Efficient representation for image recognition

Zhang, Yu

2015

Zhang, Y. (2015). Efficient representation for image recognition. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/63616>

<https://doi.org/10.32657/10356/63616>

EFFICIENT REPRESENTATION FOR IMAGE RECOGNITION

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

ZHANG YU

May 2015

Abstract

Image representation plays the key role in image recognition. An image is usually represented by a feature vector generated using methods like bag-of-visual-words (BOV) model, Fisher Vector, sparse coding, etc. In recent years, high dimensional feature vectors have become popular. However, they usually cause severe memory problem.

In this dissertation, in order to achieve an efficient and accurate image representation for image recognition, we consider three key factors: feature generation speed, discriminance, and storage. We then propose three novel computational techniques to address the impact of the three factors on image recognition: exclusive vector quantization (ExVQ), hyper-spatial matching (HSM), and feature selection for dimensional reduction, respectively.

First, considering vector quantization (VQ) is the most time consuming step in many widely used feature generation methods, we propose a fast VQ method called ExVQ. In ExVQ, an exclusive nearest neighbor search is developed to quantize visual descriptors with the most similar visual codewords. While it attains classification accuracy similar to the state-of-the-art approx NN search, ExVQ is significantly faster, leading to improved feature generation speed.

Next, in order to improve the feature discriminance, HSM is proposed to characterize the spatial relationship among regions in images. Spatial information is important for many feature generation methods. Generally, to compute image spatial information, images are split into several regions in the same way. Unlike existing methods that usually compute the similarity of spatially corresponding regions, HSM computes the similarity of all region pairs in two images. Thus, HSM can encode more comprehensive spatial information. With HSM, two fast SVM classifier learning strategies are proposed, which are hundreds of times faster than general purpose HSM SVM solvers.

Finally, we propose to use feature selection to reduce the storage for high dimensional feature vectors in large-scale image recognition problems. A supervised mutual information based feature selection is used to compute the feature importance for feature vectors. According to the sorted importance values, feature vectors can easily achieve different compression ratios. Extensive experiments show that the proposed feature selection achieves better classification accuracy than that of the existing feature compression methods, when they use the same memory.

The three proposed methods can contribute to an efficient and accurate image recognition system. ExVQ can be used with most of existing feature generation methods to improve their generation speed without decreasing their recognition discriminance. Based on the extracted features, HSM can further improve their discriminance by considering more comprehensive spatial information. Since HSM can cause possible memory problems, with the aid of feature selection to decrease the storage of high dimensional feature vectors, HSM can be safely applied on large scale image recognition problems.

Acknowledgments

I would like to express my deepest gratitude to my advisor, **Dr. Wu Jianxin**, for his patience and guidance during the past years. I have always been inspired by Jianxin's vision about the future of computer vision. Without Jianxin's guidance, the fruitful achievements of this project would not have been possible.

I also would like to express my gratitude to my supervisor in Nanyang Technological University (NTU), **Dr. Cai Jianfei**. With his guidance, my research interest is broadened. His sincere encouragement has faithfully guided me throughout the last several years of my Ph.D. time.

In addition, I would like to thank my friends and the staffs in Center for Computational Intelligence for their encouragements and technical supports.

Finally and most of all, I would like to thank my parents and my wife, Mi Siya. Your love, patience, and encouragement supported me throughout the entire Ph.D. study period.

Zhang Yu

Contents

Abstract	i
Acknowledgments	iii
List of Figures	ix
List of Tables	xi
1 Introduction	1
1.1 Relationship to previous work	7
2 Literature Review	9
2.1 Image representations	9
2.2 Vector quantization	13
2.3 Spatial matching	15
2.4 Feature compression and feature selection	17
2.4.1 Feature compression	17
2.4.2 Feature selection	19
3 Exclusive Vector Quantization (ExVQ) for Fast Feature Generation	21
3.1 Effect of ANN VQ errors to classification	22
3.2 ANN search: inclusion vs. exclusion	23
3.3 Tree-structured ExVQ	27
3.4 Experiments for ExVQ	30
3.4.1 VQ time vs. classification accuracy	30
3.4.2 Properties of VQ errors	32
3.4.3 Effects of the tree level L and exclusion portion p	33
3.4.4 ExVQ and other related methods	34
3.5 Summary of the chapter	36

4	Hyper Spatial Matching for Accurate Image Representation	37
4.1	Hyper-spatial matching	38
4.1.1	Learn an overall relationship matrix R	38
4.1.2	Stability of the affinity and relationship matrices	41
4.2	Efficient learning of the SVM classifier	41
4.2.1	The spectral linearization strategy	42
4.2.2	The gradient approximation strategy	45
4.2.3	Dot product HSM SVM	48
4.2.4	Decorrelation effect of HSM	48
4.2.5	Hyper-Spatial Features	50
4.3	Experimental for HSM	51
4.3.1	HSM combined with BOV features	53
4.3.2	Hyper-Spatial Features combined with Fisher Vector and VLAD .	57
4.3.3	Discussions	59
4.4	Summary of the chapter	60
5	Feature Selection for Feature Dimension Reduction	63
5.1	Feature selection by mutual information based importance sorting	64
5.1.1	Feature selection vs. (bi-)linear projection	64
5.1.2	Mutual information (MI) based importance sorting	66
5.1.3	Entropy calculation and further quantization	67
5.1.4	Classification using quantized features	70
5.2	A novel image recognition system: feature selection with HSM	71
5.3	Experiments for feature selection	72
5.3.1	FS for image recognition	74
5.3.2	FS+HSM on high-dimensional feature vectors	80
5.4	Summary of the chapter	82
6	Conclusions and Future Work	85
6.1	Summary of contributions	85
6.2	Future work	86

List of Related Publications

References

91

List of Figures

1.1	The flowchart of an image recognition system.	3
3.1	VQ error rate and classification accuracy (one round) using FLANN hierarchical k -means tree, on Caltech 101, when $K = 256$	23
3.2	The histograms of ANN error ranks. The vertical axis is the frequency numbers shown <i>in natural logarithm</i> . We use FLANN hierarchical k -means tree with different check numbers on Caltech 101 when $K = 256$	24
3.3	Illustration of inclusive vs. exclusive NN search. The random projection violates the inclusive assumption. The exclusive strategy, however, can correctly exclude the red or the green clusters. Please refer to the second paragraph in Sec. 3.2 to understand what the color means. This figure should be viewed in color.	25
3.4	Tree-structured ExVQ: for a given query descriptor, in each node, the red or green part is excluded depending on outputs of the associated linear classifiers. This figure should be viewed in color	29
3.5	Histograms of error ranks of ExVQ on the four datasets with the codebook size $K = 256$ (blue) and $K = 1024$ (red). The horizontal axis is the error rank and the vertical axis shows the frequency number plot in natural logarithm.	33
3.6	The classification accuracy and the VQ time with the tree level L and fixed exclusion portion p	34
3.7	The classification accuracy and the VQ time with $1/p$ and fixed tree level L	35
4.1	Spatial relationship between different regions of human faces. This figure should be viewed in color.	40

4.2	Affinity matrix (computed on the Scene 15 problem [38]) of the original dataset; of the linearized dataset after applying HSM; and the eigenvalue normalized version after applying HSM, respectively. This figure should be viewed in color.	49
4.3	Three level spatial pyramid of an image on level 2, 1, and 0, respectively [78].	53
4.4	The relationship of a region (shown in dark red) to other spatial regions. The regions with light red color have close correlation with the dark red region according to the relationship matrix R learned from all training images. This figure should be viewed in color.	55
5.1	Distribution of Pearson's correlation coefficients between dimensions in Fisher vectors.	65
5.2	Comparison of dimensions' usefulness / importance values of different quantization methods on the Scene 15 dataset. This figure is best viewed in color.	69
5.3	Histogram of values (after power normalization) in one FV dimension in the Scene 15 dataset.	70
5.4	Classifier learning time using the proposed (MI) compressed features on VOC 2007 under different compression ratios.	76
5.5	Classification accuracy of FV on Scene 15 using three methods: MI (proposed), PQ, and BPBC.	79
5.6	Classifier training time of FV on Scene 15 using three methods: MI (proposed), PQ, and BPBC.	80
5.7	Classifier testing time of FV on Scene 15 using three methods: MI (proposed), PQ, and BPBC.	80
6.1	The contribution of this thesis.	86

List of Tables

3.1	VQ time T_i (seconds) and classification accuracy (%) of various methods. T_i/T_j is the speedup ratio between two methods. $i, j = 0, 1, 2$ (0-ExVQ, 1-FLANN, 2-Exhaustive), $K = 256$	31
3.2	VQ time T_i (seconds) and classification accuracy (%) of various methods. T_i/T_j is the speedup ratio between two methods. $i, j = 0, 1, 2$ (0-ExVQ, 1-FLANN, 2-Exhaustive), $K = 1024$	31
3.3	VQ error rates (%) of ExVQ and FLANN.	32
3.4	Comparison with other related methods on Caltech 101, $K = 1024$	35
4.1	Overview of HSM methods in the experiment.	51
4.2	Results on Scene 15: BOV with 1000 visual codewords.	54
4.3	Results on Indoor 67: BOV with 4000 visual codewords.	56
4.4	Results of SUN: BOV with 4000 visual codewords.	57
4.5	HSF Classification accuracy (%) on FV and VLAD.	58
5.1	Mean average precision (mAP) on VOC 2007. The loss of mAP relative to original dense feature (ratio 1) is also computed.	75
5.2	Top-5 accuracy on the ILSVRC 2010 dataset.	77
5.3	Top-1 accuracy on the SUN 397 dataset.	77
5.4	Image classification accuracy of SPM and HSM using different portion of selected real-value features from FVs of Scene 15.	81
5.5	Image classification accuracy of SPM and HSM using different portion of selected binary features from FVs of Scene 15.	81
5.6	Image classification accuracy of HSM using different portion of selected binary features from FVs of Indoor 67.	82

5.7	Image classification accuracy of HSM using different portion of selected binary features from FVs of SUN 397.	82
-----	---	----

Chapter 1

Introduction

In this dissertation, we investigate how to efficiently and accurately represent an image for image recognition. Image representation is fundamental to image recognition, since it largely determines the subsequent recognition quality.

A large number of works focus on proposing powerful image representations to improve the recognition accuracy [45, 11, 1, 78, 53, 74, 46, 5, 83, 75, 6, 19, 55, 32, 37]. Among these works, a few focus on devising fundamental local visual descriptors like SIFT [45], HoG [11], LBP [1] and CENTRIST [78]. For the image representation, different feature generation methods with increasingly high discriminance are proposed year by year. Early works like [74, 46, 5] match the local visual descriptors in images to compute their similarity in image recognition. Later, a large number of works focus on generating powerful feature vectors from local visual descriptors like bag-of-visual-words (BOV) model, sparse coding [83, 75, 19], Fisher Vector [55, 65], VLAD [32], etc. Combined with many powerful classifiers [14, 77], these feature vectors usually have better recognition performance than the previous descriptor matching methods. In recent years, deep learning networks [37, 22, 69, 10] become popular. They follow a different feature generation pipeline in that discriminative image representations are directly learned from raw pixels in images, leading to state-of-the-art performance.

In this dissertation, we focus on the the most widely used image representations,

where an image is represented by a feature vector which is generated using local visual descriptors. Besides, the spatial information in images is also important. In image recognition, a feature vector acquired from an image is usually not characteristic enough, which needs to include the spatial information in the image to achieve a stronger discriminance. Spatial information has also been thoroughly investigated for image recognition in [38, 83, 78, 33, 67]. The most representative method is the spatial pyramid matching (SPM) [38] technique, which is a de-facto standard setup in image representation with the SIFT descriptor [45] in most image recognition systems. It is a general framework which can be used with different feature generation methods and consistently improve their recognition accuracy. In SPM, all images are split into several regions according to the same grid. In an image, a feature vector is extracted from each region and all the feature vectors are concatenated into a long one to represent the image.

As shown in Fig. 1.1, an image recognition system includes the following modular. First, local visual descriptors such as SIFT are extracted from an image, which are then encoded into a feature vector like BOV, VLAD, Fisher Vector, etc. To encode the spatial information, using spatial pyramid matching (SPM), an image is usually split into several regions and a feature vector is extracted from each region. All the feature vectors are concatenated into a long one to represent this image, which is finally used to learn a classifier. In the testing phase, the learned classifier is applied on the feature vectors from testing images to get the decision score.

In this dissertation, we aim to improve the feature generation *speed*, *discriminance* and *storage*. We believe these three factors are critical for image representation.

- **Feature generation speed.** Few works specially focus on the feature generation speed. However, with the big trend that many large scale problems emerge in image recognition, the feature generation stage usually takes great amount of time. For example, ILSVRC 2010 dataset [4], which contains 1000 classes and more than 1

Figure 1.1: The flowchart of an image recognition system.

million images, needs weeks of time to compute Fisher Vectors. Thus, how to generate efficient features with less time cost becomes a meaningful problem.

Popular feature generation methods like BOV model, sparse coding, Fisher Vector and VLAD share some common processes. These feature generation processes can be roughly divided into the following steps: first, local visual descriptors like SIFT [45] or CENTRIST [78] are extracted from an image. Then, each visual descriptor is quantized by one or more visual codewords in a pre-learned visual codebook, which is named as vector quantization (VQ). VQ essentially is a nearest neighbor (NN) search process. Finally, different pooling methods like average pooling and max pooling are used for visual descriptors to get the final feature vector for each image.

In the feature generation process, VQ takes a large portion of time. Using the Caltech 101 [15] dataset as an example, in the BOV model, when we use densely sampled points and SIFT descriptors, the VQ time occupies from 87.53% to 96.56% of the total BOV generation time, when the visual codebook size varies from 256 to 1024.

To speed up the VQ (NN search) process, it is a natural choice to use approximate nearest neighbor (ANN) search methods. Most existing ANN methods like *kd*-tree follow a branch and bound strategy: split the search space into several regions according to certain criterion, and assume that a point and its nearest neighbor will fall into the same region. Thus, the algorithm only needs to examine the few points in one region. Another category of methods use the hashing technology [12, 30, 76, 27], where all points are assigned a binary string subject to a set of linear projects. The nearest neighbor is only searched within points having the same binary string which corresponds to a region specified by the linear projections. We call this kind of strategy the *inclusive* strategy. The inclusive assumption, however, rarely holds in reality. Thus, searching only one region leads to high VQ error rate. More regions (e.g., nearby ones) must also be examined, which increases the cost of ANN search. For example, FLANN [50] has a parameter that specifies how many regions are to be checked.

- **Feature discriminance.** Feature discriminance is the most important criteria to evaluate a feature representative or not. In this dissertation, rather than to propose a feature generation method, we propose a novel method to compute the spatial relationship of images, which can be used with most existing feature generation methods and improve their recognition performance beyond the state-of-the-art spatial pyramid matching method.

Spatial pyramid matching (SPM) technique [38] is the most widely used to compute the spatial relationships within images. In SPM, spatially corresponding regions are compared in different images. Later, many works [33, 68, 34] improve it following the same spatial constraint by using more spatial regions [33, 34] or assigning weights to different regions [68]. One drawback of these methods is they don't leverage the

relationship between different regions, which may provide additional discriminance for image recognition.

- **Feature storage.** In the past few years, many powerful, yet very high dimensional features like the Fisher Vector (FV) [55, 65], VLAD [32, 2], and super vector [86] have become popular. They have achieved state-of-the-art performance on many image classification tasks [57, 63, 35]. However, when these features are applied to large scale datasets, the rendezvous of very high dimensionality and huge number of examples poses serious challenges for both feature vector storage and subsequent classifier learning. For example, 310 gigabytes are required to store the Fisher Vectors for a subset of ImageNet images [65].

Feature compression has been proposed as a remedy for this problem, which compresses high dimensional feature vectors into manageable length. One of the most popular compression methods is Product Quantization (PQ) [29, 72, 52], which is originally proposed for fast nearest neighbor search [29]. When it is used in feature compression, it quantizes segments in feature vectors with pre-learned codebooks, and store the quantized results for compression. Some methods are hashing based, which transform a real-valued feature vector into a shorter binary string [63, 24, 23, 42]. Another category of methods uses dimension reduction techniques to transform long vectors into (real-valued) shorter ones like [66, 9]. A common theme of all these compression methods is that linear projection (*i.e.*, dot-product between the long feature vector or part of it with a learned vector) is the key; and, the effectiveness of these methods is determined by the quality of the learned linear projections.

To improve the three factors for image recognition, we propose three techniques respectively: exclusive vector quantization (ExVQ) to boost the speed, hyper-spatial matching (HSM) to increase the feature discriminance, and feature selection to reduce the storage of high-dimensional feature vectors:

- **ExVQ.** In Chapter 3, we propose ExVQ, an efficient ANN method, which greatly accelerates the VQ speed, without affecting the classification accuracy of generated features. In ExVQ, we use an *exclusive* strategy instead. All clusters of visual descriptors, not only the codewords, are divided into three regions, L , I , and R . I is located between L and R . Instead of insisting that nearest neighbors of visual descriptors from L must also reside inside L (i.e., inclusive), we only assume that such neighbors will not appear in R (and I is a buffer region which is ignored in this process). ExVQ thus does not search through codewords from clusters in R . In the experiment, we will show that ExVQ has much higher VQ speed than existing ANN methods, when they achieve the similar classification accuracy.
- **HSM.** In Chapter 4, we propose the hyper-spatial matching (HSM) as a novel method to compute the spatial relationship. Unlike existing spatial matching methods which only compute the relationship of spatially corresponding regions, HSM computes the spatial relationship between all region pairs in two images, thus offering more useful information and generating more discriminance than SPM. We also prove HSM is a Mercer kernel when certain conditions are satisfied. Furthermore, we propose two fast ways to learn HSM SVM classifiers, which are thousands of times faster than a general purpose SVM classifier with HSM kernel.
- **Feature selection for dimension reduction.** In Chapter 5, we propose to use feature selection to reduce the storage of high dimensional feature vectors. Unlike existing feature compression methods compressing all the features from original feature vectors, feature selection only stores those important features for image recognition. By discarding noisy features, the selected features can improve the recognition accuracy. In the experimental part, we show that feature selection achieves higher accuracy than that of feature compression methods, when the feature vectors use the same memory.

Experimental results of image recognition demonstrate that the proposed methods, not only individually but also as a system of combined methods (cf. Sec. 5.2 and Sec. 5.3.2), achieve faster speed and higher classification accuracy with lower memory cost than existing feature generation methods.

1.1 Relationship to previous work

The following papers describe part of the research presented in Chapter 3, Chapter 4, and Chapter 5, respectively.

Yu Zhang, Jianxin Wu, and Weiyao Lin, “Exclusive Visual Descriptor Quantization,” in *The 11th Asian Conference on Computer Vision*, 2012.

Yu Zhang, Jianxin Wu, Jianfei Cai, and Weiyao Lin, “Flexible Image Similarity Computation Using Hyper-Spatial Matching,” in *IEEE Trans. on Image Processing*, 23(9), 2014: pp. 4112-4125.

Yu Zhang, Jianxin Wu, and Jianfei Cai, “Compact Representation for Image Classification: To Choose or to Compress?” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2014.

Chapter 2

Literature Review

In this chapter, we review the state-of-the-art literatures on image representations, vector quantization, spatial matching, and feature compression, respectively.

2.1 Image representations

Computer vision researchers have put emphasis on the design of powerful feature generation methods.

SIFT [45] has been widely used to extract local features, which has almost been the dominant choice in scene classification and object recognition. SIFT is a 128 dimensional feature vector which is invariant to scale and robust to orientation changes. Following SIFT, CENTRIST was proposed for place recognition [79] and used in scene recognition [78]. In CENTRIST, the gray value of one pixel is compared with that of its surrounding eight pixels, which are represented with a 8-bit binary string. Each bit corresponds to one of the eight pixels: if its pixel value is smaller than that of the central pixel, it is 1. Otherwise it is 0. Each central pixel is encoded into a 8-bit binary string, which corresponds to a decimal number between 0 to 255. For the pixels in a local region of an image, CENTRIST counts a histogram of their encoded values, which is a 256 dimensional vector. Using SIFT or CENTRIST to extract visual descriptors from local patches, an image could be represented by a set (bag) of visual descriptors.

A class of methods [74, 46, 5] directly work on the the bag of descriptors for image recognition. The similarity of two images is evaluated by the similarity of visual descriptors in them, respectively. For example, in [5], if two descriptors are both the nearest neighbours of the same visual codeword, then their similarity is 1; otherwise it is 0. Finally, the similarity of two images is the sum of the similarity of all pairs of descriptors from them. These methods [74, 46, 5] usually are computational cost (descriptor match is exhaustive) and the recognition accuracy is lower than that of most feature vectors like bag-of-visual-words (BOV) model, VLAD, Fisher Vector (FV), etc.

Many feature vectors encode images with a visual codebook. The visual codewords in the codebook is usually computed from the cluster centers of visual descriptors using k -means method. Different feature generation methods compute different statistic information from visual descriptors in an image according to the visual codewords. BOV counts the histogram of visual descriptors quantized to different visual codewords. VLAD and Fisher vector will count more comprehensive information than BOV. VLAD counts the first order variation of visual descriptors to visual codewords and Fisher vector counts both the first and second order variation of visual descriptors to visual codewords. In the following, we will give detailed introduction of these feature generation methods.

BOV model has been popular in object recognition and scene recognition. It is a histogram feature vector. In the classical BOV model, the visual descriptors in an image are vector quantized with the most similar visual codeword in a visual codebook, which can be seen as a voting process. Each value in the BOV feature vector corresponds to a visual codeword and represents the frequency how many visual descriptors vote it.

Later, a soft assignment [43] of visual descriptors to visual codewords is used in BOV, which achieves better discriminance than that of the hard assignment. A visual descriptor is quantized by several most similar visual codewords. Each visual codeword is assigned a weight according to their similarity to the visual descriptor. The weights are

ℓ_1 normalized, so they denote the posteriori probability that the visual descriptor belongs to a visual codeword. When computing the BOV feature vector, instead of summing the weights of all visual descriptors, the max pooling is used, that is for a visual codeword (one bin in the feature vector), it only uses the maximum posteriori probability from all visual descriptors. They proved that the result of max-pooling is actually a lower bound of the probability of the presence of a visual word in an image.

Another type of soft assignment is sparse coding [83, 75, 19]. In sparse coding, each visual descriptor is assumed to arise from the linear combination of all the visual codewords. The weights are learned with the sparse constraint [83], where the nonzero values correspond to those important visual codewords that contribute to the visual descriptor. As the sparse coding problem is usually solved with ℓ_0 or ℓ_1 constraint, it is computationally costly. So different relaxations are proposed to solve it. For example, LLC [75] only uses a few locally proximal visual codewords to reconstruct a visual descriptor with ℓ_2 constrain on the reconstruction weights. A good property of LLC is one can obtain a closed form solution. In [19], the Laplacian sparse coding is proposed, where the relationship between visual descriptors are considered beside the sparse constraint. They assume that similar visual descriptors must generate similar sparse codes. This constraint greatly improves the quality of sparse codes, which achieves much better classification accuracy than that of [83, 75].

BOV methods are robust to the noise in the local descriptors, however they discard the variation in descriptors, which may contain discriminate information important for image recognition.

VLAD [32] and Fisher Vector (FV) [55] are proposed to include the descriptor variations to visual codewords. In VLAD, the descriptors are quantized by the visual codebook the same as the classical BOV at first. Then, instead of only using a descriptor as a vote in BOV, the variation of descriptors to visual codewords are used to generate VLAD

feature. The variations of visual descriptors to all visual codewords are concatenated altogether to form the VLAD feature vector. VLAD has shown good recognition accuracy in image retrieval [2] and action recognition [80]. Fisher Vector (FV) can offer a better representation than VLAD. In FV, the Gaussian mixture model is used instead of the visual codebooks, which is learned from the training descriptors. The FV of the set of visual descriptors X computes from the the gradient of the log-likelihood function G *w.r.t.* the Gaussian mixture model (GMM) parameters, which usually uses the mean and variance parts [63]:

$$\frac{\partial G}{\partial \boldsymbol{\mu}_j} = \frac{1}{n\sqrt{w_j}} \sum_{i=1}^n \gamma_j(i) \left(\frac{\mathbf{x}_i - \boldsymbol{\mu}_j}{\boldsymbol{\sigma}_j} \right), \quad (\text{Eq. 2.1})$$

$$\frac{\partial G}{\partial \boldsymbol{\sigma}_j} = \frac{1}{n\sqrt{2w_j}} \sum_{i=1}^n \gamma_j(i) \left[\frac{(\mathbf{x}_i - \boldsymbol{\mu}_j)^2}{\boldsymbol{\sigma}_j^2} - 1 \right], \quad (\text{Eq. 2.2})$$

where w_j , $\boldsymbol{\mu}_j$, and $\boldsymbol{\sigma}_j$ are the the mixture weight, mean vector and standard deviation vector of the j -th Gaussian, $\gamma_j(i)$ is the weight of the i -th visual descriptor to the j -th Gaussian. The results of all Gaussians are concatenated into a FV. VLAD and FV are usually very high-dimensional, which can lead to severe memory problem for large image datasets or rapid processing constrained to CPU cache memory.

In recent several years, deep learning networks [28, 3, 61] are becoming popular. The deep convolutional neural networks (CNN) [37] are used on image classification of ILSVRC 1000 dataset and achieve state-of-the-art accuracy. Different from traditional feature generation methods using local visual descriptors, CNN directly learn the feature from original raw pixels in images through multiple layers in the networks and finally learn the classifier (a prediction layer) altogether in a unified framework. However, CNN usually contain millions of parameters, which requires GPU to speed up the learning process. In order to quickly apply CNN on other problems, some works [22, 85] use the CNN learned from an auxiliary dataset like ILSVRC 2010 and then fine tune it on the specific problem, which can save great time and get a promising result.

As we have stated in Sec. 1, in this dissertation, we focus on the most widely used feature generation methods like BOV model, sparse coding, Fisher Vector and VLAD, which generate the feature vector from local visual descriptors.

2.2 Vector quantization

To speed up vector quantization (VQ) for the feature generation process, we investigate various approximate nearest neighbor (ANN) search methods. Vector quantization and its acceleration has long been studied in image and video coding. Tree-structure VQ methods such as TSVQ are popular because of their speed advantage [21]. Various branch-and-bound methods have also been used to find the exact or approximate NN with improved running time performance.

Tree structure is a useful tool to reduce the nearest neighbor search time. The hierarchical k -means tree can organize $O(n^m)$ items in a depth m tree, in which every internal tree node has n fan-out branches. A query then involves finding the path of the most similar nodes from the root of the tree toward a leaf node, which only requires nm distance computations. It has been successfully used in object retrieval from large object databases with tens of thousands of objects [51][59]. Randomized trees and extremely randomized clustering forests have also shown excellent speed and accuracy in matching features and detecting objects [39][49].

The branch-and-bound strategy, which usually utilizes linear projections to split the search space, is also proven valuable in ANN search. kd -tree splits the search space into halves using a hyperplane (which is a special case of linear projection) on the dimension with the largest variance and continues to split iteratively until each node contains only one point. Randomized kd -trees and ensembles of them have shown acceleration up to the 30x with approximately 90% correct nearest neighbors, comparing to the exhaustive nearest neighbor search of SIFT descriptors.

FLANN [50] is a well tailored ANN search library. It can perform fast ANN searches in high dimensional spaces. It contains a collection of algorithms, which includes popular ANN search methods like *kd*-tree and hierarchical *k*-means tree, and a system for automatically choosing the best algorithm and optimum parameters depending on the dataset.

Spill tree [44] is similar to the proposed ExVQ, in that they both adopt overlap (buffer) between the two split search sets. Spill tree splits the search set using a random projection. The overlap contains those search points within a small distance to the projection boundary. The difference is, ExVQ learns an accurate SVM classifier from the available visual descriptors and uses it to determine: which clusters, rather than data points, form the search sets or the overlap. Visual descriptors in the two split search sets, especially in the overlap, have zero (or almost zero) NN search error, which is not ensured in the spill tree. Similar strategy using a buffer to eliminate the decision error is also used in multi-class problem [48][20]. However, the method in [20] is very slow, which may take days to learn on the same dataset used in this research.

Another family of methods use multiple linear projections to map the feature vector (or visual descriptor) into a short string with few binary bits, e.g., locality sensitive hashing (LSH) [12], Hamming embedding [30], and spectral hashing [76]. In NN search, a query then only searches for a small portion of database points which have the same binary bits. A common issue of these methods is how to learn the linear projections to make the search space (database points) split evenly and accurately. This strategy is particularly attractive for processing very large datasets. Their storage requirements are greatly reduced, although their NN search accuracy may be significantly lower than ANN methods such as FLANN.

In recent years, sparse coding [75][83] is popular in object recognition and classification, such as Locality-constrained Linear Coding (LLC) [75]. Multiple database points

are assigned weights under the sparse constraint. In LLC, the complex sparse constraint learning problem is relaxed by finding the linear combination of a few visual codewords which are most close to the visual descriptor.

Recently, product quantization [29] is also used for NN search. It can save the search time and storage by quantizing the sub-vectors of the original high-dimension feature vector, which is, however, more suitable for retrieval applications rather than for visual recognition. Its details will be introduced in Sec. 2.4 in this chapter.

2.3 Spatial matching

Spatial information and in particular spatial arrangement of features is important for image recognition. Spatial matching techniques can consistently improve most feature generation methods. In this section, various spatial matching techniques are investigated.

Spatial pyramid matching (SPM) [38] is one of the most important techniques to compute spatial information. In SPM, each image is divided into several regions according to the same fixed spatial grid. Each region is represented by a feature vector. The feature vectors from all regions are concatenated into a long one to represent an image.

A saliency map is proposed in [68] for each region in an image to highlight the spatial difference. An image is first split into several regions the same as SPM and represented with feature vectors. Then, each region is assigned a weight. The feature vectors of regions are used to learn a max-margin SVM classifier for images. During the classifier learning process, the region weights of each image and the SVM classifier are iteratively updated to get the optimal solution. When computing image similarity, the role of important regions in an image can be highlighted with large values of saliency maps.

Receptive field is proposed to improve SPM in [33]. Instead of using the simple spatial structure in SPM, receptive field generates many spatial regions through different combinations of basic spatial cells. When features are extracted from the receptive fields

and make a high-dimensional one for one image, a sparse SVM classifier is learned to choose important features from them for image classification. Similar to [33], by randomly generating multiple different spatial regions, [34] also achieved better results than SPM. For image classification, two strategies are proposed to learn a classifier: one is to select the best spatial patterns and use these features to learn a classifier for each class. The other is to use boosting technique to learn classifiers on all the features.

A different way to encode spatial information is used in [64, 36], which append the coordinates to the local descriptors like SIFT. In [64], the coordinates of SIFT features are augmented to them, which are used to compute Fisher Vector. Without using SPM, the proposed method achieved comparative accuracy to SPM in image classification. [36] used both augmented SIFT (with coordinates) and SPM, which always achieved better classification results than SPM.

More works focus on the feature representation rather than the improvement of spatial structures, e.g., [6, 83, 75, 58, 78]. All these works propose powerful features and apply them with the classical spatial pyramid structure [38], which achieve much better classification accuracy than [38].

However, since SPM only computes the similarity between spatially corresponding regions, if similar objects appear in different regions, there exists possible mismatch in SPM, which has been observed in [62]. An object-centric spatial pooling (OCP) approach is proposed, when the location of the object of interest is detected, which is useful for the classification. In the OCP method, all images are not split following the same spatial pyramid. Instead, images are split into the foreground and background, which is done through a sliding window manner. The object in a current window is assumed as the foreground object and the rest is as the background. Then, OCP looks for the best matching foreground in two images when computing image similarity. They learn a SVM classifier based on this criteria for image classification. OCP method is suitable for images

which have a central object. For scene images which may have multiple objects, it may lead to inferior results.

2.4 Feature compression and feature selection

To reduce the storage of feature vectors, we first introduce the details of some popular feature compression methods. Then, we introduce some related feature selection methods.

2.4.1 Feature compression

Product quantization (PQ) [29] is one of the most popular feature compression methods. For a set of high dimensional feature vectors with dimension D , PQ divides each feature vector into $G = \frac{D}{d}$ segments, where d is the segment length. For each segment, PQ learns a codebook with K words using the k -means clustering method. K is usually a power of 2, and one segment is then quantized by the index to its nearest code word. That is [29]:

$$\underbrace{x_1, \dots, x_d}_{u_1(x)}, \dots, \underbrace{x_{D-d+1}, \dots, x_D}_{u_G(x)} \rightarrow q_1(u_1(x)), \dots, q_G(u_G(x)), \quad (\text{Eq. 2.3})$$

where q_i is the quantizer for the i -th segment. PQ needs to store both the indices (*i.e.*, compressed features) and the G codebooks. Assuming single precision floating numbers (4 bytes) are used, the codebooks in total require $4GKd = 4KD$ bytes to store, and the D dimensional feature vector is compressed from $4D$ bytes to $\frac{G \log_2 K}{8}$ bytes, with a compression ratio $\frac{32D}{G \log_2 K} = \frac{32d}{\log_2 K}$.

In learning and applying a classifier, the compressed features must be decoded using the real-valued code word associated with the stored indices. Thus, PQ compressed feature has the same training and prediction complexity as the original uncompressed feature. [72] proposes an acceleration technique for SVM training on PQ compressed features. This technique (without special optimization) was shown slower than a plain SVM with optimized BLAS library [72]. Thus, we do not use it in our PQ experiments.

[31] recommends randomly rotating the original vector before applying PQ to balance the variance in different dimensions in order to reduce the PQ quantization error. Many other techniques usually have this rotation step, *e.g.*, ITQ [24], face verification [9], etc. In [31], a rotation matrix is applied to the reduced high dimensional feature vector after PCA. In ITQ [24], a binary string is learned for each feature vector. Original real value feature vectors are first dimension reduced by PCA. Then, a rotation matrix is learned for the reduced feature vectors so that the loss between rotated reduced feature vectors and the binary strings (to be acquired) is minimized. That is:

$$\min_{B,R} \|B - R^T V\|_F^2, \quad (\text{Eq. 2.4})$$

where R is the rotation matrix, B is the binary codes for all instances, V is the feature vectors of all instances after PCA, each column in B and V corresponds to an instance. In this optimization, V is pre-acquired. B and R are learned iteratively in the optimization.

Similarly, [9] learned a rotation matrix and additional linear projections for feature vectors in face verification. That is:

$$\min_{R,P} \|R^T V - P^T X\|_F^2 + \lambda \|P\|_1, \quad (\text{Eq. 2.5})$$

where X is the original feature vectors, P is the linear projection to be learned, R and V are defined same as ITQ. P is a sparse matrix which is used to approximate V with consideration of the rotation R . This optimization is also learned for R and P iteratively.

All the three methods [31, 24, 9] are not suitable for high-dimensional feature vectors, because PCA will not be feasible for high dimensional feature vectors. The storage of PCA projection matrix and rotation matrix will be too large. BPBC [23] is a remedy. It uses bilinear instead of linear projections to reduce the feature dimensionality. In BPBC, each high dimensional vector $\mathbf{x} \in \mathbb{R}^D$ is reshaped into a matrix (or 2nd order tensor) $\mathbf{x}' \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2}$, $D = d_1 d_2$. Then, two orthogonal projection matrices $R_1 \in \mathbb{R}^{d_1} \times \mathbb{R}^{c_1}$ and $R_2 \in \mathbb{R}^{d_2} \times \mathbb{R}^{c_2}$ can be learned or randomly generated. Finally, a vectorized form of $\text{sgn}(R_1^T \mathbf{x}' R_2)$ is the compressed binary features, which requires $\frac{c_1 c_2}{8}$ bytes.

2.4.2 Feature selection

Feature selection has been investigated for many years in machine learning and computer vision areas. It is used to select important features from high dimensional feature vectors.

An SVM classifier learned with ℓ_1 norm regularization [18] is a classical feature selection tool. In binary classification, given the feature vectors and their labels $(\mathbf{x}_i, y_i), y_i \in \{-1, 1\}, i = 1, \dots, N$, a linear SVM classifier is learned as:

$$\min_{\mathbf{w}} \|\mathbf{w}\|_{\ell_1} + C \sum_{i=1}^N \xi(\mathbf{w}; \mathbf{x}_i, y_i), \quad (\text{Eq. 2.6})$$

where $\mathbf{w}$ is the decision hyperplane of the linear classifier, $C > 0$ is a penalty parameter, and $\xi(\mathbf{w}; \mathbf{x}_i, y_i)$ is a loss function. After learning the SVM classifier, the nonzero values in $\mathbf{w}$ correspond to those important dimensions in feature vectors. The number of non-zero elements (sparsity) in $\mathbf{w}$ can be tuned by C .

A feature generation machine (FGM) [70] is proposed for feature selection. Different from SVM classifier with ℓ_1 norm regularization, FGM uses an additional variable $\mathbf{d}$ to control the sparsity of SVM decision hyperplane, that is:

$$\begin{aligned} \min_{\mathbf{d}} \min_{\mathbf{w}, \xi_i, \rho} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 + \frac{C}{2} \sum_{i=1}^N \xi_i^2 - \rho \\ \text{s.t.} \quad & y_i \mathbf{w}^T (\mathbf{x}_i \odot \mathbf{d}) \geq \rho - \xi_i, \quad i = 1, \dots, N. \\ & \sum_{i=1}^m d_i \leq B, d_i \in \{0, 1\} \end{aligned} \quad (\text{Eq. 2.7})$$

where $\mathbf{x}_i \odot \mathbf{d}$ is the element-wise product between two vectors, which finally outputs a vector; B is a scalar, which controls the sparsity. A good property of this optimization is it can be finally relaxed into a standard Multiple Kernel Learning (MKL) problem, which is convex and can be solved efficiently with existed solvers. After learning FGM, those nonzero values in $\mathbf{d}$ denote selected features.

Mutual information is also used to select important features in image classification [54]. A minimal-redundancy-maximal-relevance criterion (mRMR) is proposed for

feature selection. In practice, in order to select a feature set S_m with m features from the whole feature set S , an incremental selection method is used. That is each time, a new feature is selected with respect to the following criteria:

$$\max_{j \in S - S_{m-1}} \left[I(\mathbf{x}_{:j}; c) - \frac{1}{m-1} \sum_{i \in S_{m-1}} I(\mathbf{x}_{:j}; \mathbf{x}_{:i}) \right], \quad (\text{Eq. 2.8})$$

where $\mathbf{x}_{:i}$ is the i -th feature in all feature vectors, $I(\mathbf{x}_{:j}; c)$ is the mutual information of j -th feature and the class c , and $I(\mathbf{x}_{:j}; \mathbf{x}_{:i})$ is the mutual information between two features, S_{m-1} is the selected feature set after $(m-1)$ -th feature is selected. Mutual information denotes the dependence or correlation between two variables. Eq. 2.8 shows that each selected feature must have the maximum relevance *w.r.t.* the class label, meanwhile it must have the minimum redundancy *w.r.t.* already selected features, which is the mRMR criteria.

[16] selected a new feature based on its mutual information *w.r.t.* the class labels and already selected features. Each feature is selected iteratively as the following the steps:

$$v(1) = \max_i I(Y; \mathbf{x}_{:i}) \quad (\text{Eq. 2.9})$$

$$\forall k, 1 \leq k < m, v(k+1) = \max_i \left\{ \min_{l \leq k} I(Y; \mathbf{x}_{:i} | \mathbf{x}_{:v(l)}) \right\}, \quad (\text{Eq. 2.10})$$

where $v(l)$ is the selected feature at the l -th time, Y is the label set, all other variables have the same meaning as Eq. 2.8.

However, all the above mentioned feature selection methods are too expensive to apply on large scale image recognition tasks. In this dissertation, we will propose an efficient and effective feature selection method for high-dimensional feature vectors.

Chapter 3

Exclusive Vector Quantization (ExVQ) for Fast Feature Generation

We believe that the vector quantization (VQ) is the most time consuming step in many widely used feature generation methods like bag-of-visual-words (BOV) model, sparse coding, Fisher Vector (FV), VLAD, etc. Thus, improving the VQ step leads to the speedup of the feature generation process. The VQ step essentially is a nearest neighbor (NN) search process. Two things are considered in the VQ step:

1. **VQ error rate vs. VQ speed.** Generally, the VQ error rate will increase when the VQ speed increases. However, as we focus on the image recognition, we can tolerate some VQ errors when the VQ speed increases.
2. **VQ error rate vs. recognition accuracy.** The VQ error should not influence the recognition accuracy. The properties of different types of VQ errors will be investigated.

We propose a new exclusive vector quantization (ExVQ) method for fast approximate nearest neighbor (ANN) search. It is different from many existing ANN methods, which use an inclusive strategy to search nearest neighbors.

Representatively, we focus on the VQ process in the classical BOV model, which performs a nearest neighbor search. In the BOV model, given a set of visual descriptors $X = \{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{N-1}\}$, by applying the k -means clustering, the cluster centers are used

as a codebook $\{\mathbf{c}_0, \mathbf{c}_1, \dots, \mathbf{c}_{K-1}\}$, where $\mathbf{x}_i, \mathbf{c}_j \in \mathbb{R}^d, i = 0, 1, \dots, N-1, j = 0, 1, \dots, K-1$.

A visual descriptor $\mathbf{x} \in \mathbb{R}^d$ is quantized by the index of its nearest codeword as:

$$nn(\mathbf{x}) = \arg \min_{0 \leq i \leq K-1} dist(\mathbf{x}, \mathbf{c}_i), \quad (\text{Eq. 3.1})$$

where $dist(\cdot, \cdot)$ can be any distance metric between two vectors. Euclidean distance is used here, that is, $dist(\mathbf{x}, \mathbf{c}_i) = \|\mathbf{x} - \mathbf{c}_i\|_{\ell_2}$.

3.1 Effect of ANN VQ errors to classification

In an ANN method, we define the searched nearest neighbor index for a point $\mathbf{x}$ as $ann(\mathbf{x})$. During an ANN based VQ process, a VQ error happens when $ann(\mathbf{x})$ is different from $nn(\mathbf{x})$. Intuitively, a high VQ error rate will lead to more classification errors in the subsequent visual classification stage. However, we find empirically that this is not the case. When certain ANN methods are used, a high VQ error rate (e.g., 30%) does not necessarily hurt the classification accuracy. Thus, we have the freedom to allow some VQ errors in order to accelerate the VQ process.

Fig. 3.1 shows one example of how the classification accuracy varies with the VQ error rate of FLANN. Although VQ errors vary from 5.70% to 32.44%, the classification accuracies are almost the same. This result seems counter intuitive. We conjecture that *it can be explained by the properties of the VQ errors*: some VQ errors are benign and will have minimal effect on the classification performance.

To validate this conjecture, we first define the error rank of $ann(\mathbf{x})$ using the following procedure. We calculate the distance of $\mathbf{x}$ with each codeword $dist(\mathbf{c}_i, \mathbf{x}), i = 0, 1, \dots, K-1$, and store them in an array in the ascending order. The error rank of $ann(\mathbf{x})$ is defined as the index of $dist(\mathbf{c}_{ann(\mathbf{x})}, \mathbf{x})$ in this sorted array. The error rank is 0 when $ann(\mathbf{x}) = nn(\mathbf{x})$. Then we define the benign VQ error: a small portion of VQs ($< 30\%$

Figure 3.1: VQ error rate and classification accuracy (one round) using FLANN hierarchical k -means tree, on Caltech 101, when $K = 256$.

or around)¹ has error and the error ranks are small. This definition provides us with an empirical way to evaluate the property of VQ errors. In Fig. 3.2, we show the histograms of ANN error ranks using FLANN with different check numbers, where the vertical axis is shown in natural logarithm. The error ranks of most VQ errors are close to 1, which supports our conjecture on benign VQ errors. More systematic results will be presented in the experimental section.

3.2 ANN search: inclusion vs. exclusion

When inclusive methods (e.g., LSH and FLANN) search the nearest neighbor of a visual descriptor in the codebook $\{\mathbf{c}_0, \mathbf{c}_1, \dots, \mathbf{c}_{K-1}\}$, they use linear projections to reduce the search range. With a linear projection $(\mathbf{w}, b)$, $\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$, the inclusion strategy is defined as:

$$\forall \mathbf{x} \in \mathbb{R}^d, \text{sgn}(\mathbf{w}^T \mathbf{c}_{nn(\mathbf{x})} + b) = \text{sgn}(\mathbf{w}^T \mathbf{x} + b), \quad (\text{Eq. 3.2})$$

¹We find 30% is valid for most cases. More results will be shown in the experiment.

Figure 3.2: The histograms of ANN error ranks. The vertical axis is the frequency numbers shown *in natural logarithm*. We use FLANN hierarchical k -means tree with different check numbers on Caltech 101 when $K = 256$.

where $\text{sgn}(\cdot)$ is the sign function. This implies that the boundary of the projection coincides with the Voronoi boundary.

However, in most cases, inclusive search methods fail to find the nearest neighbor in the reduced search range. Fig. 3.3 illustrates the inclusive NN search in the 2-D space. We use k -means to get 15 clusters from the randomly generated points. Each black star is the center of a cluster (i.e., codeword), which altogether form the codebook. Subject to the linear projection (the black line), 40% of the clusters (which are crossed by the line) violate the inclusive assumption. In reality, the inclusive assumption is violated even more seriously. For example, using a random linear projection on Caltech 101 ($K = 256$, $d = 128$), only about 10% clusters can guarantee Eq. 3.2. Thus the inclusive search methods usually need to check NN in multiple adjacent regions to maintain the search accuracy, which slows down the search speed.

We propose an exclusive NN search strategy to fulfill three goals: to reduce NN

Figure 3.3: Illustration of inclusive vs. exclusive NN search. The random projection violates the inclusive assumption. The exclusive strategy, however, can correctly exclude the red or the green clusters. Please refer to the second paragraph in Sec. 3.2 to understand what the color means. This figure should be viewed in color.

search time, to maintain a reasonable VQ error rate, and to ensure that the VQ errors are benign. In this strategy, the search range is reduced by excluding a portion of clusters (rather than data points) from the whole search space subject to a linear projection. The exclusive NN search involves three index sets: the available search set C containing the indexes of all clusters to be searched, the positive and negative exclusion sets $C_{+1}(L)$ and $C_{-1}(R)$, which are the index sets of clusters that can be excluded.

We illustrate the idea of exclusive NN search in Fig. 3.3. We first assign the indexes of red and green clusters to C_{+1} and C_{-1} , respectively. Then, for a point above the linear projection, we search NN from the codewords within the red and blue clusters. The green clusters (C_{-1}) can be safely excluded. In the inclusive strategy, for a point in

the upper right part above the linear projection, its NN can be one of the blue-colored centers. Thus, in order to get accurate NN, regions in both sides of the black line are to be searched, that is, no cluster can be safely ignored.

In real-world NN search problems, a random linear projection may fail to produce high quality exclusion sets (*i.e.* the two exclusion sets cannot be split with max margin along the random projection). Instead, we utilize existing examples to evaluate the irregular boundaries of exclusion clusters in C_{+1} and C_{-1} . We use visual descriptors in all exclusion clusters to train an accurate linear classifier such that a high quality linear projection direction can be learned. This linear projection can accurately split the exclusion clusters on two sides.

Specifically, first, we calculate the inner product of the codeword in each cluster $\mathbf{c}_i, i \in C = \{0, 1, \dots, K-1\}$ with a random vector $\mathbf{r} \in \mathbb{R}^d$ and sort the values in the ascending order. Then, we choose the desired exclusion portion $0 < p < 1/2$ (that is the portion of points to be excluded on one side of the linear projection) and assign the indexes of clusters which codewords are of the pK largest (smallest) values of $\mathbf{r}^T \mathbf{c}_i$ to C_{+1} (C_{-1}). Next, we generate the positive training set $X_{+1} = \{\mathbf{x} | \mathbf{x} \in X, nn(\mathbf{x}) \in C_{+1}\}$ and the negative training set $X_{-1} = \{\mathbf{x} | \mathbf{x} \in X, nn(\mathbf{x}) \in C_{-1}\}$. Finally, we train a linear SVM classifier $(\mathbf{w}, b)$ using X_{+1} and X_{-1} :

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 + \alpha \sum_{\mathbf{x} \in X_{+1}} \max(0, 1 - (\mathbf{w}^T \mathbf{x} + b))^2 \\ & + \alpha \sum_{\mathbf{x} \in X_{-1}} \max(0, 1 + (\mathbf{w}^T \mathbf{x} + b))^2, \end{aligned} \quad (\text{Eq. 3.3})$$

where α is the trade off parameter.

After the linear SVM classifier is trained, we obtain $(\mathbf{w}, b)$, C , C_{+1} and C_{-1} . For a visual descriptor $\mathbf{x} \in \mathbb{R}^d$, the exclusive NN search is defined as:

$$ann(\mathbf{x}) = \arg \min_i \|\mathbf{x} - \mathbf{c}_i\|_{\ell_2}, \quad (\text{Eq. 3.4})$$

$$\text{s.t. } i \in C - C_{-\text{sgn}(\mathbf{w}^T \mathbf{x} + b)}, \quad (\text{Eq. 3.5})$$

where $C - C_{-\text{sgn}(\mathbf{w}^T \mathbf{x} + b)}$ is named as the active search set of $\mathbf{x}$ (in Fig. 3.3, it is the set of blue points and red points or the set of blue points and green points).

ExVQ and spill tree operate on different principles. ExVQ excludes one cluster entirely, which includes all points within the corresponding Voronoi cell. Thus, the nearest codewords of points in the rest search space never or rarely (depending on the training quality of the linear SVM classifier) exist in the exclusion set. In the spill tree, although a centroid may be far away from the exclusion region, its corresponding cluster (i.e., Voronoi cell) may have large overlap with it, that is, causing many VQ errors.

Similar exclusion strategy has been used in multi-class image classification problems [20, 82]. [20] learned a tree structure for multi-class classification. Each node in the tree splits the classes into three groups like ExVQ and learns a max-margin classifier for two groups (the third group plays the role of a buffer). [82] is a little different from ExVQ and [20]. It also uses a tree structure for efficient multi-class classification. But in each node of the tree, it splits classes into two groups and learns a max-margin classifier. A good merit is the problem in [82] can be solved efficiently with existing MKL solver after relaxation. However, both these two methods are proposed for multi-classification problem, which deal with hundreds of classes. They will be very time consuming on vector quantization, which usually involves thousands of database points.

3.3 Tree-structured ExVQ

One single exclusion is not enough in real-world applications. In ExVQ, we use multiple linear classifiers with a hierarchical tree structure. We implement ExVQ in an L -level complete binary tree, where each node is indexed level by level from 0 to $2^L - 2$. The i th node is related to three cluster index sets: C^i , C_{+1}^i and C_{-1}^i , which are the available search set, and the positive and negative exclusion sets, respectively. First, in the root node, we set $C^0 = \{0, 1, \dots, K - 1\}$. Then, in the i th node, C_{+1}^i and C_{-1}^i are chosen

Input : Descriptors $\{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{N-1}\} \subseteq \mathbb{R}^d$ and K centers $\{\mathbf{c}_0, \mathbf{c}_1, \dots, \mathbf{c}_{K-1}\}$, ExVQ tree level L and an exclusion portion parameter p .	
1	$C^0 \leftarrow \{0, 1, \dots, K-1\}$
2	for $id = 0$ to $2^L - 2$ do
3	$C_{+1}^{id} \leftarrow \emptyset, C_{-1}^{id} \leftarrow \emptyset, parent \leftarrow 0$
4	if $id > 0$ then
5	$parent = \lfloor \frac{id-1}{2} \rfloor$
6	if id <i>is odd</i> then
7	$C^{id} \leftarrow C^{parent} - C_{-1}^{parent}$
8	else
9	$C^{id} \leftarrow C^{parent} - C_{+1}^{parent}$
10	$a \leftarrow \emptyset, X_{+1} \leftarrow \emptyset, X_{-1} \leftarrow \emptyset$
11	generate a random vector $\mathbf{r} \in \mathbb{R}^d$
12	$a \leftarrow a \cup \mathbf{r}^T \mathbf{c}_i, \forall i \in C^{id}$
13	for i <i>in</i> C^{id} do
14	if $\mathbf{r}^T \mathbf{c}_i$ <i>in the</i> $p \cdot C^{id} $ <i>biggest values of</i> a then
15	$C_{+1}^{id} \leftarrow C_{+1}^{id} \cup i, X_{+1} \leftarrow X_{+1} \cup \mathbf{x}_j, \forall j, nn(\mathbf{x}_j) == i$
16	if $\mathbf{r}^T \mathbf{c}_i$ <i>in the</i> $p \cdot C^{id} $ <i>smallest values of</i> a then
17	$C_{-1}^{id} \leftarrow C_{-1}^{id} \cup i, X_{-1} \leftarrow X_{-1} \cup \mathbf{x}_j, \forall j, nn(\mathbf{x}_j) == i$
18	train a linear projection $(\mathbf{w}_{id}, b_{id})$ with positive set X_{+1} and negative set X_{-1}
Output: $\{(\mathbf{w}_i, b_i), C^i, C_{+1}^i, C_{-1}^i\}, i = 0, 1, \dots, 2^L - 2$.	

Algorithm 1: Build tree-structure ExVQ

from C^i by a random vector $\mathbf{r}_i \in \mathbb{R}^d$ as described in Sec. 3.2, and to train a linear SVM classifier $(\mathbf{w}_i, b_i)$ according to Eq. 3.3. Next, we compute the available search sets for the the left and right children of the i th node as $C^{2i+1} = C^i - C_{-1}^i$ and $C^{2i+2} = C^i - C_{+1}^i$, respectively. Finally, the whole tree can be built recursively from the root. This process is detailed in Algorithm 1.

After the tree is built, we get $\{(\mathbf{w}_i, b_i), C^i, C_{+1}^i, C_{-1}^i\}$ in every node. For a query $\mathbf{x} \in \mathbb{R}^d$, the NN search is processed as follows: first, starting from the root node, we compute $\mathbf{w}_i^T \mathbf{x} + b_i$. If the value is positive, the exclusion process goes to process its left child, otherwise to the right. This process continues iteratively until a leaf node is reached. Then we output the active search set of the leaf node. Finally, the NN of $\mathbf{x}$ is

Figure 3.4: Tree-structured ExVQ: for a given query descriptor, in each node, the red or green part is excluded depending on outputs of the associated linear classifiers. This figure should be viewed in color

```

Input : A visual descriptor  $\mathbf{q} \in \mathbb{R}^d$ ,  $\{(\mathbf{w}_i, b_i), C^i, C_{+1}^i, C_{-1}^i\}, i = 0, 1, \dots, 2^L - 2$ .
1  $id \leftarrow 0$ 
2 while  $id < 2^L - 1$  do
3   if  $\mathbf{w}_{id}^T \mathbf{q} + b_{id} > 0$  then
4      $id \leftarrow id \times 2 + 1$ 
5   else
6      $id \leftarrow id \times 2 + 2$ 
Output:  $leaf = \lfloor \frac{id-1}{2} \rfloor$ 
   if  $id$  is odd then
      $ann(\mathbf{q}) = \arg \min_{i \in (C^{leaf} - C_{-1}^{leaf})} \|\mathbf{q} - \mathbf{c}_i\|_{\ell_2}$ 
   else
      $ann(\mathbf{q}) = \arg \min_{i \in (C^{leaf} - C_{+1}^{leaf})} \|\mathbf{q} - \mathbf{c}_i\|_{\ell_2}$ 

```

Algorithm 2: NN search using ExVQ

acquired by exhaustively searching the active search set. We fully specify ANN search by ExVQ in Algorithm 2.

In ExVQ, the tree level L and the exclusion portion p can influence the search speed and the VQ error rate. Applying ExVQ to a visual descriptor, the number of distance computations in the NN search is: $\hat{K} = L + K(1 - p)^L$.

If we assume that the VQ error rates caused by the exclusions from the same level

nodes in the tree are the same, the whole VQ error rate of ExVQ can be estimated by:

$$e_{vq} = 1 - \prod_{l=1}^L (1 - pe_l), \quad (\text{Eq. 3.6})$$

where e_l is the VQ error rate caused by the exclusion in the l -th level. We note that e_l is the generalization error of the linear classifiers in level l . Take Caltech 101 as an example, when $K = 256$, $L = 10$, $p = 1/5$, we get $\hat{K}/K = 14.64\%$. If we use the maximum training error of all linear SVM classifiers as an estimation for e_l , we estimate the VQ error as $e_{vq} = 14.45\%$ using Eq. 3.6, which provides a reasonable estimation of the actual VQ error rate (which is 11.13%).

3.4 Experiments for ExVQ

We evaluate image classification on 4 datasets: Caltech 101 (15 + 20) [15], scene 15 (100 + rest) [38], sports 8 (70 + 60) [40], and indoor 67 (80 + 20) [60]. X+Y for each dataset means the number of training and testing images used in every category, respectively. Following the suggestion from [78], we use SIFT ($d = 128$) for Caltech 101, and the CENTRIST ($d = 256$) descriptor for other datasets. We use the LIBLINEAR package [14] to train a linear SVM classifier in each node of the ExVQ tree. The trade off parameter in Eq. 3.3 is $\alpha = 0.01$. After the BOV representation of all images are created, we use a one-vs-all SVM classifier with the χ^2 kernel $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \sum_{i=1}^d \frac{2x_{1i}x_{2i}}{x_{1i}+x_{2i}}$ [73, 77] for classification. For each dataset, five training / testing sets are randomly split and average accuracy and VQ time of these five rounds are reported, if not otherwise specified.

3.4.1 VQ time vs. classification accuracy

First, we compare the VQ time and classification accuracy of ExVQ, FLANN and the exhaustive search, with different codebook size K . FLANN is a state-of-the-art ANN search method [50]. For ExVQ, we fix the exclusion portion $p = 1/5$ and fix the tree

Table 3.1: VQ time T_i (seconds) and classification accuracy (%) of various methods. T_i/T_j is the speedup ratio between two methods. $i, j = 0, 1, 2$ (0-ExVQ, 1-FLANN, 2-Exhaustive), $K = 256$.

Dataset	ExVQ			FLANN		Exhaustive	
	T_2/T_0	T_1/T_0	Accuracy	T_2/T_1	Accuracy	T_2	Accuracy
Caltech 101	4.86	2.83	61.87 $\pm$ 0.25	1.72	62.22 $\pm$ 0.72	627.8	62.63 $\pm$ 0.55
Scene 15	5.77	2.22	81.16 $\pm$ 0.45	2.60	81.17 $\pm$ 0.14	2070.8	81.36 $\pm$ 0.42
Sports 8	5.93	2.29	82.50 $\pm$ 0.90	2.59	82.71 $\pm$ 1.07	1736.2	82.33 $\pm$ 1.28
Indoor 67	5.78	2.53	35.75 $\pm$ 1.09	2.28	35.04 $\pm$ 1.28	2724.6	34.76 $\pm$ 1.39

Table 3.2: VQ time T_i (seconds) and classification accuracy (%) of various methods. T_i/T_j is the speedup ratio between two methods. $i, j = 0, 1, 2$ (0-ExVQ, 1-FLANN, 2-Exhaustive), $K = 1024$.

Dataset	ExVQ			FLANN		Exhaustive	
	T_2/T_0	T_1/T_0	Accuracy	T_2/T_1	Accuracy	T_2	Accuracy
Caltech 101	8.92	1.90	65.08 $\pm$ 0.50	4.69	65.53 $\pm$ 0.62	2472.0	65.79 $\pm$ 0.53
Scene 15	12.36	2.41	83.17 $\pm$ 0.32	5.13	83.27 $\pm$ 0.33	8162.4	83.53 $\pm$ 0.20
Sports 8	12.39	2.92	83.42 $\pm$ 0.77	4.25	83.67 $\pm$ 1.22	6892.4	84.17 $\pm$ 1.11
Indoor 67	12.22	2.83	38.52 $\pm$ 1.96	4.32	38.91 $\pm$ 1.30	10818.6	38.94 $\pm$ 1.33

level $L = 10$ for $K = 256$ and $L = 15$ for $K = 1024$. The results are shown in Table 3.1 ($K = 256$) and Table 3.2 ($K = 1024$).

ExVQ is much faster than FLANN while their classification accuracies are almost the same. In Table 3.1 and Table 3.2, the classification accuracy of ExVQ and FLANN on all four datasets have no statistically significant difference. However, ExVQ is about two times faster than FLANN no matter the codebook with a small size ($K = 256$) or a large size ($K = 1024$). Both ExVQ and FLANN are much faster than the exhaustive search. ExVQ speed up NN search 5–10 times than the exhaustive search. Especially, in the indoor 67 dataset, when $K = 256$, ExVQ’s classification accuracy is even significantly better than the exhaustive search.

When the codebook size is larger, the NN search speedup of ExVQ is more prominent. The search speedup of $K = 1024$ is about twice of that when $K = 256$, which is similar for FLANN. However, the larger the codebook size is, the more tree levels ExVQ will need to achieve the speedup. This requirement usually costs more time in building the

Table 3.3: VQ error rates (%) of ExVQ and FLANN.

Dataset	$K = 256$		$K = 1024$	
	ExVQ	FLANN	ExVQ	FLANN
Caltech 101	11.13	4.91	14.15	19.32
Scene 15	4.99	3.20	18.64	7.14
Sports 8	5.82	4.23	19.19	6.63
Indoor 67	6.03	5.81	16.90	6.26

tree, mainly in training the linear SVM classifiers. When $K = 256$, building the 10-level tree costs 18.8–56.6 seconds on different datasets. The building time increases to 326–617 seconds using 15 levels when $K = 1024$. In the experiments, we keep the number of training descriptors from 40468 to 76219, to restrict the training time of linear SVM classifiers. Note that the classifiers are trained offline, and are trained for only once.

We also test a smaller check number for FLANN. When the check number is 4, FLANN is faster than default check number 32, but its VQ time is still at least 67% longer than ExVQ (or higher than 67% on different datasets). Furthermore, when $K = 1024$, the classification accuracy of FLANN with check number 4 is statistically significantly lower than that of ExVQ.

3.4.2 Properties of VQ errors

For these datasets, the VQ error rates of ExVQ and FLANN are shown in Table 3.3. In some cases, the VQ error rates of ExVQ and FLANN are close. In many cases, the VQ error rate of ExVQ is higher than that of FLANN. However, as observed from Table 3.1 and Table 3.2, the difference in VQ error rates does not have impact on the classification performance. In fact, if we increase L and reduce p , ExVQ can easily achieve both faster VQ speed and smaller VQ error rate than FLANN, with the cost of larger storage.

In practice, we do not need very large L , because ExVQ has benign VQ errors. We further evaluate the VQ errors generated by ExVQ in Fig. 3.5, which plots the histograms

Figure 3.5: Histograms of error ranks of ExVQ on the four datasets with the codebook size $K = 256$ (blue) and $K = 1024$ (red). The horizontal axis is the error rank and the vertical axis shows the frequency number plot in natural logarithm.

of error ranks for ExVQ. ExVQ error ranks are all within a small range (compared to the codebook size) close to 1, which we believe are benign to classification.

When the codebook size is larger, the maximum error rank is also larger. The maximum error rank for $K = 1024$ (red curve) is 2.57–3.15 times of that for $K = 256$ (blue curve). One may conjecture that the reduction in the number of training descriptors of each codeword makes the linear classifiers inaccurate when K is larger. However, when we increase the training descriptors for $K = 1024$, we find that the maximum error rank is almost the same as that in Fig. 3.5. In fact, the VQ error rate is mostly dependent on the tree level L and the exclusion portion p .

3.4.3 Effects of the tree level L and exclusion portion p

L and p are the two parameters that can be tuned in ExVQ. In this section, we evaluate how L and p affect the trade off between the classification accuracy and the VQ time. First, we evaluate the classification accuracy and VQ time when we change L with a fixed p . We change L from 5 to 15 (with step size 2) and test on one round of Caltech 101 with $p = 1/5$, $K = 256$. The classification accuracy and the VQ time are shown in Fig. 3.6. When L is larger than 11, the VQ time is not reduced apparently. Besides, a larger L can induce a higher VQ error. However, all classification accuracies are similar with different

Figure 3.6: The classification accuracy and the VQ time with the tree level L and fixed exclusion portion p .

L . When L is even larger, the VQ time cost will be bigger and the classification accuracy will drop.

Next, we evaluate ExVQ with different exclusion portion p with a fixed L . We change $1/p$ from 3 to 11 (with step size 2) and fix $L = 10$ on one round of Caltech 101 with $K = 256$. The classification accuracy and the VQ time are shown in Fig. 3.7. When p is smaller, the training of linear classifiers in the tree will be more accurate. So the VQ error will be smaller. Meanwhile, in the leaf node of the ExVQ tree, the active search set will be larger, which leads to longer VQ time. However, the classification accuracies keep almost the same for different p . When p is even smaller, the VQ time will cost more and the accuracy can be a little better (the linear projection is more accurate).

3.4.4 ExVQ and other related methods

Finally, we compare ExVQ with more related methods including spill tree [44], LSH [12] and LLC [75]. We evaluate the VQ time and the classification accuracy of these methods on Caltech 101 with $K = 1024$. We use the same tree structure of ExVQ for spill tree: the same L and p . So we can compare their classification accuracy under equivalent

Figure 3.7: The classification accuracy and the VQ time with $1/p$ and fixed tree level L .

Table 3.4: Comparison with other related methods on Caltech 101, $K = 1024$.

ANN methods	VQ time (s)	Accuracy (%)
ExVQ	280	65.17
Spill tree	291	63.87
LSH	3247	63.97
LLC	4817	65.77

VQ time. We use the E²LSH package² to implement LSH, which is to search all near neighbors within a radius R of the query point. In order to use it in the BOV model, we search exhaustively from the returned results of LSH to get the nearest neighbor. In E²LSH, we use multiple radius for near neighbor search to save the VQ time. For LLC, we use the published Matlab code³ for testing. The results are shown in Table 3.4.

ExVQ has the shortest VQ time. Spill tree’s VQ time is close, but its classification accuracy drops significantly. Meanwhile, the VQ error rate of spill tree ($41\% \gg 30\%$, although the error ranks are small) is much more than that of ExVQ, which influences the classification accuracy. ExVQ has similar accuracy with LLC, but is about 20 times faster. Note that this comparison is qualitative rather than quantitative, because different

²<http://www.mit.edu/~andoni/LSH/>

³<http://www.ifp.illinois.edu/~jyang29/LLC.htm>

programming languages are used. We also want to point out that ExVQ uses only a single core, while LLC utilizes multiple cores of a six-core CPU. Since LLC also needs to search k NN, ExVQ can be used to accelerate it. LSH is effective in large scale NN search problems. However, it is not suitable to get the nearest neighbor from the codebook with $K = 1024$, which costs more time even than the exhaustive NN search. The VQ error of LSH is not benign: its histogram of error ranks has a long tail, and the maximum error rank is 997. LSH has a lower classification accuracy than ExVQ, which coincides with our conjecture about how VQ errors impact the classification.

3.5 Summary of the chapter

We presented ExVQ, an approximate nearest neighbor (ANN) search method for vector quantization (VQ) in BOV models. ExVQ achieves 10x speedup than exhaustive search and 2x speedup than state-of-the-art ANN methods. The key insight in ExVQ is an exclusive strategy, which can safely reduce the search range. The exclusion’s quality is guaranteed by learning an accurate classifier. Thus, it is more efficient than the commonly used inclusive strategy in ANN search. We also systematically evaluated how VQ errors affect the accuracy of BOV-based visual classification, and conclude that some VQ methods have benign VQ errors, which does not reduce the classification accuracy even with 30% VQ errors. On four benchmark datasets, ExVQ leads to indistinguishable accuracy rates with the exhaustive search method and the FLANN method.

Chapter 4

Hyper Spatial Matching for Accurate Image Representation

Many feature generation methods can improve their performance by using the state-of-the-art spatial pyramid matching (SPM) technique. In order to incorporate more comprehensive spatial information, several points should be noticed:

1. A region in one image is usually similar to the spatially surrounding regions in images from the same category;
2. When two images come from different categories, we should require that their similarity is as small as possible;
3. For a given problem or dataset, the spatial relationship among regions should be stable, *i.e.* the spatial relationship among regions computed from different subset of images in the dataset should be similar to each other.

Following these requirements, we propose the hyper-spatial matching (HSM) technique. The HSM similarity between two images is defined as:

$$K_{HSM}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{p=1}^N \sum_{q=1}^N R_{pq} \kappa(\mathbf{x}_i^p, \mathbf{x}_j^q) \quad (\text{Eq. 4.1})$$

where $R \in \mathbb{R}^N \times \mathbb{R}^N$ is a learned weight matrix encoding the relationships among different spatial regions, $\mathbf{x}_i^p$ and $\mathbf{x}_j^q$ are the feature vectors of p -th and q -th region in two images $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively. SPM can be included within HSM, when R is an identity matrix.

$K_{HSM}(\cdot)$ is a Mercer kernel, when R is a positive semidefinite (PSD) matrix and κ is a Mercer kernel. This will be proved in Sec. 4.2.1. So it can be used within SVM directly for image classification tasks.

Before proposing the details of hyper-spatial matching, we first summarize the notations used in this chapter:

- An image $\mathbf{x}$ has N regions $\mathbf{x} = (\mathbf{x}^1, \dots, \mathbf{x}^N)$, and each region is represented by a d -dimensional feature vector $\mathbf{x}^i \in \mathbb{R}^d$;
- The training set $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ contains n images. Each image is divided into N regions $\mathbf{x}_i = (\mathbf{x}_i^1, \dots, \mathbf{x}_i^N)$, and has a class label $y_i \in \{1, 2, \dots, N_c\}$, where N_c is the number of image categories; and,
- $x_i^{p,t}$ denotes the t -th dimension in the feature vector from the p -th region in the i -th image in X , where $i = 1, \dots, n$, $p = 1, \dots, N$, and $t = 1, \dots, d$.

We use an additive kernel κ to compare two regions throughout this chapter. For more details of the additive kernels, the readers are referred to [77, 73, 47].

4.1 Hyper-spatial matching

In this section, we introduce how to learn the relationship matrix R .

4.1.1 Learn an overall relationship matrix R

In hyper-spatial matching, the relationship matrix R plays a key role. We learn an overall relationship matrix R from the training samples of all categories, and require it to maximize the similarity of images from the same category, meanwhile to minimize the similarity of images from different categories. First, we define an $n \times n$ matrix D :

$$D_{ij} = \begin{cases} -\frac{1}{n^-} & y_i = y_j \\ \frac{1}{n^+} & y_i \neq y_j \end{cases}, \quad (\text{Eq. 4.2})$$

where n^- and n^+ are the number of same class pairs ($y_i = y_j$) and the number of cross class pair ($y_i \neq y_j$) in D , respectively. This matrix indicates whether images i and j belong to the same class or not. Since in a multiclass problem, the number of same class pairs is much smaller than that of pairs with different class labels, we normalize it with n^- and n^+ .

Then, the learning process of R is formulated as:

$$\begin{aligned} \min_R \quad & \sum_{i,j=1}^n \sum_{p,q=1}^N R_{pq} D_{ij} \kappa(\mathbf{x}_i^p, \mathbf{x}_j^q) + \gamma \|R\|_F^2 \\ \text{s.t.} \quad & R \succeq 0, \quad R_{pp} = 1, \forall p. \end{aligned} \quad (\text{Eq. 4.3})$$

where $\|\cdot\|_F$ is the Frobenius norm of a matrix, $R \succeq 0$ means R is a positive semidefinite matrix, and $\gamma > 0$ is a parameter to balance the loss on the training set and the regularizer on R . Note that R is a symmetric matrix, based on the fact that the pairwise relationship is symmetric. Meanwhile, the values in its diagonal are set to 1, so HSM can keep this same spatial correspondence as SPM. More importantly, HSM can include more meaningful matching between different spatial locations if off-diagonal values in R are non-zero.

Many existing kernels [26, 74, 46, 5] can be classified as a special case of HSM. For example, the kernels in [74, 46, 5] are used to compute the similarity of two bags (images) of local descriptors. When the bag sizes are the same, the sum-max kernel [74] is equivalent to HSM, by setting one specific column in each row of R to 1 (and 0 for the rest). The kernels in [46, 5] are equivalent to HSM by setting all values of R to 1.

The optimization of R is a standard semidefinite programming (SDP) problem, which can be efficiently solved. We change the order of summations in the objective function and define a new matrix $A \in \mathbb{R}^N \times \mathbb{R}^N$, where

$$A_{pq} = \sum_{i,j=1}^n D_{ij} \kappa(\mathbf{x}_i^p, \mathbf{x}_j^q). \quad (\text{Eq. 4.4})$$

Figure 4.1: Spatial relationship between different regions of human faces. This figure should be viewed in color.

This matrix encodes the average *affinity* or correlation of different regions, with the class labels y_i incorporated. Then, we can rewrite Eq. 4.3 in a more compact form, as:

$$\begin{aligned} \min_R \quad & \text{tr}(A^T R) + \gamma \text{tr}(R^T R) \\ \text{s.t.} \quad & R \succeq 0, \quad R_{pp} = 1, \forall p, \end{aligned} \quad (\text{Eq. 4.5})$$

where $\text{tr}(\cdot)$ is the trace of a matrix and $\text{tr}(R^T R) = \|R\|_F^2$. Since N is small (we use the SPM structure proposed by [78], where $N = 31$), it is a small scale SDP problem and can be solved in a few seconds.

In Fig. 4.1, we show an example of learned spatial relationship for normalized human faces. An R matrix is learned for all 7×7 regions with the size of 49×49 . Relationships of three parts (eye, nose and mouth out) to all 49 regions (which are three corresponding rows in R) are visualized according to R .¹ In each of the three similarity images, the dark red regions are the most similar to the query. We can see obvious symmetry between both eyes, and the self-similarity around the mouth and the nose.

¹This R is generated by replacing $\|R\|_F^2$ in Eq. 4.5 to $\|R\|_{\ell_1}$, in order to get a sparse R matrix for better visualization. In our experiments, F -norm has slightly higher classification accuracy than ℓ_1 -norm, so F -norm is used in the experimental part and Eq. 4.5.

4.1.2 Stability of the affinity and relationship matrices

One important premise for the proposed hyper-spatial matching framework to work well for image similarity computation is that: *for a specific problem, the relationship matrix R learned from Eq. 4.5 should be stable.* That is, if we use different images sampled from the same problem domain to calculate the affinity matrix A and to learn the relationship matrix R , we expect the learned matrices to be similar to each other.

Empirically, we observed that in our experiments both matrices A and R are in fact stable. We first sample five subsets from all available images of a specific dataset. Then, five versions of affinity matrices $A_1, \dots, A_5$ are computed. Relationship matrices $R_1, \dots, R_5$ are learned correspondingly. Next, we measure the variations of A or R by:

$$s_A = \frac{1}{5} \sum_{i=1}^5 \frac{\|A_i - \bar{A}\|_F}{\|\bar{A}\|_F}, \quad (\text{Eq. 4.6})$$

$$s_R = \frac{1}{5} \sum_{i=1}^5 \frac{\|R_i - \bar{R}\|_F}{\|\bar{R}\|_F}, \quad (\text{Eq. 4.7})$$

in which $\bar{A} = \frac{1}{5} \sum_{i=1}^5 A_i$, $\bar{R} = \frac{1}{5} \sum_{i=1}^5 R_i$, and $\|\cdot\|_F$ is the Frobenius norm.

The computed s_A and s_R are usually very small. For example, when we randomly sample five subsets in the Scene 15 dataset (each with 10% examples from the training dataset), we find that $s_A = 0.057$ and $s_R = 0.065$ (s_R is computed by removing 1s in the diagonal of R_i). Because both A and R are stable within a given problem, and the computation of affinity matrix A is expensive, in this chapter we simply use the same R for all experiments on one dataset. This relationship matrix is computed by sampling 10% examples from the training set.

4.2 Efficient learning of the SVM classifier

HSM causes more computations than SPM in comparing two images: $O(N^2d)$ vs. $O(Nd)$. Since it is not an additive kernel, recent developments of fast training algorithms for

additive kernel SVM are not applicable either. In this section, we propose two novel fast learning strategies.

We choose SVM as the classifier and solve the dual SVM problem (without a bias term):

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & f(\boldsymbol{\alpha}) = \frac{1}{2} \boldsymbol{\alpha}^T Q \boldsymbol{\alpha} - \mathbf{e}^T \boldsymbol{\alpha} \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, i = 1, \dots, n, \end{aligned} \quad (\text{Eq. 4.8})$$

where $\boldsymbol{\alpha}$ consists of the Lagrange multipliers, $\mathbf{e}$ is a vector with all ones, and $Q_{ij} = y_i y_j K_{HSM}(\mathbf{x}_i, \mathbf{x}_j)$. Then, the classifier can be acquired as: $\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \xi(\mathbf{x}_i)$, where ξ is an implicit mapping from the original input space to the feature space of K_{HSM} .

In the next two subsections, we propose two strategies to solve the problem (Eq. 4.8): a spectral linearization of the problem (in Sec. 4.2.1) and by directly approximating the gradient in SVM learning (in Sec. 4.2.2).

4.2.1 The spectral linearization strategy

The first strategy we propose to solve Eq. 4.8 is to linearize the non-linear HSM kernel approximately, using a spectral decomposition of R and the Fourier sampling technique in [73]. Any fast linear SVM solver can solve the converted approximate linear problem efficiently.

Since the additive kernel κ is a positive definite one, there exists a mapping ϕ that guarantees $\kappa(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^T \phi(\mathbf{y})$ for any examples (images) $\mathbf{x}$ and $\mathbf{y}$. With some abuse of notation, we define a ‘matrix’ $K_{\mathbf{x}}$ for any example $\mathbf{x}$, which is simply a reshaped version of $\phi(\mathbf{x})$:

$$K_{\mathbf{x}} = \begin{pmatrix} \phi(\mathbf{x}^1)^T \\ \vdots \\ \phi(\mathbf{x}^N)^T \end{pmatrix}. \quad (\text{Eq. 4.9})$$

Then, we define $K = K_{\mathbf{x}}K_{\mathbf{y}}^T$, in which $K_{\mathbf{y}}$ is defined similar to Eq. 4.9. So,

$$\begin{aligned} K_{HSM}(\mathbf{x}, \mathbf{y}) &= \sum_{p=1}^N \sum_{q=1}^N R_{pq} \phi(\mathbf{x}^p)^T \phi(\mathbf{y}^q) \\ &= \sum_{p=1}^N \sum_{q=1}^N R_{pq} K_{pq} \end{aligned} \quad (\text{Eq. 4.10})$$

$$= \text{tr}(R^T K). \quad (\text{Eq. 4.11})$$

where $K_{pq} = \kappa(\mathbf{x}^p, \mathbf{y}^q) = \phi(\mathbf{x}^p)^T \phi(\mathbf{y}^q)$. Note that K is dependent on $\mathbf{x}$ and $\mathbf{y}$, although we did not make this dependency explicit in the notation.

Because R is a real symmetric and positive semidefinite matrix, it has a spectral decomposition as:

$$R = \sum_{p=1}^N \lambda_p \mathbf{u}_p \mathbf{u}_p^T, \quad (\text{Eq. 4.12})$$

where $\lambda_p \geq 0$ and $\mathbf{u}_p \in \mathbb{R}^N$ are the eigenvalues and corresponding eigenvectors of R , respectively. With the spectral decomposition, a linearization of the HSM kernel is not difficult to figure out:

$$\begin{aligned} K_{HSM}(\mathbf{x}, \mathbf{y}) &= \text{tr}(R^T K) \\ &= \text{tr} \left(\sum_{p=1}^N \lambda_p \mathbf{u}_p \mathbf{u}_p^T K_{\mathbf{x}} K_{\mathbf{y}}^T \right) \\ &= \sum_{p=1}^N \lambda_p \text{tr}(\mathbf{u}_p^T K_{\mathbf{x}} K_{\mathbf{y}}^T \mathbf{u}_p) \\ &= \sum_{p=1}^N \left(\sqrt{\lambda_p} K_{\mathbf{x}}^T \mathbf{u}_p \right)^T \left(\sqrt{\lambda_p} K_{\mathbf{y}}^T \mathbf{u}_p \right), \end{aligned} \quad (\text{Eq. 4.13})$$

in which we used the fact that $\text{tr}(AB) = \text{tr}(BA)$ and $\text{tr}(x) = x$ when $x \in \mathbb{R}$. Then, the following mapping Ψ provides a linearization for HSM, and $K_{HSM}(\mathbf{x}, \mathbf{y}) = \Psi(\mathbf{x})^T \Psi(\mathbf{y})$ for all $\mathbf{x}$ and $\mathbf{y}$:

$$\Psi : \mathbf{x} = (\mathbf{x}^1, \dots, \mathbf{x}^N) \mapsto (\sqrt{\lambda_1} K_{\mathbf{x}}^T \mathbf{u}_1, \dots, \sqrt{\lambda_N} K_{\mathbf{x}}^T \mathbf{u}_N). \quad (\text{Eq. 4.14})$$

1 Convert dataset: for any training example $\mathbf{x}$ (replacing ϕ with $\hat{\phi}$ in equations):

- (i) Compute $\hat{K}_{\mathbf{x}}$ using Eq. 4.9;
- (ii) Convert $\mathbf{x}$ to $\hat{\Psi}(\mathbf{x})$ using Eq. 4.14.

Train a linear SVM with the new dataset $(\hat{\Psi}(\mathbf{x}_i), y_i), i = 1, \dots, n$. The converted examples and the trained linear classifier $\mathbf{w}$ are $(2m + 1)Nd$ dimensional.

Classification: for a new test image $\mathbf{x} = (\mathbf{x}^1, \dots, \mathbf{x}^N)$,

- (i) Convert $\mathbf{x}$ to $\hat{\Psi}(\mathbf{x})$ using the above conversion procedure;
- (ii) Output: $\text{sgn}(\mathbf{w}^T \hat{\Psi}(\mathbf{x}))$.

Algorithm 3: HSM SVM with spectral linearization

For many additive kernels κ , the mapping ϕ could be high or even infinite-dimensional (or does not have an explicit form), rendering Eq. 4.14 impractical. However, we can use the random Fourier sampling techniques for additive kernels in [73] to reach an approximate linearization. An additive kernel κ can be approximated by $\kappa(\mathbf{x}, \mathbf{y}) \approx \hat{\phi}(\mathbf{x})^T \hat{\phi}(\mathbf{y})$. The feature mapping $\hat{\phi}$ transforms any scalar value x into $2m + 1 (m \geq 1)$ dimensions, and $\hat{\phi}(\mathbf{x}) \in \mathbb{R}^{(2m+1)d}$ if $\mathbf{x} \in \mathbb{R}^d$, where m is a parameter controlling the mapping length. Details of the the Fourier sampling technique and examples of closed form feature mapping $\hat{\phi}$ for some commonly used additive kernels can be found in [73].

Replacing ϕ with $\hat{\phi}$ in Eq. 4.9 and Eq. 4.14, for any $\mathbf{x}$ in an HSM setting, $\hat{K}_{\mathbf{x}}$ is now a $N \times (2m + 1)d$ matrix, $\hat{K}_{\mathbf{x}}^T \mathbf{u}_p$ is $(2m + 1)d$ dimensional, and the approximate HSM linearization $\hat{\Psi}$ is a $(2m + 1)Nd$ dimensional vector. Then, any fast linear SVM solver can be used to solve the linearized problem. This fact shows that K_{HSM} is a Mercer kernel when R is a PSD matrix. The complete learning process is shown in Algorithm 3.

Before presenting the next learning strategy, we end this subsection with a technical note. Eq. 4.14 shows that $R \succeq 0$ is a sufficient condition for HSM to be a valid Mercer kernel. In fact, $R \succeq 0$ is also a necessary condition. If R is not positive semidefinite,

```

1 Input: Given the Lagrange multipliers  $\alpha$  and the corresponding
    $\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \xi(\mathbf{x}_i)$ .
2 Compute  $Q_{ii} = \|\xi(\mathbf{x}_i)\|_{\ell_2}^2, i = 1, \dots, n$ .
3 while  $\alpha$  is not optimal do
4   for  $i = 1, \dots, n$  do
5     Compute  $G = y_i \mathbf{w}^T \xi(\mathbf{x}_i) - 1$ 
6      $\bar{\alpha}_i \leftarrow \alpha_i$ 
7      $\alpha_i \leftarrow \min(\max(\alpha_i - G/Q_{ii}, 0), C)$ 
8      $\mathbf{w} \leftarrow \mathbf{w} + y_i(\alpha_i - \bar{\alpha}_i)\xi(\mathbf{x}_i)$ 

```

Algorithm 4: The coordinate descent method [84]

then $K_{HSM}(\mathbf{x}, \mathbf{x}) < 0$ may hold, which cannot happen in a Mercer kernel. One such example is when $R = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$, $N = 2$, $d = 1$, $\mathbf{x} = (1, -2)$, and $\kappa(x, y) = xy$, we have $K_{HSM}(\mathbf{x}, \mathbf{x}) = -3$.

4.2.2 The gradient approximation strategy

The second strategy we propose directly approximates the gradient in coordinate descent SVM learning. This strategy is better than the spectral linearization strategy in practice: SVM training is faster and requires less memory; meanwhile it has a slightly higher classification accuracy (cf. Sec. 4.3).

In order to efficiently solve the problem (Eq. 4.8), we choose the dual coordinate descent method [84], which is shown in Algorithm 4. We approximate the gradient G (line 5 in Algorithm 2) using the gradient approximation method in [77].

First, we substitute K_{HSM} into G :

$$\begin{aligned}
G &= y_i \mathbf{w}^T \xi(\mathbf{x}_i) - 1 \\
&= y_i \sum_{j=1}^n \alpha_j y_j K_{HSM}(\mathbf{x}_i, \mathbf{x}_j) - 1 \\
&= y_i \sum_{j=1}^n \alpha_j y_j \sum_{p=1}^N \sum_{q=1}^N R_{pq} \kappa(\mathbf{x}_i^p, \mathbf{x}_j^q) - 1 \\
&= y_i \sum_{j=1}^n \alpha_j y_j \sum_{p=1}^N \sum_{q=1}^N R_{pq} \sum_{t=1}^d \kappa(x_i^{p,t}, x_j^{q,t}) - 1 \\
&= y_i \sum_{p=1}^N \sum_{t=1}^d \sum_{q=1}^N R_{pq} \sum_{j=1}^n \alpha_j y_j \kappa(x_i^{p,t}, x_j^{q,t}) - 1. \tag{Eq. 4.15}
\end{aligned}$$

It needs $O(nN^2d)$ steps to compute, which is very expensive. In Eq. 4.15, the essential part is $\sum_{j=1}^n \alpha_j y_j \kappa(x, x_j^{q,t})$ for any scalar value x . We define this summation as a function $g_t^q(x)$.

Then, we use an m' degree polynomial regression model as in [77] to approximate it:

$$g_t^q(x) = \sum_{j=1}^n \alpha_j y_j \kappa(x, x_j^{q,t}) \approx \sum_{s=0}^{m'} a_s^{q,t} x^s, \tag{Eq. 4.16}$$

where $a_s^{q,t}$ is the polynomial regression parameter for the t -th dimension in the q -th region, and x^s is x raised to the s -th power.

Next, we need to learn the $m' + 1$ parameters $a_s^{q,t}$ in the model. We choose $m' + 1$ values $\mathbf{c} = (c_0, \dots, c_{m'})$ between $[0, 1]$ and calculate $g_t^q(c_i)$, $i = 0, \dots, m'$. The optimal parameters of $\mathbf{a}^{q,t} = (a_0^{q,t}, \dots, a_{m'}^{q,t})$ equals [77]

$$\mathbf{a}^{q,t} = X^{-1} g_t^q(\mathbf{c}), \tag{Eq. 4.17}$$

where X is the Vandermonde matrix with $X_{ij} = (c_i)^j$, $\forall i, j$, and $g_t^q(\mathbf{c}) = (g_t^q(c_0), \dots, g_t^q(c_{m'}))$.

To simplify the notations, we denote $\mathbf{b}_s^{p,t} = \sum_{q=1}^N R_{pq} a_s^{q,t}$, and the vector form is now:

$$\mathbf{b}^{p,t} = \sum_{q=1}^N R_{pq} \mathbf{a}^{q,t}. \tag{Eq. 4.18}$$

```

1  $\alpha \leftarrow 0$ 
2  $b_s^{p,t} \leftarrow 0, p = 1, \dots, N, t = 1, \dots, d, s = 0, 1, \dots, m'$ 
3  $Q_{ii} \leftarrow K_{HSM}(\mathbf{x}_i, \mathbf{x}_i), i = 1, \dots, n.$ 
4 while  $\alpha$  is not optimal do
5   for  $i = 1, \dots, n$  do
6     Compute  $G$  using Eq. 4.19.
7      $\bar{\alpha}_i \leftarrow \alpha_i$ 
8      $\alpha_i \leftarrow \min(\max(\alpha_i - g/Q_{ii}, 0), C)$ 
9      $\mathbf{b}^{p,t} \leftarrow \mathbf{b}^{p,t} +$ 
10     $(\alpha_i - \bar{\alpha}_i)y_i X^{-1} \sum_{q=1}^N R_{pq} \kappa(\mathbf{c}, x_i^{q,t}), \forall p, t.$ 
11 Classification: For a new test image  $\mathbf{x} = (\mathbf{x}^1, \dots, \mathbf{x}^N)$  divided into  $N$  regions,
    the classification result is:
12  $\text{sgn} \left( \sum_{p=1}^N \sum_{t=1}^d \left( \sum_{s=0}^{m'} b_s^{p,t} (x^{p,t})^s \right) \right).$ 

```

Algorithm 5: HSM SVM with gradient approximation

Finally, we have the gradient approximated as:

$$G = y_i \sum_{p=1}^N \sum_{t=1}^d \left(\sum_{s=0}^{m'} b_s^{p,t} (x_i^{p,t})^s \right) - 1. \quad (\text{Eq. 4.19})$$

Meanwhile, given a learned classifier $\mathbf{w}$ and a new testing image $\mathbf{x}$, the SVM decision value is:

$$\mathbf{w}^T \xi(\mathbf{x}) = \sum_{p=1}^N \sum_{t=1}^d \left(\sum_{s=0}^{m'} b_s^{p,t} (x^{p,t})^s \right). \quad (\text{Eq. 4.20})$$

In Eq. 4.19 and Eq. 4.20, m' is usually very small, e.g., $m' = 2$, so the complexity are greatly reduced from $O(nN^2d)$ to $O(Nd)$. Note that Nd is the length of the whole feature vector for an image in SPM. Thus, *HSM has the same testing complexity as a linear classifier*. The complete learning process of HSM SVM using the gradient approximation strategy is given in Algorithm 5.

Note that if we want to compare the two proposed HSM SVM strategies, we need to make $2m + 1$ (the number of dimensions in $\mathbf{w}$ corresponding to a single dimension in the input in Algorithm 3) and $m' + 1$ (in Algorithm 5) equal to each other. In this chapter, we choose $m = 1$ and $m' = 2$ in our experiments.

```

1  $\alpha \leftarrow 0, \hat{\mathbf{w}} \leftarrow 0.$ 
2  $Q_{ii} \leftarrow \sum_{p=1}^N \sum_{q=1}^N R_{pq}(\mathbf{x}_i^p)^T(\mathbf{x}_i^q), i = 1, \dots, n.$ 
3 while  $\alpha$  is not optimal do
4   for  $i = 1, \dots, n$  do
5     compute  $G$  using Eq. 4.21.
6      $\bar{\alpha}_i \leftarrow \alpha_i$ 
7      $\alpha_i \leftarrow \min(\max(\alpha_i - g/Q_{ii}, 0), C)$ 
8      $\hat{\mathbf{w}}^{p,t} \leftarrow \hat{\mathbf{w}}^{p,t} + (\alpha_i - \bar{\alpha}_i)y_i \sum_{q=1}^N R_{pq}\mathbf{x}_i^q$ 
9 Classification: For a test image  $\mathbf{x}$ , output:  $\text{sgn}(\hat{\mathbf{w}}^T \mathbf{x}).$ 

```

Algorithm 6: Dot product HSM SVM

4.2.3 Dot product HSM SVM

Both proposed strategies deal with HSM with a base non-linear additive kernel. When the base kernel κ is a dot product kernel, we can directly derive an *exact* formulation for HSM SVM:

$$G = y_i \sum_{p=1}^N \sum_{t=1}^d x_i^{p,t} \left(\sum_{q=1}^N R_{pq} \sum_{j=1}^n \alpha_j y_j x_j^{q,t} \right) - 1, \quad (\text{Eq. 4.21})$$

and the decision value is:

$$\mathbf{w}^T \xi(\mathbf{x}) = \sum_{p=1}^N \sum_{t=1}^d x^{p,t} \left(\sum_{q=1}^N R_{pq} \sum_{j=1}^n \alpha_j y_j x_j^{q,t} \right). \quad (\text{Eq. 4.22})$$

By defining $\hat{\mathbf{w}}$, where $\hat{w}^{p,t} = \sum_{q=1}^N R_{pq} \sum_{j=1}^n \alpha_j y_j x_j^{q,t}$, we get $\mathbf{w}^T \xi(\mathbf{x}) = \hat{\mathbf{w}}^T \mathbf{x}$, which has the same complexity as a linear classifier. The learning process is shown in Algorithm 6.

4.2.4 Decorrelation effect of HSM

The fundamental motivation of hyper-spatial matching comes from the observation that different spatial regions are in fact interacting with each other; or, *correlated with each other*. Thus, we expect the matrix A that encodes affinity of regions (computed from Eq. 4.4) has many non-zero off-diagonal entries. These correlations make HSM highly non-linear.

Figure 4.2: Affinity matrix (computed on the Scene 15 problem [38]) of the original dataset; of the linearized dataset after applying HSM; and the eigenvalue normalized version after applying HSM, respectively. This figure should be viewed in color.

However, as shown in the spectral linearization based strategy we just proposed, HSM can be approximately linearized. Since the linearized version is an approximation of the original non-linear HSM problem, in the linearized problem, *different regions should not correlate with each other any more*. That is, we would expect *the relationship matrix R of HSM to decorrelate those spatial regions that are originally correlated*. This statement is empirically verified in Fig. 4.2.

The left image in Fig. 4.2 visualized the affinity matrix A computed from the original dataset. It is obvious that although the diagonal entries have the largest absolute values, many off-diagonal entries have correlations that can not be ignored. However, if we compute the affinity matrix A again on the linearized dataset, the middle image demonstrates that most off-diagonal entries are close to 0. That is, after applying HSM to linearize the problem, different spatial regions now are not correlated with each other in most cases.

Two points deserve further attention. First, because in Eq. 4.14, we sort the eigenvalues as $\lambda_1 \geq \lambda_2 \geq \dots$, the values in the middle image decreases from top to bottom and from left to right. If we remove the effect of eigenvalues (i.e., remove all λ_i from Eq. 4.14), the affinity matrix of the linearized dataset is shown in the right part of Fig. 4.2. It is obvious that the decorrelation effect still holds.

Second, there are a few remaining off-diagonal entries with relatively large values after linearization. However, all of them involve regions from the level 0 or level 1 in the spatial pyramid. Such regions cover the entire image (level 0) or a quarter of the entire image (level 1). Thus, it is possible that they still have remaining correlation with level 2 region that are contained in them.

Overall, we believe that

- Correlations exist among regions at different spatial locations, which cannot be safely ignored; and,
- Through the learned relationship matrix R , HSM successfully decorrelates different spatial regions.

These observations may explain why HSM has an advantage over SPM in various image categorization tasks in our experiments (cf. Sec. 4.3).

4.2.5 Hyper-Spatial Features

For very high dimensional visual representation such as Fisher vector [55], even Algorithm 6 could be too costly. Thus, we propose a simple way to append “hyper-spatial features” (HSF) to approximate HSM. Given an image with N regions, $\mathbf{x} = (\mathbf{x}^1, \dots, \mathbf{x}^N)$, we append the following $\frac{N(N-1)}{2}$ values to the end of $\mathbf{x}$:

$$(\mathbf{x}^p)^T(\mathbf{x}^q), \quad 1 \leq p < q \leq N. \quad (\text{Eq. 4.23})$$

When using the linear (dot-product) kernel on the expanded vectors, the above appended within-image spatial similarity values will approximate the HSM similarity.

Encoding spatial information is related to [64], which appends coordinates into local visual descriptors to include the spatial information. In contrast, HSF encodes the spatial relation between regions rather than local descriptors. [64] requires the spatial smoothness among local visual descriptors. Instead, HSF values are additional useful contextual information within spatial regions in an image.

Table 4.1: Overview of HSM methods in the experiment.

notation	κ	SVM solving method
HSM-ga- χ^2	χ^2	gradient approximation
HSM-ga-HI	HIK	gradient approximation
HSM-sl- χ^2	χ^2	spectral linerization
HSM-sl-HI	HIK	spectral linerization
HSM-linear	linear	direct solving
HSM-exact- χ^2	χ^2	precomputed kernel

4.3 Experimental for HSM

In this section, we empirically compare HSM with state-of-the-art methods. The two proposed strategies are abbreviated as “ga” and “sl”, to denote gradient approximation and spectral linearization, respectively. We use the two most widely used additive kernels for κ : χ^2 and histogram intersection (HI). They are denoted as HSM-ga(sl)- χ^2 (HI). We also denote HSM with linear kernel as HSM-linear.

Eq. 4.5 is solved by using the CVX [25] software package and $\gamma = 0.2$. In Eq. 4.8, $C = 0.01$ for gradient approximation with non-linear kernels, $C = 1$ for spectral linearization with non-linear kernels, and $C = 1$ for the linear kernel, respectively.² In multi-class classification tasks, we use the one-vs-rest strategy. An overview of HSM methods is listed in Table 4.1.

In Algorithm 5, $X^{-1} \sum_{q=1}^N R_{pq} \kappa(\mathbf{c}, x_i^{q,t})$ in Line 9 is used frequently during the iterations, which can be precomputed and stored to save lots of computations in this process. It is the same for Line 8 of Algorithm 6. If the dataset is too large to fit into the memory, we have to compute it in every iteration, however, which is still more efficient than other available method, e.g., LIBSVM [7]. In this work, all datasets can fit into the memory, so we use the former strategy.

Experiments are organized as follows. First, we compare the proposed approximate learning strategies with exact HSM SVM. We use LIBSVM [7] to implement the exact

²These values are default values for the PmSVM and LIBLINEAR software package, respectively.

HSM kernel K_{HSM} . In order to use the HSM kernel in LIBSVM, we precompute the kernel matrices for both training and testing purposes, which are used as the input of LIBSVM with the precomputed kernel option “-t 4”.

Second, we compare HSM with SPM using both proposed strategies and available fast classifiers with linear / nonlinear base kernels. Two fast methods with nonlinear kernels are used for SPM:

PmSVM [77] (SPM-ga). PmSVM includes a family of additive kernels $\kappa(x, y) = (\frac{x^p + y^p}{2})^{1/p}$, $x, y \in \mathbb{R}$. When $p = -1$, it is the χ^2 kernel, we denote it as SPM-ga- χ^2 . When $p = -\infty$, it is the histogram intersection kernel $\kappa(x, y) = \min(x, y)$. In [77], the author uses $p = -8$ to approximate the HI kernel denoted by SPM-ga-HI. We set $C = 0.01$.

Feature mapping (fm) [73] (SPM-sl). We map each dimension of the original feature vector into a 3-dimensional vector (that is, $m = 1$) and concatenate them altogether into a long vector. These mapped feature vectors are used to train a linear classifier with LIBLINEAR, where $C = 1$.

The fast linear classifier for SPM we adopted in our experiments is:

LIBLINEAR [14] (SPM-linear). LIBLINEAR is a fast linear SVM solver using dual coordinate descent. We use its default parameter $C = 1$.

We evaluate the above methods on three datasets. They are:

Scene 15 [38]. It contains 15 scene classes. 100 images from each class are used for training the classifier and all the rest images are used for testing.

Indoor 67 [60]. The dataset contains 67 classes. 80 images of each class are used for training and 20 for testing.

SUN [81]. It has 397 classes. In each class, we use 50 training images and 50 testing images.

A three level spatial pyramid in Fig. 4.3 is used for BOV, following the pyramid structure in [78]. The region numbers are 1, 5, and 25 on three levels of the spatial

Figure 4.3: Three level spatial pyramid of an image on level 2, 1, and 0, respectively [78].

pyramid, respectively. We extract SIFT descriptors for all the datasets. In the BOV process, different codeword numbers are used for different datasets in order to compare with existing works fairly. We also evaluate HSF + Fisher vector (FV) [55]. All feature vectors are subject to the power normalization [57] and ℓ_2 normalization for each region. Following the setup in [36], we use non-linear classifiers for BOV and linear classifier for FV.

We report the classification accuracy, the training time and the testing time of all methods in the experiments. All the results of each dataset are averaged on 5 rounds by randomly splitting the training and testing sets. We also compare our methods with recent published results on these three datasets. Experiments are run on a computer with Intel Core i7 4930K CPU (using one core) and 32G main memory.

4.3.1 HSM combined with BOV features

In this section, we first study the performance of HSM when BOV features are used.

4.3.1.1 Scene 15

In Table 4.2, we show the results (training time, testing time and classification accuracy) of different methods on Scene 15 using BOV feature. 1000 codewords are used for this dataset, following [68, 36].

All HSM accuracy rates are consistently higher than all the SPM accuracy results, which demonstrates the effectiveness of HSM in providing better spatial matching. The

Table 4.2: Results on Scene 15: BOV with 1000 visual codewords.

Method	Training(s)	Testing(s)	Accuracy(%)
HSM-ga- χ^2	23.66	8.50	85.51 $\pm$ 0.19
HSM-ga-HI	22.15	8.54	85.60$\pm$0.21
HSM-sl- χ^2	49.54	8.66	85.21 $\pm$ 0.33
HSM-sl-HI	48.13	8.65	85.22 $\pm$ 0.31
HSM-linear	13.00	6.59	83.58 $\pm$ 0.42
HSM-exact- χ^2	16181.88	31264.56	85.35
SPM-ga- χ^2	10.36	8.47	83.59 $\pm$ 0.28
SPM-ga-HI	13.72	8.47	83.62 $\pm$ 0.29
SPM-sl- χ^2	17.09	8.66	83.31 $\pm$ 0.33
SPM-sl-HI	16.85	8.65	83.34 $\pm$ 0.27
SPM-linear	7.92	6.64	82.42 $\pm$ 0.35
Spatial saliency [68]			85.5 $\pm$ 0.6
Spatial Fisher Vector [36]			85.0 $\pm$ 0.8

first block shows the results of HSM and the second block shows the results of SPM. The difference between the training time of HSM-ga- χ^2 (HI) and SPM-ga- χ^2 (HI) is that HSM needs to compute $\mathbf{b}$ from $\mathbf{a}$ (Eq. 4.18), which is similar for “sl” method (needs to do data transformation, Eq. 4.14). HSM-ga is faster than HSM-sl in the training phase, which is related to the observation that HSM-ga converges faster than HSM-sl.

A general purpose SVM solver is not suitable for our HSM kernel. We run one round using a general purpose SVM solver (LIBSVM) to solve HSM with χ^2 kernel (HSM-exact- χ^2). LIBSVM uses precomputed kernel matrices for training and testing, which costs much more time than all the fast learning methods.

Compared with related methods [68, 36], HSM has slightly higher classification accuracy than them. It is worth noting that HSM is more stable than the methods in [68, 36]. The standard deviation of different HSM algorithms’ accuracy rates ranges from 0.2–0.4, while [68, 36] have 0.6 and 0.8 standard deviation, respectively.

Finally, we show in Fig. 4.4 the relationship between regions according to the learned overall matrix R . A region usually has a high correlation with its surrounding regions. This relationship matrix is learned from the Scene 15 dataset where an object can appear

Figure 4.4: The relationship of a region (shown in dark red) to other spatial regions. The regions with light red color have close correlation with the dark red region according to the relationship matrix R learned from all training images. This figure should be viewed in color.

Table 4.3: Results on Indoor 67: BOV with 4000 visual codewords.

Method	Training(s)	Testing(s)	Accuracy(%)
HSM-ga- χ^2	896.02	14.77	53.98 $\pm$ 0.68
HSM-ga-HI	768.99	14.63	54.44$\pm$0.71
HSM-sl- χ^2	1492.91	22.16	53.14 $\pm$ 0.65
HSM-sl-HI	1471.29	22.07	53.47 $\pm$ 0.69
HSM-linear	295.63	10.71	50.25 $\pm$ 0.69
HSM-exact- χ^2	599472.03	144550.15	53.43
SPM-ga- χ^2	300.29	14.63	52.56 $\pm$ 0.59
SPM-ga-HI	328.75	14.64	52.89 $\pm$ 0.67
SPM-sl- χ^2	460.59	21.94	52.37 $\pm$ 0.86
SPM-sl-HI	460.59	22.01	52.56 $\pm$ 0.84
SPM-linear	270.22	10.96	48.58 $\pm$ 0.54
BOVW+aug. [71]			50.22

(almost) anywhere. When learning from object images, e.g., frontal human faces in Fig. 4.1, stronger relation can be found between local regions. However, even for scene recognition, the simple neighborhood relationship can consistently improve recognition accuracy, as shown by our experiments.

4.3.1.2 Indoor 67

In Table 4.3, we show the results on Indoor 67, with 4000 codewords.

All HSM (the first block) gets better accuracy than all methods using SPM (the second block). HSM also shows better results than related methods. The state-of-the-art results using BOV features is [71], which augmented coordinates to SIFT features. We use a codebook size of 4000, which is similar to but smaller than theirs (4096). The proposed HSM-linear approach (50.25%) achieves comparable accuracy to theirs (50.22%) using the linear classifier, and nonlinear HSM results are 4% higher (54.44%).

4.3.1.3 SUN

SUN is a large vision dataset involving about 20,000 images in both the training and the testing processes. 4000 visual codewords are used.

Table 4.4: Results of SUN: BOV with 4000 visual codewords.

Method	Training(s)	Testing(s)	Accuracy(%)
HSM-ga- χ^2	3711.59	382.96	35.89$\pm$0.15
HSM-ga-HI	3691.44	383.41	35.88 $\pm$ 0.18
HSM-linear	2419.63	248.30	32.10 $\pm$ 0.23
SPM-ga- χ^2	2539.59	386.33	33.98 $\pm$ 0.13
SPM-ga-HI	2717.42	384.28	34.18 $\pm$ 0.19
SPM-linear	2128.53	266.34	30.53 $\pm$ 0.22

The results are reported in Table 4.4. HSM (block 1) always achieves better accuracy than SPM (block 2). In [81], different features are used to evaluate on this dataset. The accuracy using dense SIFT feature is 21.5%, which only use 300 visual codewords (fewer than our experiments) to compute BOV. When multiple features are used altogether, the accuracy reaches 38.0%. We also expect HSM to achieve higher accuracy using the more powerful features.

4.3.2 Hyper-Spatial Features combined with Fisher Vector and VLAD

We evaluate HSF on two high-dimensional features: Fisher vector (FV) [55] and VLAD [32]. We use the SPM structure with 8 spatial regions: 1×1 , 2×2 , and 3×1 , which is a commonly used SPM structure with FV and VLAD. SIFT is used as base feature, which is reduced to 64 dimension by PCA. We extract FV with 100 Gaussian mixture models (GMM) and VLAD with 100 visual codewords. Only the mean and standard deviation prats in FV are used as [55]. Thus, one image has $8 \times 2 \times 100 \times 64 = 102400$ dimensions in FV and $8 \times 100 \times 64 = 51200$ dimensions in VLAD.

One virtue of HSF is that it is almost “free”. The appended HSF features have only $\frac{8 \times 7}{2} = 28$ values, 0.027% and 0.055% increase in FV and VLAD, respectively, which has negligible effect on the storage size, the training and testing time. FV (VLAD) and HSF experimental results are shown in Table 4.5.

Table 4.5: HSF Classification accuracy (%) on FV and VLAD.

Datasets	FV	FV+HSF	t -test
Scene 15	87.86 $\pm$ 0.29	88.21 $\pm$ 0.21	7.6×10^{-3}
Indoor 67	58.85 $\pm$ 0.64	59.35 $\pm$ 0.53	2.0×10^{-3}
SUN	38.57 $\pm$ 0.27	38.82 $\pm$ 0.29	1.6×10^{-3}
Datasets	VLAD	VLAD+HSF	t -test
Scene 15	87.58 $\pm$ 0.25	87.93 $\pm$ 0.17	3.8×10^{-2}
Indoor 67	54.38 $\pm$ 0.44	55.16 $\pm$ 0.70	2.7×10^{-2}
SUN	34.77 $\pm$ 0.43	35.46 $\pm$ 0.52	1.1×10^{-4}
	Scene 15	Indoor 67	SUN
CNN [17]	84.23 $\pm$ 0.37	56.79	42.61 $\pm$ 0.16

It is obvious that HSF improves both FV and VLAD, although the improvements are smaller compared with HSM. This is because the dimension of the appended HSF features is much less than that of FV and VLAD, which contributes less scores when they are computed with classifiers. We want to emphasize that the HSF improvement is noticeable for two reasons. First, HSF improves upon FV / VLAD almost “for free” (i.e., with negligible additional cost). Second, the improvements are consistent and significant: FV (VLAD)+HSF outperforms FV (VLAD) in all the $5 \times 3 = 15$ runs in Table 4.5; and the differences on all three datasets are significant by paired t -test at significance level 0.05.

In the FV experiment, we use the same number of GMM as the setup of the best result in [36]. They achieved the best accuracy $88.2 \pm 0.6\%$ on Scene 15 dataset. Our result $88.21 \pm 0.21\%$ on this dataset is almost similar to theirs. We also compare our methods with deep convolutional neural networks (CNN) on these datasets. We cite the baseline results in [17] for comparison, *i.e.*, each image is encoded into a 4K-dimensional vector with the pre-learned CNN model. We can see that FV+HSF has better results than CNN on Scene 15 and Indoor 67, but a lower result on SUN. We conjecture this is because the small size of the datasets (Scene and Indoor) limits the performance of CNN.

4.3.3 Discussions

After presenting experimental results and related observations, we now provide some more subjective comments to summarize the findings of this chapter, and to discuss the limitations and drawbacks of the proposed methods.

First, the focus of hyper-spatial matching is to take into account of matching of regions at different spatial locations. Concerning spatial matching in image similarity computation, we find that:

- Although Spatial Pyramid Matching (SPM) and almost all of its improvements focused on matching between regions at the same spatial locations in two images, regions at different spatial locations have significant correlations that cannot be ignored (cf. Fig. 4.2);
- In the image classification application, hyper-spatial matching (HSM), by explicitly considering relationships among all spatial regions, consistently achieves higher accuracy than SPM (cf. Tables 4.2-4.4);
- HSF, a simplified version of HSM, considers the relationship between spatial regions in one image. HSF consistently improves the classification ability for very long features (cf. Table 4.5), although the improvement value is not very significant, it almost does not incur additional cost in memory or CPU usage.
- In the relationship learned by HSM from scene images, a region only has non-trivial relationship with spatially nearby regions, and the relationship strength at the same spatial location is the strongest amongst all regions. In this sense, SPM intuitively captures the most important components of all spatial relationships in HSM (cf. Fig. 4.4).

- The learned spatial relationships are stable inside a specific problem or dataset (cf. Sec. 4.1.2) and can be efficiently computed even for large scale datasets.

Furthermore, we have proposed two learning strategies (gradient approximation “ga” and spectral linearization “sl”) for HSM SVM. The following findings can be concluded concerning HSM SVM learning:

- The effectiveness of relationship matrices (R) can be explained by the fact that when it is used in the HSM kernel, it effectively decorrelates the relationships among regions at different spatial locations (cf. Fig. 4.2);
- The HSM-ga strategy (cf. Algorithm 5) consistently achieves the best accuracy and efficiency during both training and testing (cf. Tables 4.2-4.4). In image classification with the hyper-spatial matching kernel, HSM-ga is the preferred method;
- HSM-sl (cf. Algorithm 3), on the other hand, is inferior in terms of practical performance. However, this strategy provides convenient tools to understand and visualize spatial matching (cf. Fig. 4.2). And, while HSM-ga is mostly confined to be used with SVM learning, explicit mapping (cf. Eq. 4.14) has applications in wider domains;
- One more note is that although we present the HSM framework and learning strategies based on SPM, it has wider range of applications. For example, it can also be combined with the receptive fields in [33].

4.4 Summary of the chapter

In this chapter, we propose hyper-spatial matching (HSM), a framework for flexibly matching two images. HSM takes into account the interactions among all pairs of spatial regions in two images, which alleviates the mismatching problem in SPM. Thus, HSM

provides more flexibility and better similarity measures for comparing two images than SPM.

We use relationship matrices to capture the relationship or correlation between any two spatial locations of all images. Training images are used to learn a relationship matrix R for a dataset. To deal with the high computational cost induced by HSM, we propose two fast learning strategies for HSM SVM learning and testing. HSM-ga is based on the dual coordinate descent SVM framework, where polynomial regression is used to approximate the SVM gradient on each dimension of the feature vector. It is hundreds of times faster than a general purpose SVM solver in training and testing without loss of accuracy. In the experiment, we compare HSM-ga with state-of-the-art methods on three large scale scene datasets and one multi-label image classification problem. The proposed fast classifier with HSM kernel shows better classification accuracy than compared methods.

The other proposed learning strategy is HSM-sl, which provides an approximate linearization strategy for HSM and uses the feature mapping approach for additive kernels. Although HSM-sl is inferior to HSM-ga in practical performance, it provides ways to study the spatial matching behaviors. HSM-sl reveals that the hyper-spatial matching kernel is effective because it decorrelates the interactions among spatial regions at different locations.

For very long features like Fisher vector and VLAD, HSF is proposed to approximate HSM. HSF computes the relationship between all spatial regions in one image. By appending HSF to the original features, the classification accuracy is improved consistently, although not significantly, it generates little additional cost in memory and CPU usage.

Chapter 5

Feature Selection for Feature Dimension Reduction

In this chapter, we investigate how to reduce the storage of high dimensional feature vectors. When the feature dimension becomes high, many useful methods such as HSM cannot apply on them, where the memory becomes a severe problem. Thus, we want to represent images compactly to reduce their storage.

Existing feature compression methods like product quantization (PQ) [29], BPBC [23], ITQ [24] use the linear transformations to change original high-dimensional feature vectors into shorter ones. The discriminance of these compressed features usually decreases monotonically when the compression ratio increases.

We propose to use feature selection (FS) to reduce the storage of high-dimensional feature vectors. It selects important features for image recognition and store them. The selected features can improve the discriminance by discarding noisy / redundant features from original feature vectors. Specifically, we use a mutual information (MI) based importance sorting algorithm to select useful features.

5.1 Feature selection by mutual information based importance sorting

In this section, we first study properties of high dimensional features and show that selection may be a better choice than compression. We then propose a mutual information based importance sorting feature selection method.

5.1.1 Feature selection vs. (bi-)linear projection

We start by studying the correlation between any two dimensions in FV vectors. Pearson’s correlation coefficient measures correlation between dimensions i and j :

$$r = \frac{\mathbf{x}_{:i}^T \mathbf{x}_{:j}}{\|\mathbf{x}_{:i}\| \|\mathbf{x}_{:j}\|}, \quad (\text{Eq. 5.1})$$

where $\mathbf{x}_{:i}$ is the vector formed by the i -th dimension of FV of all images in a dataset, and $r \in [-1, 1]$ measures the linear dependency between the two dimensions (± 1 for total correlation and 0 for no correlation).

We show the histograms of r values in Fig. 5.1 computed from Fisher vectors of the Scene 15 dataset [38]. There are 128 Gaussian components and 64 dimensional SIFT (after PCA) in FV and 8 spatial regions, leading to 131072 dimensions. Three types of r values are computed for all pairs of dimensions in differently sampled subsets:

- **Random:** randomly sample 1000 dimensions;
- **Same region:** randomly choose a spatial region, then randomly sample 10% dimensions from this region;
- **Same Gaussian:** randomly choose a Gaussian component in FV, then use all dimensions corresponding to it.

Fig. 5.1 shows that in all subsets, almost all r values are very close to zero—more than 99.9% satisfy that $|r| < 0.2$. The RANDOM and SAME REGION curves are almost identical

Figure 5.1: Distribution of Pearson’s correlation coefficients between dimensions in Fisher vectors.

except in the zero point, meaning that feature correlation is not affected by spatial pyramid. Dimensions sampled from the same Gaussian component (*i.e.*, supposedly visually similar) exhibits slightly higher correlations, but the correlation level is still low.

We observed similar results using the VLAD representation [32] and in all datasets we evaluated in Sec. 5.3. Thus, linear dependency between two dimensions (that is, collinearity) almost does not exist in high dimension visual representations. Although statistical tests for multicollinearity (strong linear relationship among more than 2 features) are too expensive to perform for FV or VLAD, given the missing of collinearity, we have reasons to conjecture that the chance of multicollinearity is also small.

An argument now naturally follows: if we have belief in the missing of multicollinearity, feature compression does not seem to be the best choice for reducing feature dimensionality, because linear and bilinear projections both hinge on strong multicollinearity among dimensions, especially among those features that are contiguous in FV or VLAD. Note that the SAME REGION and SAME GAUSSIAN subset in Fig. 5.1 both contain such contiguous dimensions.

We will also show in the next section that *most* dimensions are noise and are not useful (if not harmful) for classification. Feature compression use linear projections to include all these noisy features into the new compressed features, which is also sub-optimal. Thus, we propose to choose a subset of “good” features (*i.e.*, feature selection) for dimension reduction.

5.1.2 Mutual information (MI) based importance sorting

There is a vast literature on feature selection. For example, given a decision function $\mathbf{w}^T \mathbf{x}$, if $\mathbf{w}$ is learned with the ℓ_1 norm regularizer, the nonzero elements in $\mathbf{w}$ correspond to the selected features. [70] proposes a Feature Generating Machine, which learns a binary indicator vector to show which dimension is useful or not. [54] considers the dependency between a dimension and labels, and the redundancy between features to select useful features based on mutual information. [16] iteratively selects a new feature based on its mutual information with labels and already selected features. However, these methods are all too expensive to be applied in our large scale image classification problems. Thus, we will use a classic mutual information based importance sorting approach. It is similar to [54] in that mutual information (MI) between a dimension and the labels is used to compute the importance of it, but dependencies among different dimensions are ignored such that we can afford the computational cost.

This is a supervised method—we use the image labels¹ to estimate how useful a single dimension is. Specifically, we denote image labels as $\mathbf{y}$, the i -th dimension FV values as $\mathbf{x}_{:i}$, and the mutual information as $I(\mathbf{x}_{:i}, \mathbf{y})$. In general, the larger the mutual information, the more useful this dimension is for classification. The MI value is our importance score for each dimension, and is computed as:

$$I(\mathbf{x}_{:i}, \mathbf{y}) = H(\mathbf{y}) + H(\mathbf{x}_{:i}) - H(\mathbf{x}_{:i}, \mathbf{y}), \quad (\text{Eq. 5.2})$$

where H is the entropy of a random variable. Since $\mathbf{y}$ remains unchanged for different i , we just need to compute $H(\mathbf{x}_{:i}) - H(\mathbf{x}_{:i}, \mathbf{y})$.

We then sort the dimensions according to a decreasing order of their importance values. Then, if we want to reduce to D' dimensions, the top D' in the sorted list can be directly selected. When the compression ratio changes in feature compression methods, they must update codebooks, and update compressed features too. Both steps are very time consuming. In feature selection, we enjoy the simplicity when D' changes: just choose an updated subset of dimensions and no additional computations are needed at all.

5.1.3 Entropy calculation and further quantization

In Eq. 5.2 we need to compute the entropy values. A typical method to compute the differential entropy of a real-valued random variable ($\mathbf{x}_{:i}$) is to estimate its probability distribution function by kernel density estimation, which is again computationally too heavy in large scale problems. Rather, we use quantized discrete variables to compute entropy and further reduce storage costs.

We use two kinds of quantization:

¹A set of values in the set $\{1, \dots, C\}$ in a problem with C categories.

- **1-bit:** quantize a real number x into 2 discrete bins according to:

$$x \leftarrow \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases} \quad (\text{Eq. 5.3})$$

- **N-bins:** find the minimum and maximum value in a dimension, and uniformly quantize all values in this dimension into N bins. That is,

$$x \leftarrow \left\lfloor N \frac{x - \underline{v}}{\bar{v} - \underline{v} + \delta} \right\rfloor, \quad (\text{Eq. 5.4})$$

where $\underline{v}$ and $\bar{v}$ are the minimum and maximum value in one dimension, respectively; $\delta > 0$ is a small number, so that each value can be quantized into one of $\{0, \dots, N-1\}$ bins.

The discrete entropy is $H(x) = -\sum_j p_j \log_2(p_j)$ where p_j is the probability that x falls in the j -th bin.

A good quantization should maximize the mutual information between dimensions and the label vector. We show the histogram of mutual information values of different quantization setups in Fig. 5.2. We use the quantized empirical distributions to compute importance / usefulness scores. Note that this density estimation method just sequentially scan the training vectors once without additional computation, which makes it suitable for large scale dataset.

An observation common to all quantization methods is that most dimensions have small importance values. Except in 32-BINS, more than 90% dimensions have their MI values smaller than 0.1. This observation shows that most dimensions may not be useful for image classification. Feature selection, beyond having the virtue in reducing storage and CPU costs, also has the potential to improve classification accuracy by removing noisy or irrelevant dimensions.

Another important observation is that the 1-BIT quantization is better than 4-BINS and 8-BINS. Two points are worth noting. First, 1-BIT has 2 quantization bins. It is

Figure 5.2: Comparison of dimensions’ usefulness / importance values of different quantization methods on the Scene 15 dataset. This figure is best viewed in color.

different from 2-BINS: in 1-BIT the quantization threshold is fixed to 0 *a priori*, rather than determined empirically as in 2-BINS. Second, in general for the same x , a bigger N in N-BINS will lead to higher entropy value. For example, if x is uniform in $[0\ 1]$, 2-, 4- and 8-BINS quantization will have discrete entropy values 1, 2 and 3, respectively. Thus, this fact (1-BIT is better than 8-BINS) proves that proper quantization is essential. In particular, in FV and VLAD, 0 must be a quantization threshold.

Many feature compression methods use the 1-BIT quantization to store features compactly, *e.g.*, BPBC [23] and FV compression [57]. An explanation of its effectiveness is missing, though. From our quantization perspective, Fig. 5.2 quantitatively shows that

Figure 5.3: Histogram of values (after power normalization) in one FV dimension in the Scene 15 dataset.

1-BIT is good at preserving useful information. Furthermore, Fig. 5.3 qualitatively explains why 0 must be the threshold. After power normalization, 9 out of 10 randomly chosen dimensions follow bimodal distributions, where the two modes are separated exactly at the zero point. Thus, in order to keep the discriminative power of a feature, we must use 0 as a quantization threshold.

Based on the above results, we use 1-BIT to compute importance scores and choose a subset of dimensions. Finally, we further quantize the chosen dimensions using the 1-BIT quantization to further reduce storage cost. If D' dimensions are chosen, we just need $\frac{D'}{8}$ bytes to store an image. The first bin ($x \geq 0$) is stored as a value 1 in the bit, while the second bin ($x < 0$) is stored as a value 0.

5.1.4 Classification using quantized features

We then learn linear SVM classifiers on top of the quantized features. A simple table lookup trick can accelerate linear SVM training and testing.

Efficient linear SVM learners, such as stochastic gradient descent (SGD) [56] or dual coordinate descent (DCD) [84], spend most of their training and almost all testing time in the following two types of computations:

- **Update:** $\mathbf{w} \leftarrow \lambda_1 \mathbf{w} + \lambda_2 \mathbf{x}$.
- **Dot-product:** $\mathbf{w}^T \mathbf{x}$.

where $\mathbf{w}$ and $\mathbf{x}$ are the classifier boundary and one example, respectively.

When we use a selected and quantized feature vector $\hat{\mathbf{x}} \in \mathbb{R}^{D'/8}$, both steps can be made efficient by using a lookup table. Note that each byte within $\hat{\mathbf{x}}$ corresponds to 8 chosen dimensions, and we define a table $Z \in \mathbb{R}^{256} \times \mathbb{R}^8$:

$$Z = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \quad (\text{Eq. 5.5})$$

A byte i ($0 \leq i \leq 255$) corresponds to the i -th row in Z , which expands the byte into its 8 quantized dimensions.

We assume the length of $\mathbf{w}$ can be divided by 8, which is easy to satisfy. Then, when we perform either the update or the dot-product computation, we process 8 dimensions in a batch processing style: read one byte from $\hat{\mathbf{x}}$, then find its corresponding row in Z , finally compute the summation or dot-product of two small (length is 8) vectors.

5.2 A novel image recognition system: feature selection with HSM

We propose a novel system using HSM with selected features for large-scale image recognition. HSM can efficiently increase the spatial information of feature vectors. However, high-dimensional real-value feature vectors can cause severe memory problems during the

HSM SVM learning process in large scale recognition problems. With the feature selection technique, it is feasible to use HSM on selected features to improve their recognition accuracy.

The way to select features in HSM is a little different from that in Sec. 5.1.2. Unlike from all features in Sec. 5.1.2, in HSM, the same subset of features must be selected from the feature vector corresponding to each region. This is the requirement of HSM that the feature vectors of any two regions must be computed in the same way. For example, the same dimension from the feature vectors of any two regions must come from the same Gaussian in Fisher Vectors.

Specifically, feature selection in HSM can be done as the following. First, given the feature vectors of M regions in each image, all the M feature vectors are assigned the same image label. Then, feature selection is performed the same as Sec. 5.1.2 on all the (region) feature vectors from all images. The same subset of features in the feature vectors of M regions in each image is selected. Finally, HSM is computed on the selected feature vectors.

Compared to existing image recognition systems, the proposed FS+HSM large-scale image recognition system has several merits. First, it has very low memory cost. The selected feature vectors after quantization can save tens to hundreds times more memory than that of uncompressed real-value feature vectors. Second, it can reach comparative recognition accuracy to existing image recognition system using real-value feature vectors. With HSM, quantized features can improve their recognition accuracy comparative to that of real-value feature vectors.

5.3 Experiments for feature selection

We evaluate the proposed mutual information based importance sorting feature selection method on several large scale benchmarks. It is denoted as **MI**, which is compared

with PQ (product quantization) and BPBC. Since the datasets are large scale and time consuming to evaluate, we use PQ results from the literature when they are available for a dataset, otherwise we report PQ results from our own implementation.

We use the Fisher Vector to represent all images, following the setup in [63]. Only the mean and variance part in FV are used. The base visual descriptor is SIFT, which is reduced from 128 to 64 dimensional using PCA. The number of Gaussian components is 256. We use the spatial pyramid matching structure in [8] which extracts 8 spatial regions from an image. Its structure is: the whole image, three horizontal regions, and two by two split regions. The total number of dimensions in FV is $D = 64 \times 2 \times 256 \times 8 = 262144$. We revise the dual coordinate descent algorithm to learn a linear SVM classifier from our selected and quantized features or BPBC; and use the LIBLINEAR software package in our PQ experiments.

The following benchmark datasets are used:

- **VOC 2007** [13]. It has 20 object classes. Each image may contain more than one object. We use all the training and validation images (5K) for training and the testing images (5K) for testing.
- **ILSVRC 2010** [4]. It has 1000 classes and 1.2M training images. We use all provided training and testing images for training and testing, respectively.
- **SUN 397** [81]. It has 397 classes. In each class, we use 50 training images and 50 testing images.
- **Scene 15** [38]. It has 15 classes. In each class, 100 images are used for training, and the rest images are used for testing.

In PQ, we use the segment length $d = 8$, which has the overall best performance in [63] under different compression ratios and also used in BPBC [23]. We use k -means to generate codebooks. Then, we change the codebook size K to achieve different compression ratios in PQ.

In BPBC, we reshaped FV into a 128×2048 matrix, and learn bilinear projections to achieve different compression ratios. BPBC parameters need iterative updates, for which a maximum of 10 iterations is used in our experiments.

The results are averaged on 5 random train/test splits in Scene 15 and SUN 397. In VOC 2007, we use the predefined split, but run 5 times to get different GMM models and report the average mAP. For ILSVRC 2010, we run one time using the given split. All experiments are tested on a computer with Intel i7-3930K CPU and 32G main memory. All CPU cores are used during feature compression. In classifier learning and testing, only one core is used.

We first report the absolute classification performance (top 1 accuracy, top 5 accuracy, or mAP). Where space permits, we also report the loss of performance (delta between the performance obtained from uncompressed and compressed data) for easier comparisons. Finally, we compare the efficiency of feature selection or feature compression, classifier learning and testing for these three methods.

5.3.1 FS for image recognition

We test the proposed feature selection methods in image classification on four datasets.

5.3.1.1 VOC 2007

Mean average precisions (mAP) of various methods are shown in Table 5.1. We use the code from [8]² to generate FV with the same length as [63], thus it is fair to compare MI's performance with the PQ result from [63].

Under the same compression ratio, MI's mAP is higher than that of PQ on VOC 2007. The uncompressed result in our experiment and two cited PQ methods are close,

²The FV code is available at http://www.robots.ox.ac.uk/~vgg/software/enceval_toolkit/, version 1.1.

Table 5.1: Mean average precision (mAP) on VOC 2007. The loss of mAP relative to original dense feature (ratio 1) is also computed.

Method	Compression ratio	mAP (%)	Loss (%)
MI	1	58.57 ± 0.19	0
	32	60.09 ± 0.09	-1.52
	64	60.05 ± 0.16	-1.48
	128	58.97 ± 0.23	-0.40
	256	56.82 ± 0.49	1.75
	512	52.70 ± 0.44	5.87
	1024	46.52 ± 0.40	12.05
PQ [72]	1	58.8	0
	32($d = 6$)	58.2	0.6
	64($d = 8$)	56.6	2.2
	128($d = 8$)	54.0	4.8
	256($d = 8$)	50.3	8.5
PQ [63]	1	58.3	0
	32($d = 8$)	57.3	1.0
	64($d = 8$)	55.9	2.4
	64($d = 16$)	56.2	2.1

but the accuracy loss of MI is less than that of PQ. For example, when the ratio is 256, MI only loses 1.75% mAP, while PQ in [72] lost 8.5%.

In MI, a compression ratio 32 means that all dimensions are kept but quantized. Similarly, ratio 128 means that a quarter dimensions are selected and quantized. Ratio 32 provides the best discriminative ability in classification, which confirms yet another time that the 1-BIT quantization is effective in keeping useful information in features.

Another important fact is that when compression ratio is smaller than 128, MI’s mAP is higher than the uncompressed one. For example, ratio 64 (half dimensions used) has almost the same mAP as ratio 32 (all dimensions used). This observation corroborates that removing (a large portion of) noisy features will not hurt classification.

In contrast, PQ’s accuracy decreases quickly and monotonically when the compression ratio increases. The classification results of MI with more compression ratios are shown from 32 to 1024. When the compression ratio is 256, MI is comparable to that of PQ with compression ratio 32. Even with compression ratio 1024, MI’s mAP (46.52%) is still

Figure 5.4: Classifier learning time using the proposed (MI) compressed features on VOC 2007 under different compression ratios.

acceptable—remember that only 1024 bytes are needed to store the FV for an image at this compression level!

MI’s classifier training time on VOC 2007 is shown in Fig. 5.4. When the compression ratio changes from 32 to 1024 (doubling each time), the training time approximately halves each time. In other words, training time is roughly linearly proportional to storage size.

5.3.1.2 ILSVRC 2010

We report top-5 accuracy on the ILSVRC 2010 dataset in Table 5.2. Limited by our computer’s memory capacity, we need to start from compression ratio 64 in MI. As shown in Table 5.2, MI’s result is better than PQ’s [63] with the same FV setup and the same compression ratio. MI with compression ratio 128 has similar result as PQ at ratio 32.

If absolute accuracy rates are concerned, [56] reported that PQ with a well-tailored SGD classifier achieves 66.5% top-5 accuracy. When combining more visual descriptors like color descriptors [63, 65] and LBP [41], higher accuracy can be achieved on this

Table 5.2: Top-5 accuracy on the ILSVRC 2010 dataset.

Method	Compression ratio	Accuracy (%)
MI	64	61.06
	128	56.64
	256	50.15
PQ [63]	32($d = 8$)	56.2
	64($d = 8$)	54.2
	64($d = 16$)	54.9

Table 5.3: Top-1 accuracy on the SUN 397 dataset.

Method	Compression ratio	Accuracy (%)
dense FV [65]	1	43.3
multiple features [81]	1	38.0
spatial HOG [20]	1	26.8
MI	32	41.88±0.31
	64	42.05±0.36
	128	40.42±0.40
	256	37.36±0.34
PQ	32	42.72±0.45
	64	41.74±0.38
	128	40.13±0.33
	256	37.84±0.33

dataset. We conjecture that the proposed feature selection framework can also achieve better results than PQ in these richer representations.

5.3.1.3 SUN 397

We show accuracy of MI and PQ on the SUN 397 dataset in Table 5.3. Limited by our main memory size, we do not evaluate accuracy of the uncompressed dataset. Comparing with uncompressed FV [65], MI is inferior yet close to its accuracy when the compression is small, *e.g.*, 32 and 64. Note that when color descriptors are combined with SIFT, higher absolute accuracy is achieved [65] on this dataset.

Because we did not find any PQ compression results on this dataset, we implemented and evaluated our own PQ feature compression, and learned classifiers using a DCD linear SVM classifier. Comparing with PQ, MI is 0.8% inferior to PQ when compression ratio is

32, but better than or comparable with it when compression ratio gets larger. However, as we will show in Sec. 5.3.1.4, the required time in feature compression/selection, classifier learning and testing of MI is much less than that of PQ. Thus, overall MI is more efficient and effective than PQ.

One final note for Table 5.3 is that MI exhibits non-monotone accuracy again: ratio 64 accuracy is slightly higher than ratio 32.

5.3.1.4 Scene 15

On the Scene 15 dataset, we not only compare the accuracy of different methods. Because this dataset is small scale, we also perform additional experiments to compare the efficiency of the proposed MI method with PQ and BPBC, including feature compression/selection time, the classifier learning time, and the testing time.

The minimum compression ratio that MI and BPBC can reach is 32, and the maximum compression ratio of PQ can reach is 256 when group size $d = 8$. So, we restrict the compression ratio to the set $\{32, 64, 128, 256\}$. The class average accuracy is reported in Fig. 5.5. Overall all three methods have similar accuracies. PQ has an edge at ratio 32, while MI wins at all other ratios.

We further evaluate the feature generation time of these three methods, including extraction of visual descriptors (SIFT), FV generation, and feature compression/selection. Note that the first two parts of time is the same for all methods. The feature selection time in MI is negligible because it only involves keeping a small subset of values from the FV vector, while mutual information based selection of the subset is also very efficient. In PQ, when the compression ratio is small, *e.g.*, 32, feature compression (including the codebook generation time) takes about 30% of the total feature generation time in our experiments. When the compression ratio gets larger, PQ’s feature compression time decreases significantly till less than 5% of the whole feature generation time. BPBC

Figure 5.5: Classification accuracy of FV on Scene 15 using three methods: MI (proposed), PQ, and BPBC.

costs most time in the feature compression step. At least 50% of the entire feature generation time is spent on feature compression, which includes feature vector reshaping, learning of projection matrices, and feature quantization. Another issue is that BPBC is not suitable for learning two high dimensional matrices. In our experiments, the second dimension of the reshaped matrix is 2048, which makes the learning process very lengthy. Our evaluations find that the most efficient dimension range of the reshaped matrix is on the order of hundreds. That is, BPBC highly relies on the intrinsic structure of a feature vector, which is not as flexible as MI or PQ in dimensionality reduction for high dimensional vectors.

Finally, the classifier training and testing time of these three methods are shown in Fig. 5.6 and Fig. 5.7, respectively. The training and testing times of MI and BPBC are similar to each other at all compression levels, and both are much less than that of PQ. This is because MI and BPBC only uses the compressed/selected dimensions, while PQ has to decode the short compressed vector into a vector as long as the original features on-the-fly. A similar pattern is observed in Fig. 5.7 for testing time.

Figure 5.6: Classifier training time of FV on Scene 15 using three methods: MI (proposed), PQ, and BPBC.

Figure 5.7: Classifier testing time of FV on Scene 15 using three methods: MI (proposed), PQ, and BPBC.

5.3.2 FS+HSM on high-dimensional feature vectors

As we use Fisher Vectors (FVs) and their binary forms rather than histogram features, we only use HSM SVM with a linear base kernel, which is HSM-linear (cf. Table 4.1). The relationship matrix R in HSM (cf. Eq. 4.1) is learned on the 8-cell SPM structure used in FVs.

We first test FS+HSM on Scene 15. The selected features, with not only quantized binary forms but also real values, are tested with HSM. The results are shown in Table 5.4 and Table 5.5, respectively. Feature selection can improve both SPM and HSM. When

Table 5.4: Image classification accuracy of SPM and HSM using different portion of selected **real-value** features from FVs of Scene 15.

Portion of selected features	SPM (%)	HSM (%)
1	88.91	89.68
0.75	88.94	89.88
0.5	89.01	89.48
0.25	87.17	88.24
0.125	83.82	84.96

Table 5.5: Image classification accuracy of SPM and HSM using different portion of selected **binary** features from FVs of Scene 15.

Portion of selected features	SPM (%)	HSM (%)
1	87.40	88.21
0.75	87.91	88.71
0.5	87.54	88.21
0.25	86.40	86.87
0.125	82.58	83.15

a suitable portion (75%) of features are selected from the vectors of all spatial regions, SPM and HSM all / almost achieve the highest image recognition accuracy with the real-valued and binary features. Note that for the SPM experiments, the binary features are different from those in Fig. 5.5. Different features are selected in the two experiments.

HSM can improve SPM on the selected features. When 75% features are selected, HSM reaches 89.88% and 88.71% using real-value and binary features, respectively. The selected binary features cost much less memory than that of the uncompressed real-value feature vectors. Meanwhile, selected binary features with HSM can achieve similar accuracy (portion 0.75, 88.71%) as that of the uncompressed real value feature vectors (portion 1, 88.91%).

Next, we test HSM with selected binary features on Indoor 67 and SUN 397 to evaluate their performance on large-scale datasets. Their classification results are shown in Table 5.6 and Table 5.7, respectively. With quantized binary features, HSM outperforms SPM at different compression ratios. For Indoor 67 dataset, we compare quantized fea-

Table 5.6: Image classification accuracy of HSM using different portion of selected **binary** features from FVs of Indoor 67.

Portion of binary selected features	SPM (%)	HSM (%)
1	60.34	61.17
0.75	59.75	61.02
0.5	59.22	60.42
0.25	57.80	58.55
0.125	54.12	55.47
Uncompressed real-value features (SPM)	61.69%	

Table 5.7: Image classification accuracy of HSM using different portion of selected **binary** features from FVs of SUN 397.

Portion of binary selected features	SPM (%)	HSM (%)
1	42.01	43.09
0.75	42.33	43.12
0.5	40.56	41.23
0.25	37.21	37.98
0.125	34.87	35.31

tures with original real-value feature vectors. Quantized features with HSM have similar accuracy as that of uncompressed real-value features (61.17 vs. 61.69). For SUN 397 dataset, we did not evaluate on the real-value feature vectors, which is limited by the memory. HSM also achieves better accuracy than that of SPM on the quantized binary feature vectors. HSM’s higher accuracy (43.12%) on binary features is comparative to that of real-value features (43.3 % in Table 5.3).

5.4 Summary of the chapter

In this chapter, we propose that in order to reduce the dimensionality of powerful yet very high dimensional features (such as Fisher vector or VLAD), feature selection (*i.e.*, choosing a subset of dimensions from the original feature vector) is a better approach than existing feature compression methods (which perform linear projections on the original feature to extract new lower dimensional ones).

We first empirically show that collinearity almost does not exist in FV or VLAD, a fact

that also hints on the missing of multicollinearity (*i.e.*, strong linear relationships among more than 2 dimensions). This observation undermines the effectiveness of compression methods. We propose to use mutual information to choose a subset of dimensions for dimensionality reduction, which is a natural choice if we assume both collinearity and multicollinearity do not exist. Feature selection also removes noisy dimensions.

We use 1-BIT quantization to compute mutual information. We show that when quantizing dimensions, the zero value must be a threshold because it leads to bimodal distributions. Later, we also quantize the chosen dimensions into 1-BIT and design a linear SVM classifier to learn directly on the quantized values. Experiments show that the proposed feature selection and quantization method achieves better accuracy than feature compression methods, and is more efficient.

Finally, we evaluate the system using selected features with HSM. HSM is not feasible for high-dimensional real-value feature vectors in large scale recognition problem, which is solved by using selected binary features. HSM can improve the performance of selected binary features. The FS+HSM system achieves comparative recognition accuracy to that of real-value feature vectors, but incurs less cost on memory.

Chapter 6

Conclusions and Future Work

6.1 Summary of contributions

In this dissertation, to acquire an efficient and accurate representation for images in image recognition, three methods are proposed to improve three critical steps in the feature generation process: vector quantization, spatial matching, and the dimension reduction in large scale problems, which are illustrated in Fig. 6.1. We now summarize the contributions as follows:

- (i) We proposed exclusive vector quantization (ExVQ) for fast vector quantization to boost feature generation speed. When attaining similar classification / recognition accuracy, ExVQ is much faster than state-of-the-art approximate nearest neighbor search methods in vector quantization.
- (ii) We proposed hyper-spatial matching (HSM) to improve the spatial matching phase in feature generation. HSM is better than state-of-the-art spatial pyramid matching (SPM) methods because HSM considers all pairwise relationships among regions in two images, while SPM only considers relationships between spatially corresponding regions. We proved HSM is a Mercer kernel, when the relationship matrix is positive semidefinite. Based on this fact, we proposed two fast learning strategies for HSM SVM classifiers, which are hundreds of times faster than general purpose SVM

Figure 6.1: The contribution of this thesis.

solvers with HSM kernel. On several challenging benchmark datasets, HSM all showed better classification accuracy than SPM.

- (iii) We proposed to use feature selection to reduce the storage of high-dimensional feature vectors in large-scale image recognition problems. We first found that strong multicollinearity may not exist among features in high dimensional feature vectors. As a consequence we argue feature selection is a better choice than existing feature compression methods, which widely use linear projections. We proposed to use a supervised mutual information (MI) based feature selection method to reduce the storage of high dimensional feature vectors. Combined with the 1-BIT quantization, MI based feature selection shows better classification accuracy than state-of-the-art feature compression methods, when compared at the same memory size.

6.2 Future work

A number of improvements to the image representation can be made to further enhance the recognition efficiency and accuracy.

- (i) For dimension reduction, unsupervised feature selection methods can be considered. In many recognition problems, labels are missing. Thus, it is necessary to devise

efficient unsupervised methods which utilize the information of features themselves to compute the feature importance for feature selection.

- (ii) How to include more local information in spatial matching. HSM captures the relationship of spatially close regions in a global formulation for all images, which is better than SPM but still a bit coarse. It is worth investigating whether it is possible to compute the spatial relationship specific to each image pair, so that image similarity can be characterized finer.
- (iii) How to generate task specific image representations? The features used in this dissertation are all hand crafted, which are generated by pre-determined parameters, e.g., codebook size, SPM structure, etc. Given a specific image recognition task, how to learn the image representation from local visual descriptors or raw pixels, so that it has the best discriminance.
- (iv) How to devise a suitable image representation for other tasks in computer vision like image retrieval, object recognition? Unlike high dimensional feature vectors in image recognition, different areas have different requirements. For example, in image retrieval, an image representation is usually allocated with very small memory space like hundreds of bytes. How to generate an efficient image representation with limited memory space is a challenging problem for us to investigate.
- (v) How to generate efficient image representations which can be used across different problems? Currently, we use the same feature representation for one problem / dataset. It is important to explore what is shared in the features for different problems, so that the recognition model can be shared by different problems, which avoids learning separate models for individual new problems.

Publications

- **Yu Zhang**, Jianxin Wu, Jianfei Cai, “Compact Image Representation for Recognition and Retrieval: To Choose or to Compress?” in *IEEE Trans. on Image Processing*. (Under Review)
- **Yu Zhang**, Jianxin Wu, Jianfei Cai, and Weiyao Lin, “Flexible Image Similarity Computation Using Hyper-Spatial Matching,” in *IEEE Trans. on Image Processing*, 23(9), 2014: pp. 4112-4125.
- **Yu Zhang**, Jianxin Wu, and Jianfei Cai, “Compact Representation for Image Classification: To Choose or to Compress?” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2014.
- Jianxin Wu, **Yu Zhang**, Weiyao Lin, “Towards Good Practices for Action Video Encoding,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2014.
- **Yu Zhang**, Jianxin Wu, and Weiyao Lin, “Exclusive Visual Descriptor Quantization,” in *The 11th Asian Conference on Computer Vision*, 2012.

References

- [1] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28:2037–2041, 2006.
- [2] Relja Arandjelovic and Andrew Zisserman. All about VLAD. In *CVPR*, 2013.
- [3] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Proc. Advances in Neural Information Processing Systems*, 2006.
- [4] A. Berg, J. Deng, and F.-F. Li. ILSVRC 2010. <http://www.image-net.org/challenges/lsvrc/2010/index>.
- [5] Liefeng Bo and Cristian Sminchisescu. Efficient match kernels between sets of features for visual recognition. In *Proc. Advances in Neural Information Processing Systems*, 2009.
- [6] Y-Lan Boureau, Francis Bach, Yann LeCun, and Jean Ponce. Learning mid-level features for recognition. In *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, pages 2559–2566, 2010.
- [7] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27, 2011. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [8] Ken Chatfield, Victor Lempitsky, Andrea Vedaldi, and Andrew Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *Proc. British Machine Vision Conference*, 2010.

- [9] Dong Chen, Xudong Cao, Fang Wen, and Jian Sun. Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification. In *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2013.
- [10] Dan Ciresan, Ueli Meier, and Jurgen Schmidhuber. Multi-column deep neural networks for image classification. In *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2012.
- [11] Navneet Dalal, Bill Triggs, and Cordelia Schmid. Human detection using oriented histograms of flow and appearance. In *Proc. European Conf. Computer Vision*, 2006.
- [12] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Symposium on Computational Geometry*, pages 253–262, 2004.
- [13] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. <http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html>.
- [14] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, Aug 2008.
- [15] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training example: an incremental Bayesian approach tested on 101 object categories. In *CVPR 2004, Workshop on Generative-Model Based Vision*, 2004.
- [16] Francois Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research*, 5:1531–1555, 2004.
- [17] Learning Deep Features for Scene Recognition using Places Database. Bolei zhou and agata lapedriza and jianxiong xiao and antonio torralba and aude oliva. In *Proc. Advances in Neural Information Processing Systems*, 2014.
- [18] Glenn M. Fung and O.L. Mangasarian. A feature selection newton method for support vector machine classification. *Computational Optimization and Applications*, 28:185–202, 2004.

- [19] Shenghua Gao, Ivor Wai-Hung Tsang, Liang-Tien Chia, and Peilin Zhao. Local features are not lonelylaplacian sparse coding for image classification. In *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2010.
- [20] Tianshi Gao and Daphne Koller. Discriminative learning of relaxed hierarchy for large-scale visual recognition. In *Proc. IEEE Int'l Conf. on Computer Vision*, 2011.
- [21] Allen Gersho and Robert M. Gray. *Vector Quantization and Signal Compression*. Springer, 1991.
- [22] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2014.
- [23] Yunchao Gong, Sanjiv Kumar, Henry A. Rowley, and Svetlana Lazebnik. Learning binary codes for high-dimensional data using bilinear projections. In *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2013.
- [24] Yunchao Gong and Svetlana Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2011.
- [25] Michael C. Grant and Stephen Boyd. *CVX: Matlab software for disciplined convex programming*.
- [26] David Haussler. Convolution kernels on discrete structures. Technical report, Department of Computer Science, University of California at Santa Cruz, 1999.
- [27] Kaiming He, Fang Wen, and Jian Sun. K-means hashing: an affinity-preserving quantization method for learning binary compact codes. In *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2013.
- [28] G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

- [29] H. Jégou, Matthijs Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.
- [30] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Improving bag-of-features for large scale image search. *International Journal of Computer Vision*, 87(3):316–336, 2010.
- [31] Herve Jegou, Florent Perronnin, Matthijs Douze, Jorge Sanchez, Patrick Perez, and Cordelia Schmid. Aggregating local descriptors into a compact image representation. In *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, 2010.
- [32] Herve Jegou, Florent Perronnin, Matthijs Douze, Jorge Sanchez, Patrick Perez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 34(9):1704–1716, 2012.
- [33] Yangqing Jia, Chang Huang, and Trevor Darrell. Beyond spatial pyramids: Receptive field learning for pooled image features. In *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, pages 3370–3377, 2012.
- [34] Yuning Jiang, Junsong Yuan, and Gang Yu. Randomized spatial partition for scene recognition. In *Proc. European Conf. Computer Vision*, LNCS 7573, pages 730–743, 2012.
- [35] Mayank Juneja, Andrea Vedaldi, C. V. Jawahar, and Andrew Zisserman. Blocks that shout: Distinctive parts for scene classification. In *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, 2013.
- [36] Josip Krapac, Jakob Verbeek, and Frdric Jurie. Modeling spatial layout with fisher vectors for image categorization. In *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, 2012.
- [37] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. Advances in Neural Information Processing Systems*, 2012.

- [38] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, volume II, pages 2169–2178, 2006.
- [39] Vincent Lepetit and Pascal Fua. Keypoint recognition using randomized trees. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(9):1465–1479, 2006.
- [40] Li-Jia Li and Li Fei-Fei. What, where and who? Classifying events by scene and object recognition. In *Proc. IEEE Int'l Conf. on Computer Vision*, pages 1–8, 2007.
- [41] Yuanqing Lin, Fengjun Lv, Shenghuo Zhu, Ming Yang, Timothee Cour, and Kai Yu. Large-scale image classification: Fast feature extraction and SVM training. In *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2011.
- [42] Yue Lin, Rong Jin, Deng Cai, Shuicheng Yan, and Xuelong Li. Compressed hashing. In *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2013.
- [43] Lingqiao Liu, Lei Wang, and Xinwang Liu. In defense of soft-assignment coding. In *Proc. IEEE Int'l Conf. on Computer Vision*, 2011.
- [44] Ting Liu, Andrew W. Moore, Alexander Gray, and Ke Yang. An investigation of practical approximate nearest neighbor algorithms. In *Proc. Advances in Neural Information Processing Systems*, 2004.
- [45] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
- [46] Siwei Lyu. Mercer kernels for object recognition with local features. In *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2005.
- [47] Subhransu Maji, Alexander C. Berg, and Jitendra Malik. Efficient classification for additive kernel SVMs. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 35(1):66–77, 2013.
- [48] Marcin Marszalek and Cordelia Schmid. Constructing category hierarchies for visual recognition. In *Proc. European Conf. Computer Vision*, 2008.

- [49] Frank Moosmann, Eric Nowak, and Frederic Jurie. Randomized clustering forests for image classification. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 30(9):1632–1646, 2008.
- [50] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *Proc. 4th Int’l Conf. Computer Vision Theory and Applications*, volume 1, pages 331–340, 2009.
- [51] David Nistér and Henrik Stewénus. Scalable recognition with a vocabulary tree. In *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, volume 2, pages 2161–2168, 2006.
- [52] Mohammad Norouzi and David J. Fleet. Cartesian k-means. In *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, 2013.
- [53] AUDE OLIVA and ANTONIO TORRALBA. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.
- [54] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, 2005.
- [55] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, 2007.
- [56] Florent Perronnin, Zeynep Akata, Zaid Harchaoui, and Cordelia Schmid. Towards good practice in large-scale learning for image classification. In *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, pages 3482–3489, 2012.
- [57] Florent Perronnin, Yan Liu, Jorge Sánchez, and Hervé Poirier. Large-scale image retrieval with compressed fisher vectors. In *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, 2010.

- [58] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the Fisher kernel for large-scale image classification. In *Proc. European Conf. Computer Vision*, LNCS 6314, pages 143–156, 2010.
- [59] James Philbin, Ondřej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, 2008.
- [60] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, pages 413–420, 2009.
- [61] MarcAurelio Ranzato, Christopher Poultney, Sumit Chopra, and Yann LeCun. Efficient learning of sparse representations with an energy-based model. In *Proc. Advances in Neural Information Processing Systems*, 2006.
- [62] Olga Russakovsky, Yuanqing Lin, Kai Yu, and Li Fei-Fei. Object-centric spatial pooling for image classification. In *Proc. European Conf. Computer Vision*, LNCS 7573, pages 1–15, 2012.
- [63] Jorge Sánchez and Florent Perronnin. High-dimensional signature compression for large-scale image classification. In *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, pages 1665–1672, 2011.
- [64] Jorge Sánchez, Florent Perronnin, and Teófilo de Campos. Modeling the spatial layout of images beyond spatial pyramids. *Pattern Recognition Letters*, 33:2216–2223, 2012.
- [65] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector: Theory and practice. *International Journal of Computer Vision*, 105(3):222–245, 2013.
- [66] William Robson Schwartz, Aniruddha Kembhavi, David Harwood, and Larry S. Davis. Human detection using partial least squares analysis. In *Proc. IEEE Int’l Conf. on Computer Vision*, 2009.

- [67] Gaurav Sharma and Frederic Jurie. Learning discriminative spatial representation for image classification. In *Proc. British Machine Vision Conference*, 2011.
- [68] Gaurav Sharma, Frederic Jurie, and Cordelia Schmid. Discriminative spatial saliency for image classification. In *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, pages 3434–3441, 2012.
- [69] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep fisher networks for large-scale image classification. In *Proc. Advances in Neural Information Processing Systems*, 2013.
- [70] Mingkui Tan, Li Wang, and Ivor W. Tsang. Learning sparse svm for feature selection on very high dimensional datasets. In *Proc. Int’l Conf. on Machine Learning*, 2010.
- [71] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. <http://www.vlfeat.org/>, 2008.
- [72] A. Vedaldi and A. Zisserman. Sparse kernel approximations for efficient classification and detection. In *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, 2012.
- [73] Andrea Vedaldi and Andrew Zisserman. Efficient additive kernels via explicit feature maps. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 34(3):480–492, 2012.
- [74] Christian Wallraven and Barbara Caputo. Recognition with local features: the kernel recipe. In *Proc. IEEE Int’l Conf. on Computer Vision*, 2003.
- [75] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, pages 3360–3367, 2010.
- [76] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Proc. Advances in Neural Information Processing Systems*, pages 1753–1760, 2009.
- [77] Jianxin Wu. Power Mean SVM for large scale visual classification. In *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, pages 2344–2351, 2012.

- [78] Jianxin Wu and James M. Rehg. CENTRIST: A visual descriptor for scene categorization. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 33(8):1489–1501, 2011.
- [79] Jianxin Wu and James M. Rehg. Where am i: Place instance and category recognition using spatial PACT. In *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, 2014.
- [80] Jianxin Wu, Yu Zhang, and Weiyao Lin. Towards good practices for action video encoding. In *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, 2014.
- [81] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, pages 3485–3492, 2010.
- [82] Jian-Bo Yang and Ivor W. Tsang. Hierarchical maximum margin learning for multi-class classification. In *Conf. on Uncertainty in Artificial Intelligence*, 2011.
- [83] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, 2009.
- [84] Guo-Xun Yuan, Chia-Hua Ho, and Chih-Jen Lin. Recent advances of large-scale linear classification. *Proceedings of the IEEE*, 100(9):2584–2603, 2012.
- [85] Ning Zhang, Jeff Donahue, Ross Girshick, and Trevor Darrell. Part-based r-cnns for fine-grained category detection. In *Proc. European Conf. Computer Vision*, 2014.
- [86] X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. In *Proc. European Conf. Computer Vision*, 2010.