of the background, and 2) it may ignore the fine details of the objects. Due to the influence of the background, GC may mistakenly classify the regular object within complex background into unusual object like the "shoes" on the right side of Fig. 6.6. Also, only looking at the global appearance makes it hard for GC to identify some unusual objects with fine unusualities such as the "broken boat" and "broken shoes" in Fig. 6.6. SC has similar deficiency that is it can be distracted or even dominated by the background. For example, the "capsized boat" is identified as "regular boat" while "regular motorbike" within complex background is regarded as "unusual motorbike". Comparing to these two methods our method is more robust. While using detection scores enables us to getting rid of the distraction influence of the background, modelling the inter-dependencies of the detection scores at multiple regions can help us to effectively discover the finer unusualities.

6.7 Summary

We have proposed a novel approach for the task of unusual object identification in an "open world" setting via inspecting the detection score patterns of an image. We propose to use Gaussian Processes to model the values as well the spatial distribution of the detection scores. It shows superior performance to some compared methods on a large dataset presented in this work. To summarize, our contributions are:

- We propose a large dataset and present a more realistic "open world evaluation protocol for the task of unusual-object identification from images.
- Instead of directly modelling high-dimensional region-level CNN features, we map them using scalar detection scores, which can help to git rid of the distraction influence of noisy content.
- We propose a novel approach for unusual-object detection by looking into the detection score values as well as the spatial distributions of the detection scores of the image regions. We propose to use Gaussian Processes (GP) to simultaneously model the detection score at a single region and the score dependencies between multiple regions.

Chapter 7

Conclusion and future work

7.1 Conclusion

In this thesis, we study how to build effective local visual pattern modelling methods for image and video classification. Specifically, we addressed the following problems:

- Considering the incompatibility between the video format and the input format of traditional CNN structure, in Chapter 3 we propose a flexible CNN structure that can handle a video with arbitrary number of frames. The key module of the structure is composed of an encoding layer and a pooling layer. While the former is used to map the frame-level features into a space more suitable for pooling, the latter aggregates the frame-level coding vectors into a video-level representation. To explicitly exploit the weak temporal structure of the video, we realize the pooling layer as a temporal pyramid pooling layer. Also, our network is a hybrid network that can take multiple frame-level representations as input. In this work, we exploit both appearance information and motion information. However, our network structure can generalize beyond these two kinds of features to model other frame-level patterns.

- The CNN structure proposed in Chapter 3 is flexible that can handle varying number of video frames. It, however, suffers from two drawbacks. First, the temporal pyramid pooling layer captures only weak temporal structure and may risk losing the important clues hidden in the temporal space. Second, directly performing encoding on top of the frame-level features requires a large number of parameters, which may result in parameter explosion. To better exploit the important temporal information as well as maintaining a controllable amount of parameters, we propose a convolutional pooling network in Chapter 4. It extracts the dynamic information via performing convolution operation over the temporal domain. The key novelty is that instead

of applying convolution directly on the original frame-level features, we view the temporal evolution of each feature dimension as a 1D signal and learn a unique filter bank for each of these 1D signals. This strategy can substantially reduce the amount of model parameters. Comparing to traditional pooling methods e.g., sum pooling, max pooling, our method leads to obvious performance improvement on video based action recognition. Note that our method is not limited to video analysis but can be applied to other time-series tasks like sentence classification or speech recognition.

- In Chapter 5, we propose a hybrid generative model based Fisher vector coding scheme for image classification. Specifically, we study how to effectively encode a set of high-dimensional local features in an image towards better image classification performance. Different from traditional Fisher vector coding and its variations that directly model the generation process of a local feature, we decompose it into a discriminative part and a residual part and devise a generative model based on this decomposition. The decomposition is realised by a supervised coding method where an active dimension of the coding vector indicates the emergence of an elementary pattern. Our method shares the merits of both Fisher vector coding and supervised coding and consequently leads to superior classification performance. Besides better performance, our method is more theoretical complete because our model considers a more general case where a local feature consists of an informative component and a noisy component.

- Local visual patterns of an image may interact with each other to tell more about the image. In Chapter 6, we study the challenging problem of identifying unusual instances of known objects within an "open world" setting via modelling the local visual pattern distribution. In order to get rid of the distraction influence of the irrelevant content, we map the region-level CNN features into detection scores and propose to identify the unusual instance of an object by inspecting the score distribution. We propose to use Gaussian Process (GP) to model the score distribution of both regular-object images and other-objects images. A new covariance function is devised to both non-parametrically model the detection score at a single region, and capture the inter-dependencies between scores over multiple regions. Note that unlike the conventional use of GP in computer vision, our model does not assume that the region scores of an image are i.i.d. This treatment allows our method to capture the spatial dependencies between detection scores, which turns out to be crucial for identifying unusual instances of an object.

- Essentially, the works in Chapter 3, 4, 5 are all focusing on how to effectively encode the local

visual patterns of videos or images. The methods in Chapter 3 and Chapter 4 can be regarded as two forms of supervised encoding. They use the video label to guide the encoding of the local features to make the coding vectors more discriminative and both of them adopt a three-step encoding pipeline: encoding, pooling and classification. The difference between the methods in these two chapters lie in how they perceive the videos. In Chapter 3, we treat the frame-level representation as the local feature and in Chapter 4 we assume the feature dimensions are independent and the local features are some temporal segments of such dimensions. Convolutional Neural Network provides a flexible way to model the different forms of local features. In Chapter 5, we melt the supervised encoding strategies into another standard unsupervised encoding scheme, Fisher vector coding, to boost the image classification performance. The core idea is to use the supervised coding vector to decompose a local feature into a discriminative part and a residual part. The encoding methods studied in the aforementioned three chapters can generalize beyond visual classification and apply to other tasks. In Chapter 6, we first obtain the object detection scores for the local regions of an image and this essentially can be regarded as a supervised encoding process as well. On top of these detection scores, we investigate the importance of modelling the interactions between these regions for irregular object identification. And the other way around, we believe taking into consideration the context information can also benefit the local feature encoding. We may explore this in the future work.

7.2 Future Work

In addition to the problems addressed in this thesis, we also point out the following open problems that we expect to explore in the future:

- Our temporal pyramid pooling network in Chapter 3 and convolutional pooling network in Chapter 4 can generalize beyond video classification to other time-series tasks. We plan to apply these two types of networks to tasks like sentence classification and speech recognition to evaluate their performance and make further improvement according to the characteristics of the specific task.

- The models proposed in this thesis exploit supervised information to model the local visual patterns which tends to require a considerable scale of training instances to guarantee the generalization ability of the learned models. In most practical problems, however, there can be few

or even one instance existing for a class. How to learn an effective model using few training instances is a problem worth exploring.

- To further investigate the unusual object identification task, we plan to study how to use Graph CNN to model the inter-dependencies between multiple local regions, where each local region in an image can be abstracted into a node in a graph.

- The hybrid Fisher vector coding scheme proposed in this thesis has an outer-product form, which results in a high-dimensional coding vector. Next step, we are interested in improving the model so that it can create compact coding vector without sacrificing the representation power.

References

[1] Matconvnet: Cnns for matlab. `http://www.vlfeat.org/matconvnet/`.

[2] R. Arandjelović and A. Zisserman. All about VLAD. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2013.

[3] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson. From generic to specific deep representations for visual recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn. Workshops*, 2014.

[4] F. R. Bach and M. I. Jordan. A probabilistic interpretation of canonical correlation analysis. Technical report, 2005.

[5] H. Bay, A. Ess, T. Tuytelaars, and L. J. V. Gool. Speeded-up robust features (surf). *Comp. Vis. Image Understanding*, 110(3):346–359, 2008.

[6] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016.

[7] O. Boiman and M. Irani. Detecting irregularities in images and in video. *Int. J. Comp. Vis.*, 2007.

[8] M. Bregonzio, S. Gong, and T. Xiang. Recognising action as clouds of space-time interest points. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2009.

[9] Z. Cai, L. Wang, X. Peng, and Y. Qiao. Multi-view super vector for action recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2014.

[10] C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27, 2011.

[11] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1-3):131–159, 2002.

[12] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *Proc. Brit. Mach. Vis. Conf.*, 2014.

[13] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *Proc. Brit. Mach. Vis. Conf.*, 2014.

[14] M. J. Choi, J. Lim, A. Torralba, and A. Willsky. Exploiting hierarchical context on a large database of object categories. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2010.

[15] M. J. Choi, A. Torralba, and A. S. Willsky. Context models and out-of-context objects. *Pattern Recognition Letters*, 2012.

[16] M. Cimpoi, S. Maji, and A. Vedaldi. Deep convolutional filter banks for texture recognition and segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015.

[17] R. G. Cinbis, J. Verbeek, and C. Schmid. Approximate fisher kernels of non-iid image models for image categorization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2015.

[18] R. G. Cinbis, J. J. Verbeek, and C. Schmid. Image categorization using Fisher kernels of non-iid image models. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2012.

[19] A. Coates and A. Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *Proc. Int. Conf. Mach. Learn.*, 2011.

[20] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Proc. Eur. Conf. Comp. Vis.*, 2004.

[21] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2005.

[22] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *Proc. Eur. Conf. Comp. Vis.*, 2006.

[23] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *Proc. Eur. Conf. Comp. Vis.*, 2006.

[24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2009.

[25] M. Dixit, S. Chen, D. Gao, N. Rasiwasia, and N. Vasconcelos. Scene classification with semantic fisher vectors. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015.

[26] C. Doersch, A. Gupta, and A. A. Efros. Mid-level visual element discovery as discriminative mode seeking. In *Proc. Adv. Neural Inf. Process. Syst.*, 2013.

[27] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *IN n VS-PETS*, 2005.

[28] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015.

[29] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015.

[30] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: A deep convolutional activation feature for generic visual recognition. In *Proc. Int. Conf. Mach. Learn.*, 2014.

[31] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comp. Vis.*, 2010.

[32] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, 2008.

[33] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 2008.

[34] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2005.

[35] B. Fernando, E. Gavves, J. Oramas, A. Ghodrati, and T. Tuytelaars. Modeling video evolution for action recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015.

[36] J. C. Gemert, J.-M. Geusebroek, C. J. Veenman, and A. W. Smeulders. Kernel codebooks for scene categorization. In *Proc. Eur. Conf. Comp. Vis.*, 2008.

[37] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2014.

[38] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. *Multi-scale Orderless Pooling of Deep Convolutional Activation Features*. 2014.

[39] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *Proc. Eur. Conf. Comp. Vis.*, 2014.

[40] R. Hamid, A. Johnson, S. Batta, A. Bobick, C. Isbell, and G. Coleman. Detection and explanation of anomalous activities: representing activities as bags of event n-grams. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2005.

[41] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2015.

[42] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Proc. Adv. Neural Inf. Process. Syst.*, 1998.

[43] A. Jain, A. Gupta, M. Rodriguez, and L. Davis. Representing videos using mid-level discriminative patches. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2013.

[44] M. Jain, H. Jegou, and P. Bouthemy. Better exploiting motion for better action recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2013.

[45] H. Jegou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid. Aggregating local image descriptors into compact codes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(9):1704–1716, 2012.

[46] H. Jhuang, T. Serre, L. Wolf, and T. Poggio. A biologically inspired system for action recognition. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2007.

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

[48] Z. Jiang, Z. Lin, and L. S. Davis. Learning a discriminative dictionary for sparse coding via label consistent k-svd. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2011.

[49] M. Juneja, A. Vedaldi, C. Jawahar, and A. Zisserman. Blocks that shout: Distinctive parts for scene classification. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2013.

[50] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2014.

[51] M. S. Khurram Soomro, Amir Roshan Zamir. Ucf101: A dataset of 101 human actions classes from videos in the wild. In *arXiv:1212.0402*, 2012.

[52] A. Klser, M. Marszalek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. In *Proc. Brit. Mach. Vis. Conf.*, 2008.

[53] J. Krapac, J. J. Verbeek, and F. Jurie. Modeling spatial layout with fisher vectors for image categorization. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2011.

[54] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. Adv. Neural Inf. Process. Syst.* 2012.

[55] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2011.

[56] I. Laptev and T. Lindeberg. Space-time interest points. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2003.

[57] I. Laptev and T. Lindeberg. Local descriptors for spatio-temporal recognition. In *In First International Workshop on Spatial Coherence for Visual Motion Analysis*, 2004.

[58] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2008.

[59] S. Lazebnik and M. Raginsky. Supervised learning of quantizer codebooks by information loss minimization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(7):1294–1309, 2009.

[60] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *Proc. Adv. Neural Inf. Process. Syst.*, 2007.

[61] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *Proc. Adv. Neural Inf. Process. Syst.*, 2007.

[62] Y. Li, L. Liu, C. Shen, and A. van den Hengel. Mid-level deep pattern mining. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015.

[63] X.-C. Lian, Z. Li, B.-L. Lu, and L. Zhang. Max-margin dictionary learning for multiclass image categorization. In *Proc. Eur. Conf. Comp. Vis.*, 2010.

[64] T. Lin, A. RoyChowdhury, and S. Maji. Bilinear CNN models for fine-grained visual recognition. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2015.

[65] L. Liu, C. Shen, and A. van den Hengel. The treasure beneath convolutional layers: Cross-convolutional-layer pooling for image classification. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015.

[66] L. Liu, C. Shen, L. Wang, A. van den Hengel, and C. Wang. Encoding high dimensional local features by sparse coding based Fisher vectors. In *Proc. Adv. Neural Inf. Process. Syst.*, 2014.

[67] L. Liu and L. Wang. A scalable unsupervised feature merging approach to efficient dimensionality reduction of high-dimensional visual data. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2013.

[68] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comp. Vis.*, 60(2):91–110, 2004.

[69] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, 1981.

[70] Y. Luo, L.-F. Cheong, and A. Tran. Actionness-assisted recognition of actions. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2015.

[71] S. G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, (7):33973415, 1993.

[72] M. Marszałek, I. Laptev, and C. Schmid. Actions in context. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2009.

[73] S. Mathe and C. Sminchisescu. Dynamic eye movement datasets and learnt saliency models for visual action recognition. In *Proc. Eur. Conf. Comp. Vis.* 2012.

[74] P. K. Matikainen, M. Hebert, and R. Sukthankar. Trajectons: Action recognition through the motion analysis of tracked features. In *Proc. IEEE Int. Conf. Comp. Vis. Workshops*, 2009.

[75] R. Messing, C. Pal, and H. Kautz. Activity recognition using the velocity histories of tracked keypoints. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2009.

[76] J. Y. Ng, M. J. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015.

[77] B. Ni, P. Moulin, X. Yang, and S. Yan. Motion part regularization: Improving action recognition via trajectory selection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015.

[78] A. Oikonomopoulos, I. Patras, and M. Pantic. Spatiotemporal salient points for visual recognition of human actions. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 36(3):710–719, 2005.

[79] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Is object localization for free? - weakly-supervised learning with convolutional neural networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015.

[80] S. Park, W. Kim, and K. M. Lee. Abnormal object detection by canonical scene-based contextual model. In *Proc. Eur. Conf. Comp. Vis.*, 2012.

[81] X. Peng, L. Wang, X. Wang, and Y. Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. In *arXiv:1405.4506*, 2014.

[82] X. Peng, C. Zou, Y. Qiao, and Q. Peng. Action recognition with stacked fisher vectors. In *Proc. Eur. Conf. Comp. Vis.* 2014.

[83] X. Peng, C. Zou, Y. Qiao, and Q. Peng. Action recognition with stacked fisher vectors. In *Proc. Eur. Conf. Comp. Vis.* 2014.

[84] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *Proc. Eur. Conf. Comp. Vis.*, 2010.

[85] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *Proc. Eur. Conf. Comp. Vis.*, 2010.

[86] M. Raptis and S. Soatto. Tracklet descriptors for action modeling and video analysis. In *Proc. Eur. Conf. Comp. Vis.*, 2010.

[87] C. E. Rasmussen and H. Nickisch. Gaussian processes for machine learning (gpml) toolbox. *J. Mach. Learn. Res.*, 2010.

[88] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. 2005.

[89] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn. Workshops*, 2014.

[90] M. S. Ryoo, B. Rothrock, and L. Matthies. Pooled motion features for first-person videos. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015.

[91] B. Saleh, A. Elgammal, J. Feldman, and A. Farhadi. Toward a taxonomy and computational models of abnormalities in images. 2016.

[92] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local svm approach. In *ICPR*, 2004.

[93] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. In *ACM Multimedia Conference*, 2007.

[94] M. Simon and E. Rodner. Neural activation constellations: Unsupervised part model discovery with convolutional networks. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2015.

[95] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep fisher networks for large-scale image classification. In *Proc. Adv. Neural Inf. Process. Syst.*, 2013.

[96] K. Simonyan and A. Zisserman. Two-Stream Convolutional Networks for Action Recognition in Videos. In *Proc. Adv. Neural Inf. Process. Syst.*, 2014.

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

[98] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of International Conference on Learning Representations*, 2015.

[99] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. *CoRR*, abs/1502.04681, 2015.

[100] J. Sun, X. Wu, S. Yan, L.-F. Cheong, T.-S. Chua, and J. Li. Hierarchical spatio-temporal context modeling for action recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2009.

[101] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2015.

[102] V. Sydorov, M. Sakurada, and C. H. Lampert. Deep fisher kernels - end to end learning of the fisher kernel gmm parameters. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2014.

[103] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *Int. J. Comp. Vis.*, 2013.

[104] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2011.

[105] A. Vezhnevets and V. Ferrari. Object localization in imagenet by looking out of the window. In *Proc. Brit. Mach. Vis. Conf.*, 2015.

[106] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Action Recognition by Dense Trajectories. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2011.

[107] H. Wang and C. Schmid. Action recognition with improved trajectories. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2013.

[108] H. Wang and C. Schmid. Action recognition with improved trajectories. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2013.

[109] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2010.

[110] L. Wang, Y. Qiao, and X. Tang. Motionlets: Mid-level 3d parts for human motion recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2013.

[111] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015.

[112] X. Wang, B. Wang, X. Bai, W. Liu, and Z. Tu. Max-margin multiple instance dictionary learning. In *Proc. Int. Conf. Mach. Learn.*, 2013.

[113] G. Willems, T. Tuytelaars, and L. Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *Proc. Eur. Conf. Comp. Vis.*, 2008.

[114] K.-Y. K. Wong and R. Cipolla. Extracting spatiotemporal interest points using global information. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2007.

[115] T. Xiao, Y. Xu, K. Yang, J. Zhang, Y. Peng, and Z. Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015.

[116] Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative cnn video representation for event detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015.

[117] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2009.

[118] J. Yang, K. Yu, and T. Huang. Supervised translation-invariant sparse coding. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2010.

[119] D. Yoo, S. Park, J. Lee, and I. Kweon. Fisher kernel for deep neural activations, 2014.

[120] K. Yu and T. Zhang. Improved local coordinate coding using local tangents. In *Proc. Int. Conf. Mach. Learn.*, 2010.

[121] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015.

[122] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang. Real-time action recognition with enhanced motion vector cnns. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016.

[123] N. Zhang, J. Donahue, R. Girshick, and T. Darrell. Part-based r-cnns for fine-grained category detection. In *Proc. Eur. Conf. Comp. Vis.*, 2014.

[124] N. Zhang, R. Farrell, F. Iandola, and T. Darrell. Deformable part descriptors for fine-grained recognition and attribute prediction. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2013.

[125] Y. Zhang, J. Wu, and J. Cai. Compact representation for image classification: To choose or to compress? In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2014.

[126] B. Zhao, L. Fei-Fei, and E. P. Xing. Online detection of unusual events in videos via dynamic sparse coding. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2011.

[127] J. Zheng, Z. Jiang, R. Chellappa, and J. P. Phillips. Submodular attribute selection for action recognition in video. In *Proc. Adv. Neural Inf. Process. Syst.* 2014.

[128] H. Zhong, J. Shi, and M. Visontai. Detecting unusual activity in video. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2004.

[129] X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. In *Proc. Eur. Conf. Comp. Vis.*

[130] J. Zhu, B. Wang, X. Yang, W. Zhang, and Z. Tu. Action recognition with actons. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2013.

Appendix

.1 Matching pursuit based optimization for Eq. (5.9)

We introduce, in detail, a matching pursuit based method to optimize the following problem (Eq. (5.9) in the main text of our paper).

$$\begin{aligned} &\min_{\mathbf{u}_d,\mathbf{u}_c} \|\mathbf{x}-\mathbf{B}_d\mathbf{u}_d-\mathbf{B}_r\mathbf{u}_r\|_2^2+\lambda\|\mathbf{u}_d-\mathbf{c}\|_2^2 \\ &s.t.\ \|\mathbf{u}_d\|_0\le k_1,\quad \|\mathbf{u}_r\|_0\le k_2. \end{aligned} \tag{1}$$

Given the bases $\mathbf{B}_d\in\mathbb{R}^{d\times m_1}$ and $\mathbf{B}_r\in\mathbb{R}^{d\times m_2}$, matching pursuit will sequentially update one dimension of $\mathbf{u}_d$ (or $\mathbf{u}_r$) at each iteration in order to minimize Eq. (1). In practice, we firstly solve $\mathbf{u}_d$ and then $\mathbf{u}_r$. The optimization problem at each iteration is as follows:

for solving $\mathbf{u}_d$:

$$\begin{aligned} \min_{\mathbf{e}_{d_j},u_{d_j}} \|\mathbf{x}-\mathbf{B}_d\mathbf{u}_d^t-\mathbf{B}_r\mathbf{u}_c^t-\mathbf{B}_d\mathbf{e}_{d_j}u_{d_j}\|_2^2+ \\ \lambda\|\mathbf{u}_d^t-\mathbf{c}+\mathbf{e}_{d_j}u_{d_j}\|_2^2 \end{aligned} \tag{2}$$

for solving $\mathbf{u}_r$:

$$\min_{\mathbf{e}_{r_j},u_{r_j}} \|\mathbf{x}-\mathbf{B}_d\mathbf{u}_d^t-\mathbf{B}_r\mathbf{u}_r^t-\mathbf{B}_r\mathbf{e}_{r_j}u_{r_j}\|_2^2, \tag{3}$$

where $\mathbf{u}_d^t$ and $\mathbf{u}_c^t$ are the solutions for $\mathbf{u}_d$ and $\mathbf{u}_r$ at the tth iteration respectively. $\mathbf{e}_{d_j}$ and $\mathbf{e}_{r_j}$ are binary vectors with only one nonzero entry at the jth dimensions, $j\in[0,1,\cdots,m_1]$ for $\mathbf{e}_{d_j}$ and $j\in[0,1,\cdots,m_2]$ for $\mathbf{e}_{r_j}$. Therefore there are m_1 possible choices for $\mathbf{e}_{d_j}$ and m_2 possible choices for $\mathbf{e}_{r_j}$ respectively. They indicate which dimension of $\mathbf{u}_d$ ($\mathbf{u}_r$) is to be updated and the scalar u_{d_j} (u_{r_j}) denotes the value to be updated at the chosen dimension.

To solve $\mathbf{e}_{d_j}$ ($\mathbf{e}_{r_j}$), we simply test its all possible choices and for each candidate $\mathbf{e}_{d_j}$ ($\mathbf{e}_{r_j}$) its corresponding optimal $\bar{u}_{d_j}$ ($\bar{u}_{r_j}$) can be analytically worked out as:

$$\begin{aligned} \bar{u}_{d_j} &= \frac{\mathbf{r}^\top \mathbf{B}_{d_j} + \lambda c_j}{\mathbf{B}_{d_j}^\top \mathbf{B}_{d_j} + \lambda}, \\ \bar{u}_{c_j} &= \frac{\mathbf{r}^\top \mathbf{B}_{c_j}}{\mathbf{B}_{r_j}^\top \mathbf{B}_{c_j}}, \end{aligned} \tag{4}$$

where $\mathbf{B}_{d_j} = \mathbf{B}_d \mathbf{e}_{d_j}$ ($\mathbf{B}_{r_j} = \mathbf{B}_r \mathbf{e}_{r_j}$) is the jth column of $\mathbf{B}_{d_j}$ ($\mathbf{B}_{r_j}$). Thus the objective value in Eq. (2) (Eq. (3)) can be calculated by substituting $\mathbf{e}_{d_j}$ ($\mathbf{e}_{c_j}$) and its corresponding $\bar{u}_{d_j}$ ($\bar{u}_{r_j}$). We select the best $\mathbf{e}_{d_j}$ ($\mathbf{e}_{r_j}$) and its corresponding $\bar{u}_{d_j}$ ($\bar{u}_{r_j}$) which minimize Eq. (2) (Eq. (3)) as the solution, denoted by $\mathbf{e}^*_{d_j}$ ($\mathbf{e}^*_{r_j}$) and $u^*_{d_j}$ ($u^*_{r_j}$). Then $\mathbf{u}_d, \mathbf{u}_r$ can be updated as $\mathbf{u}_d^{t+1} = \mathbf{u}_d^t + \mathbf{e}^*_{d_j} u^*_{d_j}$ and $\mathbf{u}_r^{t+1} = \mathbf{u}_r^t + \mathbf{e}^*_{r_j} u^*_{r_j}$. To avoid the redundant calculation, we also define a residual term $\mathbf{r} = \mathbf{x} - \mathbf{B}_d \mathbf{u}_d^t - \mathbf{B}_r \mathbf{u}_r^t$ and update it by $\mathbf{r} \leftarrow \mathbf{r} - \mathbf{B}_d \mathbf{e}^*_{d_j} u^*_{d_j}$ and $\mathbf{r} \leftarrow \mathbf{r} - \mathbf{B}_r \mathbf{e}^*_{c_j} u^*_{c_j}$.