AN ABSTRACT OF THE THESIS OF

Junyuan Lin for the degree of Master of Science in Computer Science presented on February 18, 2013.

Title: A Study of Methods for Fine-grained Object Classification of Arthropod Specimens

Abstract approved: _____

Thomas G. Dietterich

Object categorization is one of the fundamental topics in computer vision research. Most current work in object categorization aims to discriminate among generic object classes with gross differences. However, many applications require much finer distinctions. This thesis focuses on the design, evaluation and analysis of learning algorithms for fine-grained object classification. The contributions of the thesis are three-fold. First, we introduce two databases of high-resolution images of arthropod specimens we collected to promote the development of highly accurate fine-grained recognition methods. Second, we give a literature review on the development of *Bag-of-words* (BOW) approaches to image classification and present the stacked evidence tree approach we developed for the fine-grained classification task. We draw connections and analyze differences between those two genres of approaches, which leads to a better understanding about the design of image classification approaches. Third, benchmark results on our two datasets are presented. We further analyze the influence of two important variables on the performance of fine-grained classification. The experiments corroborate our hypotheses that a) high resolution images and b) more aggressive information extraction, such as finer descriptor encoding with large dictionaries or classifiers based on raw descriptors, is required to achieve good fine-grained categorization accuracy.

A Study of Methods for Fine-grained Object Classification of Arthropod Specimens

by

Junyuan Lin

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented February 18, 2013
Commencement June 2013

Master of Science thesis of Junyuan Lin presented on February 18, 2013.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Junyuan Lin, Author

ACKNOWLEDGEMENTS

I would like to acknowledge my major advisor, Professor Thomas Dietterich, for his continuous guidance and support. His extensive knowledge and creative way of thinking have been a great help to me. He has been incredibly supportive and patient with me during my graduate study.

My thanks go out to former BUG-ID group members, Gonzalo Martinez-Munoz, Natalia Larios and Wei Zhang. Their kindly help made it smooth for me to take over the legacy codes and conduct my research.

Finally, I would also like to thank my parents. All of this would not be possible without their love and support.

This thesis is based in part upon work supported by the National Science Foundation under Grant Nos. 0705765 and 0832804. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

TABLE OF CONTENTS

Page

1 Introduction 1

1.1 Background and Motivation . . . 1

1.2 Datasets . . . 2

1.3 Thesis Outline . . . 3

2 Literature Review of Bag-of-Words Approaches 4

2.1 Overview . . . 4

2.2 Notation . . . 7

2.3 A Baseline Bag-of-Words Approach . . . 9

2.4 Dictionary Construction . . . 11

2.4.1 Unsupervised Dictionary . . . 11

2.4.2 Supervision at the Descriptor Level . . . 13

2.4.3 Supervision at the Image Level . . . 15

2.5 Descriptor Encoding . . . 16

2.5.1 Visual Word Ambiguity . . . 17

2.5.2 Sparse Coding . . . 19

2.6 Feature Pooling . . . 22

2.6.1 Basic Pooling Function . . . 23

2.6.2 Spatial Pooling . . . 25

2.6.3 Local Neighborhood Pooling . . . 26

2.7 Classifier Design . . . 27

2.7.1 Generative models . . . 27

2.7.2 Discriminative models . . . 29

2.8 Summary . . . 32

3 Stacked Evidence Trees 35

3.1 Overview . . . 35

3.2 Classification Architecture . . . 35

3.3 Relationship to BOW Approaches . . . 38

3.4 Hypotheses Concerning the Requirements for Successful Fine-grained Categorization . . . 39

TABLE OF CONTENTS (Continued)

Page

4 Benchmarks and Experimental Tests of Our Hypotheses 41

4.1 Benchmarks 41

4.2 Tests of the Image Resolution Hypothesis 43

4.2.1 The Effect of Reduced Resolution on SIFT Descriptors 43

4.2.2 The Effect of Reduced Image Resolution on Classification Accuracy 44

4.3 Tests of the Information Extraction Hypothesis 45

4.3.1 Varying the Size of Dictionaries 46

4.3.2 Varying the Difficulty of Classification Problems 47

4.4 Discussion 50

5 Conclusions and Future Directions 52

Bibliography 52

Appendices 62

LIST OF FIGURES

Figure Page

2.1 The pipeline of *Bag-of-Words* framework for image classification 5

2.2 Naive Bayes model 28

2.3 Supervised Latent Dirichlet Allocation model 29

2.4 The illustration of super-vector representation. Different clusters of visually similar descriptors are mapped to separate local code blocks. Within each local code block, descriptors are represented by a similar linear combination of codes, this gives a rich and pooling-robust representation. . . . 34

3.1 The overall classification architecture of stacked evidence trees. Descriptors are directly drop through the random forests without computing a visual dictionary. Each leaf node in the forest stores a histogram of class labels of the training examples. The class histogram at all such leaves are accumulated to produce a single image level class histogram which is then fed to a stacked classifier to make the final prediction. 36

3.2 The comparison between the *BOW* model and the stacked evidence trees model. For the *BOW* model, the pooling happens before classification, while the evidence trees model first makes predictions on the individual descriptors and then pools the class distributions before the second image level classification. 38

4.1 Performance under resolution reduction on STONEFLY9 dataset 45

4.2 Performance under resolution reduction on EPT54 dataset. 45

4.3 Performance of *LLC* under different dictionary sizes on the Caltech-101 dataset. The error bars are computed as one standard deviation of accuracy over 10-fold cross validation. 46

4.4 Performance of *LLC* under different dictionary sizes on (a) STONEFLY9 and (b) EPT54. The error bars are computed as one standard deviation of accuracy over 3-fold cross validation. 47

4.5 Performance under different dictionary sizes at genus level with phylogenetic tree distance of 2. 49

LIST OF FIGURES (Continued)

Figure Page

4.6 Performance under different dictionary sizes at family level with phylogenetic tree distance of 4. 49

4.7 Performance under different dictionary sizes at order level with phylogenetic tree distance of 6. 50

LIST OF TABLES

Table Page

2.1 Symbol and Notation Used in the Paper 7

2.2 The train and test time complexity comparison of different classifiers. N is the number of images; d is the dimensionality of the feature vectors, T is the depth of the decision trees, and k is the number of nearest neighbors. 31

4.1 Performance on the Stonefly9 dataset. The top row shows the results using only SIFT descriptors on a regularly sampled grid, and the bottom rows shows the results of combining SIFT descriptors applied to patches found by four different detectors. The error bars are computed as one standard deviation of accuracy over 3-fold cross validation. 42

4.2 Performance on the EPT54 dataset. The error bars are computed as one standard deviation of accuracy over 3-fold cross validation. 42

4.3 Classification accuracy of four Detector/descriptor combinations on STONEFLY9. The four detectors are the PCBR detector[17], the regular grid, the Kadir-Brady salient region detector [42], and the Hessian affine detector [60], We also show the result of combining all four detectors. The error bars are computed as one standard deviation of accuracy over 3-fold cross validation. 42

4.4 A quantitative measure of the image downsizing effect on SIFT descriptors on STONEFLY9. The median city block distance of the SIFT descriptors at the same location in different scales is computed. 44

4.5 The number of training examples for each class (at genus level) and its taxonomic information in the pairwise difficulty controlled experiments. . 48

LIST OF APPENDIX FIGURES

Figure Page

1 Visualization of confusion matrix of EPT54 dataset. The figure shows a hierarchical clustering tree based on the class-wise distances. The distances are computed as the reciprocal of the off-diagonal values of the confusion matrix. The table at left shows the hierarchy of biological classification for each category in the dataset. The three orders in the datasets are color coded: Ephemeroptera (Mayflies) in red, Plecoptera (Stoneflies) in blue, and Trichoptera (Caddisflies) in green. 63

Chapter 1: Introduction

1.1 Background and Motivation

Image classification is one of the fundamental topics in computer vision. The problem has drawn considerable research attention. Over the last decade with the advance of machine learning techniques, significant progress in this area has greatly improved state-of-art performance on challenging benchmark datasets such as scene-15 [76], Caltech-101 [25], Caltech-256 [36] and the Pascal Visual Object Classes dataset [23]. All of those datasets are of generic object classes with gross differences. While this addresses important fundamental questions in computer vision, it does not solve any pressing application problem. In real applications such as biomonitoring, identifying much finer distinctions between classes is desired. Such fine-grained object classification presents new challenges to the computer vision community. The small intra-class differences and large inter-class differences among fine-grained object categories are so subtle that even human experts cannot categorize them easily.

This thesis focuses on the design, evaluation and analysis of learning algorithms for fine-grained object classification. First, we introduce two databases of high-resolution images of arthropod specimens. Over the past eight years, we have collected, photographed and manually labeled over fifty taxa (species or genus) of fresh water arthropod specimens to produce two image databases: STONEFLY9 and EPT54. Those two datasets cover common fresh water stream macroinvertebrates in the Pacific Northwest. These organisms are a robust indicator of stream health and water quality. The system we developed based on these databases offers a practical solution that largely automates the tedious work of classifying those specimens for the purpose of biomonitoring. In the broader view, we hope the publication of our databases will promote further development of highly-accurate fine-grained recognition methods in computer vision.

From the machine learning point of view, the object categorization problem can be formulated in a weakly supervised setting, where the only supervised information in training is the object class label associated with an entire image. The goal of object

categorization is to predict the object class that is present in the image. The general approach to this problem is to transform an original image into a bag of region descriptor vectors. This reduces the object classification problem to a multiple-instance classification problem. Various learning algorithms have been proposed in the computer vision community. Among them, the *bag-of-words* (BOW) model [15] is the most popular one, and it has achieved satisfying performance on various benchmark datasets.

In our previous work [47], we applied the standard *bag-of-words* approach to our databases. On STONEFLY9, it achieves an error rate of 16.1% , which is not accurate enough for our application. To achieve better performance, we developed a novel algorithm called *stacked evidence trees* [59] that achieves high performance on both datasets. Recently, there have been great developments in improving the standard *bag-of-words* approach. In this thesis, we will show that of some of these state-of-art approaches also provide outstanding results on our databases. We give a literature review of the development of *Bag-of-words* (BOW) approaches to object classification and present the stacked evidence tree approach we developed for the fine-grained classification task. We draw connections and analyze differences between those two genres of approaches. We believe that this analysis leads to a better understanding of object classification approaches.

Finally, in the experiment chapter, benchmark results are presented on our two datasets. We further analyze the influence of two important variables on the performance of fine-grained classification. The experiments corroborate two hypotheses, namely that a) high resolution images and b) more aggressive information extraction, such as finer descriptor encoding with large dictionaries or classifiers based on raw descriptors, is required to achieve good performance in fine-grained categorization.

1.2 Datasets

Stoneflies inhabit freshwater streams and are known to be a sensitive and robust indicator of stream health and water quality. While it is easy to collect specimens, a high degree of expertise and a large amount of time are required to manually classify specimens to the level of species or genus.

The STONEFLY9 database we created consists of 3826 images obtained by imaging 773 stonefly larvae specimens. The dataset contains 9 taxa (species or genera) of stoneflies. Each taxon is common in streams in the Pacific Northwest. The specimens range

in size from 0.1 to 10mm.

In order to provide a better benchmark for computer vision research and deliver a more practical system for biomonitoring, we collected a larger and more general EPT54 database. The EPTs: Ephemeroptera (Mayflies), Plecoptera (Stoneflies), and Trichoptera (Caddisflies) are the most commonly-used groups of organisms for biomonitoring of freshwater streams. Species-rich EPT assemblages are a robust indication of clean water. The EPT54 dataset consists of 10,173 images of 3394 specimens belonging to 54 taxa of EPTs.

Each specimen was photographed multiple times using a semi-automated apparatus under fixed lighting, focus, and exposure conditions. The images are captured at high resolution (2560 x 1920 pixels) in RAW format. For each database, descriptors have been extracted and can be downloaded from our website in addition to the images themselves.

1.3 Thesis Outline

The rest of the thesis is organized as follows. In chapter 2, we give a literature review on the development of *Bag-of-words* (BOW) approaches to object classification and present the alternative stacked evidence tree approach. Then we draw connections and analyze differences between these two approaches and propose two hypotheses that explain what is needed to achieve acceptable performance in fine-grained categorization.

In chapter 3, we present benchmark results on both datasets. We further analyze the influence of two important variables — the image resolution and the dictionary size — on the performance of the methods. The experiments corroborate our hypotheses that a) high resolution images and b) more aggressive information extraction are required to achieve good performance of fine-grained categorization.

Chapter 4 concludes the thesis with additional discussion and future directions.

Chapter 2: Literature Review of Bag-of-Words Approaches

2.1 Overview

The *Bag-of-words* (BOW) methods have enjoyed great popularity in image classification. In this chapter, we present a broad survey of recent approaches under the *Bag-of-words* framework.

First we describe the general pipeline of the *Bag-of-words* framework. We then instantiate the pipeline to define a standard configuration for the BOW approach.

Based on this baseline, we describe various alternatives in four major dimensions of the design space, specifically dictionary construction, descriptor encoding, feature pooling, and classifier design. We analyze the strengths and weaknesses of different approaches and conclude general guidelines for the design of *Bag-of-words* image classification systems.

Inspired by text classification [40], and first introduced in [15] [81], the *bag-of-words* model dominates modern approaches to image classification. The *bag-of-words* model treats an image as a set of visual features extracted from local image patches, encodes each of them using a dictionary of visual words, and then aggregates them to form a compact histogram representation of the image. Following the BOW framework, the pipeline of most *state-of-the-art* image classification systems consists of the following steps as shown in figure 2.1.

(i) Local image patches are extracted by interest point detectors or densely sampled on a regular grid.

(ii) Each patch is described by a descriptor that is invariant to local transformation and illumination, such as SIFT [54] or HOG [16].

(iii) A visual dictionary is built based on the descriptors in the training images.

(iv) All descriptors are encoded into "codebook space" using the visual dictionary.

(v) A fixed-length feature vector representing the entire image is constructed by pooling the *bag-of-descriptors* in the codebook space.

(vi) The feature vectors are used as input to a classifier, which makes the final prediction of the image class label.

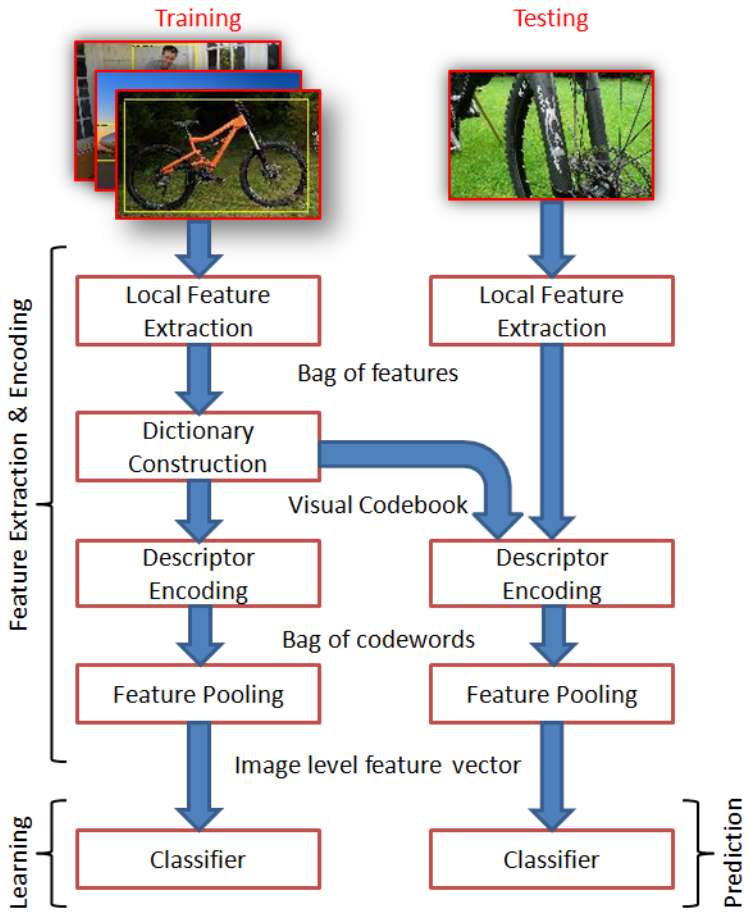

Figure 2.1: The pipeline of *Bag-of-Words* framework for image classification

The BOW framework in the image domain bears some similarity to the document domain. As in text classification, the goal of the BOW representation is to represent each document as a fixed-length feature vector that can be used to train a classifier in step (vi). The BOW representation of the image is a compact representation of the semantics of the image, and it is robust to variations prevailing in natural images, such as transformations (in-plane rotation, translation, scale), occlusions, and background

clutter. It also shares the same shortcomings. One major criticism is that the BOW representation discards all the spatial relationships among the local patches; therefore it fails to model the spatial relations among image patches.

However there are many differences from text analysis as well. One of the key difference is that unlike the document domain, there is no natural definition of discrete codewords in the image domain. Because of that, we need to explicitly design local detectors and descriptors to extract informative patches from an image in steps (i) and (ii). Local descriptors usually lie in a continuous high dimensional space. However an image category often "lives" in a much lower-dimensional manifold that can be described as a set of tight clusters of distinctive features in this feature space. Therefore in step (iii), a visual dictionary is constructed in the hope of discovering and describing those meaningful clusters. Given a dictionary, in step (iv) the encoding process maps descriptor vectors in continuous space to the discrete codebook space. This can be more sophisticated than the typical one-to-one word matching in the document domain. Different coding methods have be explored. Likewise, since the feature encodings are not limited to binary indicator vectors, the pooling process in step (v) can be more complicated than computing a histogram of word counts.

As pointed out by [64], a large number of local patches is essential to good performance. Experience has shown that interest point detectors such as [17], [60], [42] often fail to extract enough discriminative patches over all images. Consequently, the research community has reached a consensus that using densely-sampled local patches instead of an interest point detector in step (i) is better, because it is more robust, repeatable, and doesn't miss important patches . Various descriptors can be used to describe local patches. In any case, after step (ii), the image is represented as a *bag-of-descriptors*.

In this chapter, we survey a wide range of information processing approaches that transform the *bag-of-descriptor* representation into a single feature vector representation of the whole image and the subsequent classifier design step. We divide the various dimensions of the design space into four major steps (iii)$\sim$(vi), namely *dictionary construction*, *descriptor encoding*, *feature pooling*, and *classifier design*. We first give a standard configuration of the *bag-of-words* method as our baseline. Then for each of the above four aspects, we describe the alternatives that have been explored by different research groups and give an analysis of their strengths and weaknesses.

2.2 Notation

Table 2.1 defines the notation used in the chapter.

Definition	**Symbol/Notation**
Number of images	N
Number of codewords in the dictionary	K
Number of local descriptors per image	M
Dimension of a local descriptor	d
Number of spatial bins for pyramid representation	S
Dimension of the image-level feature vector	H
Local descriptor vector	$\mathbf{x} \in \mathcal{R}^d$
Individual visual codeword	$\mathbf{w}$
Local patch location	$\mathbf{s} \in \mathcal{R}^2$
Image as bag-of-descriptors	$I = \{(\mathbf{x}_j, \mathbf{s}_j)\}_{j=1}^M$
Image as bag-of-descriptors in matrix form	$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M] \in \mathcal{R}^{d \times M}$
Training set for feature construction	$\{(I_i, Y_i)\}_{i=1}^N$
Training pool of bag-of-descriptors	$\mathcal{X} = \{\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_N\}$
Class label for training images	$\mathcal{Y} = \{Y_1, Y_2, ..., Y_N\}$
Visual Dictionary	$\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_K] \in \mathcal{R}^{d \times K}$
Image as encoded bag-of-descriptors	$\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_M] \in \mathcal{R}^{K \times M}$
Image as encoded feature vector	$\mathbf{h} \in \mathcal{R}^H$
Training set for classification	$\{(\mathbf{h}_i, Y_i)\}_{i=1}^N$

Table 2.1: Symbol and Notation Used in the Paper

After local descriptor extraction, an image can be represented by a *bag-of-descriptors* $I = \{(\mathbf{x}_j, \mathbf{s}_j)\}_{j=1}^M$.

The *dictionary construction*, *descriptor encoding* and *feature pooling* steps can be viewed as a feature construction process that transforms the *bag-of-descriptors* representation into a fixed-length feature vector representation of the whole image used in *classifier design*. We define the process and input and output of all four steps formally using the notation defined in table 2.1.

Dictionary Construction. Given a set of training images $\{(I_i, Y_i)\}_{i=1}^N$, we denote

the pool of *bag-of-descriptors* as $\mathcal{X} = \{\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_N\}$ and the class label set as $\mathcal{Y} = \{Y_1, Y_2, ..., Y_N\}$. The goal of dictionary construction is to learn a set of visual codewords $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_K] \in \mathcal{R}^{d \times K}$.

Dictionary construction can be formulated in both supervised and unsupervised ways.

In the unsupervised setting, we seek to construct a general dictionary that provides a good representation of the local image patches in the domain:

$$\mathcal{X} \longrightarrow \mathbf{D}.$$

In the supervised setting, the image label is also available, so the goal is to construct a dictionary that encodes only the information needed to perform the classification task:

$$(\mathcal{X}, \mathcal{Y}) \longrightarrow \mathbf{D}.$$

Descriptor Encoding. Given the visual dictionary $\mathbf{D}$ for each image, the descriptor encoding step encodes the *bag-of-descriptors* in the codebook space.

$$\mathbf{X}_i \xrightarrow{\mathbf{D}} \mathbf{C}_i,$$

where $\mathbf{C}_i = [\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_{M_i}] \in \mathcal{R}^{K \times M_i}$ is the encoded *bag-of-descriptors* in the codebook space.

Note that, more often than not, dictionary construction and descriptor encoding are intertwined. Many methods obtain them simultaneously by minimizing $\mathbf{D}$ and $\mathbf{C}$ over a joint objective function. But different methods have different focuses. We classify them by which step they primarily optimize.

Image Pooling. A feature vector representing the entire image is constructed by pooling the encoded *bag-of-descriptors*. We define a pooling function g:

$$\mathbf{C}_i \xrightarrow{g} \mathbf{h}_i, \quad \text{where } \mathbf{h} \in \mathcal{R}^H$$

During testing, given a testing image I_i and its *bag-of-descriptors* representation after step (ii), since the dictionary has already learned, only *descriptor encoding* and *feature pooling* are needed to map the *bag-of-descriptors* $\mathbf{X}_i$ to $\mathbf{h_i}$. We denote the combined

process of those two steps as the feature construction function ϕ:

$$\mathbf{h_i} = \phi(\mathbf{X}_i, \mathbf{D})$$

Classifier Design. After the above feature construction steps, every image can be represented as an encoded fixed-length feature vector $\mathbf{h}$.

We denote the training examples for the classifier as $\{(\mathbf{h}_i, Y_i)\}_{i=1}^{N}$. A classifier T is trained on the set of training examples:

$$\{(\mathbf{h}_i, Y_i)\}_{i=1}^{N} \longrightarrow T.$$

In testing, after applying the feature construction function ϕ, the test feature vector $\mathbf{h}_i$ is fed to the classifier which outputs a predicted label Y_{test}:

$$\mathbf{h}_i \xrightarrow{T} Y_{test}.$$

2.3 A Baseline Bag-of-Words Approach

In this section, we instantiate the *bag-of-words* pipeline to define a baseline approach. The baseline configuration of the *bag-of-words* approach was first proposed by [15]. It is a simple adaptation from the text domain. The major differences are the dictionary learning and vector quantization steps.

After local patch extraction, a visual dictionary is constructed using k-means clustering on patch descriptors. The objective function minimizing the within-cluster scatter is defined as

$$\min_{\mu_k, \mathbf{c}_j} J = \sum_{j=1}^{M} \sum_{k=1}^{K} c_{jk} \|\mathbf{x}_j - \mu_k\|_2^2.$$

The algorithm iteratively reassigns points to their nearest cluster centers and updates of the cluster centers, until the iteration converges to a local minimum.

k-means clustering results in a partitioning of the high dimensional descriptor space into Voronoi cells. This naturally leads to a hard vector quantization in the descriptor encoding step.

In hard vector quantization, each descriptor is mapped to the nearest codeword in

the dictionary. This outputs a 1-of-K of binary indicator vector $\mathbf{c_j} \in \{0,1\}^K$ for each descriptor. The hard vector quantization is defined as

$$\mathbf{c}_j \in \{0,1\}^K, \quad c_{jk} = 1 \quad \texttt{iff} \quad k = \underset{1 \le k \le K}{\operatorname{argmin}} \|\mathbf{x}_j - \mu_k\|_2^2,$$

This binary indicator vector is exactly the same representation as in the text domain. Therefore, an average pooling is adopted. A histogram $\mathbf{h}$ is generated by taking the vector sum of the bag of binary codes and normalizing by the number of local descriptors the image:

$$\mathbf{h} = \frac{1}{M} \sum_{j=1}^{M} \mathbf{c}_j.$$

So the image-level feature vector is a normalized histogram of individual codewords that appear in the image.

Finally, a linear *support vector machine (SVM)* classifier is trained over the image-level feature vectors to make predictions.

Although this baseline configuration gives satisfying results, there are several drawbacks as well:

(a) In dictionary construction, the k-means clustering algorithm only converges to a local minimum and does not guarantee to find the most representative codewords. In addition, the dictionary size K is chosen empirically. When a large dictionary is needed, the speed of the k-means algorithm becomes a major concern.

(b) In descriptor encoding, the hard vector quantization loses a lot of information. For example, a 128 dimensional SIFT descriptor when mapped to a dictionary of 3000 codewords retains at most 12 bits of information from the original SIFT descriptor.

(c) In feature pooling, the simple average pooling over the entire image completely neglects spatial information.

(d) The above feature construction steps are completely unsupervised, while the classifier is trained with supervision. Those two steps employ different objectives. In this sense, such feature representation is likely to be suboptimal for the image classification task.

(e) In classifier design, different classifiers can be considered. For example, a kernel SVM could give better performance. Other popular classifiers might do well on the

BOW representation as well.

A large amount of research effort has been put into improving the baseline model to overcome or explore the aforementioned shortcomings. In the following sections, we summarize the advances in the literature along four dimensions of the design space.

2.4 Dictionary Construction

The construction of a good dictionary that discovers informative visual codewords to capture image content is critical to the subsequent encoding and classification steps. In the baseline, k-means clustering is used. Numerous studies have been devoted to learning a better visual dictionary. We categorize different approaches into two classes: unsupervised dictionary learning and supervised dictionary learning.

2.4.1 Unsupervised Dictionary

In the unsupervised setting, we seek to construct a general dictionary that is well representative of local patches in the training images. The objectives of unsupervised approaches vary. In general, the unsupervised dictionary learning approaches aim to improve over k-means clustering in two main aspects: (a) a more robust and uniform coverage of the local descriptors in the high dimension descriptor space, and (b) a more efficient algorithm for generating dictionaries of large size.

Robust and Uniform Clustering

Jurie and Triggs [41] analyse the distribution of densely-sampled local descriptors in the high dimension descriptor space. They show that the distribution is highly non-uniform due to the extremely unbalanced occurrence of visual patterns in natural images. For classification, the most informative local patches those of intermediate frequency, because the most frequent patches are typically generic image structures such as edges or corners with low discriminative value, while rare patches do not generalize well.

From the above analysis, they argue that k-means clustering is suboptimal, because it tends to over-partition the dense regions while under-partitioning the sparse ones due to the updating of the cluster centers to denser regions in the M step of the iterative algorithm. Also k-means is not robust to outliers (points far from any cluster centers). To overcome those drawbacks, they propose a radius-based clustering algorithm. It

sequentially finds a high density region by applying the mean-shift mode estimator [32]. All descriptor vectors that are within a fixed radius r of the cluster center are assigned to the cluster. Those assigned vectors are eliminated from the dictionary training set. The process is repeated until the desired number of clusters has been found. The radius r determines the similarity threshold between visual words.

Leibe *et al.* [51] adopt an agglomerative clustering approach to creating efficient codebook. Agglomerative clustering is suitable for non-uniform data distributions and is robust to outliers. The number of clusters can be determined meaningfully by specifying a similarity threshold. However, the complete bottom-up agglomerative clustering is very inefficient for large datesets in dictionary learning. Their solution is a combined partitional-agglomerative method. k-means clustering is first used to coarsely partition the decriptor vector space. Agglomerative clustering is applied within each partition to produce sets of cluster centers. A final agglomerative clustering step is applied again on all cluster centers. A ball tree data structure can be easily computed from the agglomerative clustering structure for fast matching.

Tree Structured Clustering

Dictionaries of large size are favored based on the reasoning that a fine partition of the descriptor space provides more discriminative information for classification. In general, the vocabulary size should be large enough to distinguish relevant differences in objects, but not so large as to distinguish irrelevant variations such as noise.

When the dictionary size K is huge, both the dictionary construction and the nearest neighbor based vector encoding of k-means become inefficient. Some approximate nearest neighbor search algorithm such as locality sensitive hashing [34] can be used in encoding [43]. However, the $O(NKd)$ computational complexity of k-means makes the algorithm inefficient for constructing the dictionary when k is large.

Tree structured space partitioning is more efficient in both construction and encoding. Nevertheless for high dimensional descriptor vectors in $\mathcal{R}^d$, the widely used space-partitioning *k-d tree* suffers from the curse of dimensionality. In order to reduce the cell radius by half, d levels of a *k-d tree* should be built, which requires 2^d data points. Therefore, the clustering-based method is generally preferred over space partitioning.

Tuytelaars *et al.* [87] observe that due to the highly non-uniform distribution of the descriptor space [41], most of the bins in the *k-d tree* partition will be empty. They propose a uniform partition of the descriptor space and use hashing techniques to store only

non-empty bins. Although, this significantly reduces the number of bins, the absolute vector length still remains huge (typically 10^7).

Nister *et al.* [63] propose a hierarchical k-means clustering approach. It builds a vocabulary tree. At each level of the tree, a k-means clustering with k equal to a branching factor B ($B \ll K$) is performed. Every subtree is built recursively within each cluster, until the desired dictionary size is reached. The tree structure offers logarithmic-time encoding. And the dictionary construction complexity is $O(NB \log_B Kd)$.

Moosmann *et al.* [61] propose randomized clustering forests. The approach combines *random forests* [13, 33] and *clustering trees* [6, 53]. A sufficiently diversified ensemble of random trees is able to explore different partitions of the high dimensional descriptor space. The time complexity of building a random clustering forest is $O(N \log k\sqrt{d}|T|)$, where $|T|$ is the number of trees. Supervision can be easily incorporated in the construction of the trees as shown in the following section.

2.4.2 Supervision at the Descriptor Level

In the supervised setting, the image label is used in dictionary learning with the goal of learning a discriminative dictionary optimized for the classification task. An ideal discriminative dictionary will distribute its reconstruction power according to the discriminative information in the descriptor space. Highly-discriminative patches will be reconstructed with high fidelity while uninformative generic or background clutter patches will be coarsely reconstructed with fewer bits to suppress the noise in the signal.

In dictionary learning with descriptor level supervision, the image class label is assigned to each descriptor in the *bag-of-descriptors*. The discriminative dictionary is then constructed either by growing the quantizer discriminatively or by merging codewords in the unsupervised clustering dictionary to optimize the mutual information between class label and codewords. Note, however, that such an objective is not necessarily consistent with the goal of maximizing image-level classification performance, and therefore might lead to a suboptimal solution.

For tree-structured models, node splitting can be easily adapted to maximize information gain and partition the descriptor space discriminatively. Randomized clustering forests [61] and texton forests [79] both train an ensemble of trees with bootstrap resamlping of the training data and selection of random subsets of node-splitting tests as

in *random forests*[13]. The random forests are treated as a discriminative quantizer, and (interal or leaf) nodes of the trees are considered as dictionary codewords. A histogram is constructed by dropping descriptors from an image through the trees and keeping track of which nodes that are visited.

Another possible method for supervised dictionary learning is to merge or adapt the unsupervised clustering dictionary with respect to mutual information criteria [49, 94, 29]. All those methods are based on the *information bottleneck* [85] framework.

Yang *et al.* [99] perform mutual information-based feature selection on visual words. The results show that the percentage of uninformative codewords in images is much lower than in documents. That is to say almost every visual codeword has discriminative power contributing to the classification. Therefore, clustering over codewords is preferred rather than feature selection.

The *Information Bottleneck* method is a general theoretical framework for clustering. It aims to find a compressed representation $\tilde{X}$ of the data X under the constraint that enough mutual information between $\tilde{X}$ and another relevant random variable Y is preserved. The objective function is of the form:

$$\max_{\tilde{X}} I(\tilde{X};Y) - \beta I(\tilde{X};X),$$

In our case, Y is the image class label. The objective is to maximize the mutual information $I(\tilde{X};Y)$ under the constraint of minimizing $I(\tilde{X};X)$. This information theoretical clustering method was first used in the text analysis domain to cluster words into word clusters. This produces a more compact representation of documents and achieves better classification accuracy [84, 4].

Winn *et al.* [94] and Fulkerson *et al.* [29] both tried to optimize a compact dictionary by merging an initially large dictionary. In [94] the goal is to maximize the mutual information $I(\mathbf{h};Y)$ between image-level histogram $\mathbf{h}$ and the class label Y. In order to estimate $I(\mathbf{h};Y)$ between Y and $\mathbf{h}$, a strong generative assumption is made that histograms $\mathbf{h}$ are distributed according to a mixture of Gaussians. Fulkerson *et al.* [29] optimize over the mutual information $I(w;Y)$ between the individual codeword w and the class label Y. They use *agglomerative information bottleneck* [83], which is a bottom up hard version of the information bottleneck method. The initial dictionary is generated by hierarchical k-means [63]. Together with their fast implementation of the

agglomerative information bottleneck algorithm, their approach is efficient and can scale to large dictionaries.

Lazebnik *et al.* [49] present a supervised quantizer learning algorithm that works directly in the continuous descriptor space with no need for an initial dictionary. The result shows that initialized by k-means clustering, the algorithm can optimize the Voronoi partition of the feature space to minimize the information loss. However, due to complexity issues, the algorithm is limited to small dictionaries.

2.4.3 Supervision at the Image Level

One drawback of the aforementioned supervised dictionary learning approaches is that they are trying to optimize visual words independently at the descriptor level, while the true objective is to learn a discriminative image level representation for better classification.

One naive approach to image level supervision is to learn a class-specific dictionary for each category and concatenate them together into a single dictionary [90]. However, the size of the dictionary is $C \times K$, which scales with the number of classes C making it impractical for a large number of classes.

In addition, different classes share a large proportion of generic descriptors of low discriminative power, which must be repeatedly modeled in the different class-specific dictionaries. Perronnin *et al.* [68] combine universal and class-specific dictionaries. A universal dictionary is first learned using a *Gaussian mixture model* on all training data. A class-specific dictionary is adapted from the universal dictionary by MAP estimation to class-specific data with the parameters of the universal GMM model as a prior. In classification, a *one-versus-all* classifier for each class is trained. Each feature vector is a two-part histogram obtained by merging the class-specific and universal dictionaries. This reduces the feature length from $C \times K$ to $2K$. The universal dictionary models the generic visual features across classes, while the class-specific dictionary models the discriminative visual patches of each particular class.

Recent work tries to directly optimize the visual dictionary by minimizing the image level classification loss [103, 100, 45, 52, 98, 10].

Zhang *et al.* [103] adopt a boosted dictionary approach. They wrap the entire BOW pipeline in a boosting framework. In each iteration, a dictionary is built to capture

non-redundant discriminative information missed by the preceding dictionaries and classifiers. A weighted k-means clustering using Adaboost weights is employed as the weak dictionary learner.

Yang *et al.* [100] adopt a similar unified framework. In their work, a *visual bit*— a linear function that maps the local descriptors to a binary bit—is used as the building block for a visual dictionary. Each descriptor vectors is encoded by a sequence of *visual bits* that are optimized iteratively based on the classification loss of the preceding *visual bits*.

Krapac *et al.* [45] extends the tree-structured quantizers [61] with image-level supervision. Instead of growing the tree using information-gain at the descriptor level, they incrementally grow the trees to quantize the descriptor space by greedily selecting the best split with a criterion that directly evaluates image-level classification performance.

Another method for image-level supervised dictionary learning is to formulate a joint optimization problem over the dictionary and the classification model [52, 98, 10]. Since the objective function is not jointly convex, a coordinate descent algorithm is adopted to alternate between updating the dictionary and learning the parameters of the classifier. Since the dictionary construction step in their work is based on sparse coding, we will discuss it in detail when sparse coding is presented.

2.5 Descriptor Encoding

In descriptor encoding, continuous local descriptors are mapped to the discrete codebook space. Standard hard vector quantization maps each descriptor to the nearest codeword in the dictionary. Such hard vector quantization outputs a 1-of-K binary indicator vector for each descriptor:

$$\mathbf{c}_j \in \{0,1\}^K, \quad c_{jk} = 1 \quad \texttt{iff} \quad k = \underset{1 \le k \le K}{\operatorname{argmin}} \|\mathbf{x}_j - \mu_k\|_2^2.$$

This hard assignment is problematic and leads to large reconstruction error and information loss. When the dictionary becomes large, the volume of each Voronoi cell is so small that it makes such nearest neighbor assignments unstable. This hurts the generalization performance of the classifier.

To overcome this drawback, soft assignment has been studied by various researchers.

Moosmann *et al.* [61] build an ensemble of random clustering trees. Each descriptor drops through all trees and turns on multiple leaf nodes, which are defined as codewords in the dictionary. Jiang *et al.* [39] and Tuytelaars *et al.* [87] assign a descriptor to the k nearest visual words in the descriptor space. More generally, a probabilistic soft weighting scheme has been used [1, 70, 68, 24]. A probabilistic mixture model is fitted to the distribution of descriptors in descriptor space. For a new descriptor, the weight of each codeword is assigned as the posterior mixture component membership probabilities given the descriptor.

2.5.1 Visual Word Ambiguity

Gemert *et al.* [89, 88] provide a systematic analysis of visual word ambiguity in the *bag-of-words* representation of an image. They model two types of ambiguity in hard vector quantization: *visual word uncertainty* and *visual word plausibility*. Uncertainty refers to selecting the nearest codeword from several close neighboring candidates, while plausibility means lack of suitable candidates in the dictionary, in that even the nearest codeword is far away from the descriptor vector. It is essentially a measure of quantization error.

Their analysis is performed in the *kernel codebook* framework. Recall in the baseline, with hard vector quantization and average pooling, the image-level representation is a histogram estimate of the distribution of codewords in the image. The proposed *kernel codebook* adopts the technique of kernel density estimation to provide a robust and smooth alternative to the histogram estimator.

The traditional vector quantization histogram is defined as

$$\texttt{VQ}(\mathbf{c}) = \frac{1}{M}\sum_{j=1}^{M}\begin{cases}1 & if\ \mathbf{c}_k = \text{argmin}_{\mathbf{c}\in\mathbf{C}}\,||\mathbf{c}-\mathbf{x}_j||_2 \\ 0 & otherwise\end{cases}$$

By replacing the histogram with a kernel density estimator, the *kernel codebook* is defined as

$$\texttt{KCB}(\mathbf{c}) = \frac{1}{M}\sum_{j=1}^{M} K_\sigma(\mathbf{c},\mathbf{x}_j).$$

The *kernel codebook* takes both *visual word uncertainty* and *visual word plausibility* into account. In the experiment, a Gaussian kernel with fixed kernel width is selected. In this

case, the kernel density estimator is equivalent to *Gaussian mixture model* with identical Gaussian components placed at each codeword. The *kernel codebook* representation is the descriptor density distribution at each codeword.

In order to analyze the individual effect of the two ambiguity types, they propose to model each of those separately.

Codeword uncertainty is defined as

$$\texttt{UNC}(\mathbf{c}) = \frac{1}{M} \sum_{j=1}^{M} \frac{K_\sigma(\mathbf{c}, \mathbf{x}_j)}{\sum_{k=1}^{K} K_\sigma(\mathbf{c}_k, \mathbf{x}_j).}$$

This is equivalent to the soft weighting scheme in parametric mixture model using posterior membership probabilities.

Codeword plausibility is defined as

$$\texttt{PLA}(\mathbf{c}) = \frac{1}{M} \begin{cases} K_\sigma(\mathbf{c}_k, \mathbf{x}_j) & if\ \mathbf{c}_k = \text{argmin}_{\mathbf{c} \in \mathbf{C}} ||\mathbf{c} - \mathbf{x}_j||_2 \\ 0 & otherwise. \end{cases}$$

This is a weighted version of hard vector quantization, where each descriptor is weighted by its distance to the nearest codeword.

Their extensive experiments show that the *codeword uncertainty* consistently outperforms all other types of ambiguity modelling, while *codeword plausibility* is even worse than hard vector quantization. This suggests that explicit *codeword plausibility* modelling will hurt the classification performance. A possible explanation is that *codeword plausibility* suppresses the weight of faraway descriptors leaving them unrepresented, which leads to severe information loss.

Another interesting experiment shows that large dictionary size does not always improve classification accuracy. When increasing the size of the dictionary, both soft and hard encoding performance decrease after certain point. However, soft encoding is more robust with only a slight decrease, while significant deterioration is observed for hard encoding. This can be explained by the better generalization capability of soft encoding.

2.5.2 Sparse Coding

Sparse coding aims to represent signals with a sparse linear combination of bases or codewords in a over-complete dictionary. The learned basis functions capture high-level features in the data. The use of sparse coding to represent natural images was first proposed by [65] to model the receptive fields of neurons in the visual cortex. It has been successfully applied to numerous low-level image processing tasks [2, 22] and visual categorization [95, 74, 96] as well. Sparse coding can be regarded as a soft encoding method with an additional sparsity constraint. The sparse representation will have fewer collisions during pooling, so less information will be lost.

Given a signal $\mathbf{x}_j$, sparse coding finds a sparse linear combination representation $\mathbf{c}_j$ of bases in an over-complete dictionary $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_K] \in \mathcal{R}^{d \times K}$. Let $\mathrm{C} = [c_1, c_2, ..., c_M]$ denote the encoded representations of the input signals $x_1, ..., x_M$. We find $\mathbf{D}$ and $\mathbf{C}$ by jointly minimizing the following objective function:

$$\min_{\mathbf{D} \in \mathcal{R}^{d \times K}, \mathbf{C} \in \mathcal{R}^{K \times M}} \sum_{j=1}^{M} \frac{1}{2} \|\mathbf{x}_j - \mathbf{D}\mathbf{c}_j\|_2^2 + \lambda \|\mathbf{c}_j\|_1,$$

$$\texttt{s.t.} \quad ||\mathbf{d}_i||_2^2 \leq 1, \ \forall i = 1, ..., k,$$

where λ is the regularization factor. The first term in the cost function is the error of reconstructing the descriptor $\mathbf{x}_j$, and the second term is the L_1 regularization to encourage sparsity.

Other sparsity-inducing norms such as the L_0 norm can also be used. However the L_0 regularization problem is NP-hard and is often approximated by a greedy algorithms [58]. In practice, L_1-regularized sparse coding is more stable and less sensitive to small perturbations of the input signal than L_0.

The L_1-regularized optimization problem is not convex in $\mathbf{D}$ and $\mathbf{C}$ simultaneously, but it is convex in $\mathbf{D}$ or $\mathbf{C}$ individually when the other is fixed. So it is usually approximated by coordinate descent between $\mathbf{D}$ and $\mathbf{C}$. Given $\mathbf{C}$, learning the dictionary $\mathbf{D}$ is a least squares problem with quadratic constraints. Given $\mathbf{D}$, learning the reconstruction coefficients $\mathbf{C}$ is an L_1-regularized least squares problem. Least angle regression [19] or a more efficient feature-sign algorithm [50] can be used to efficiently solve the problem at large scale.

Traditionally, sparse coding is performed on either the entire image or on patches of raw pixels. Yang *et al.* [96] first proposed applying sparse coding on SIFT descriptors. Combined with spatial pyramid pooling and a linear SVM, their ScSPM system achieves state-of-the-art performance on several benchmarks. This shows the power of sparse coding with the *Bag-of-Words* representation.

Note that sparse coding includes both dictionary learning and descriptor encoding phases. We present it in the encoding section, because the success of sparse coding in image classification has been mainly attributed to the encoding step [14].

Given the dictionary $\mathbf{D}$, the descriptor encoding step is defined as an L_1-regularized least squares problem:

$$\mathbf{c}_j = \underset{\mathbf{c}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{x}_j - \mathbf{D}\mathbf{c}\|_2^2 + \lambda\|\mathbf{c}\|_1$$

The descriptor encoding step of sparse coding can be viewed as a soft assignment that better reconstructs the signal than hard vector quantization. Ng *et al.* [14] show that sparse coding based on randomly-sampled dictionary basis vectors has classification accuracy comparable to that of a learned dictionary. This suggests that soft encoding is more important than dictionary learning.

In additional to soft encoding, the sparsity constraint is also important for reducing the information lost during the pooling step in the *Bag-of-words* representation. That may account for the improved performance of sparse coding over other soft encoding methods. We will discuss this in detail in the next section.

There are several variants of sparse coding adapted for the classification task. We will briefly introduce them in the following.

Supervised Sparse Coding. In standard sparse coding, the objective function is to minimize the reconstruction error. This is suboptimal for discriminant analysis. Recent work is interested in learning discriminative dictionaries for sparse coding. Mairal *et al.* [56, 55] proposed to optimize the sparse coding jointly with a linear prediction model such as logistic regression with the logistic loss function. Their approach only works for standard single-instance supervised learning. Yang *et al.* [98] and Boureau *et al.*[10] extend the discriminative dictionary learning to the *bag-of-words* representation by taking the feature pooling step into account. The optimization problem is formulated directly over the image-level representation $\mathbf{h}$ in a linear classification model:

$$\min_{\mathbf{w},\mathbf{D}} \sum_{i=1}^{N} \mathcal{L}(y_i, f(\phi(\mathbf{X}_i, \mathbf{D}), \mathbf{w})) + \lambda \|\mathbf{w}\|_2^2,$$

where $f(\mathbf{h}, \mathbf{w})$ is the linear predictive model with parameter $\mathbf{w}$ and evaluated on feature vector $\mathbf{h}$; $\phi(\mathbf{X}_i, \mathbf{D})$ is defined as the feature construction process (descriptor encoding + feature pooling). The complex feature construction process makes the gradient computation with respect to $\mathbf{D}$ very inefficient. The algorithm usually converges to local optima close to the point where it is initialized. The results reported on benchmarks do not show signification improvement over unsupervised dictionaries [10].

Random Sampled Dictionary. Another line of research [74, 14] downplays the importance of well trained dictionaries in sparse coding. They argue that encoding is more important than dictionary learning as long as the dictionary provides a reasonable tiling of the input space. In [74], they show that a dictionary learned from unlabelled randomly retrieved data can provide an informative representation for discriminative classification tasks. In [14], they demonstrate that a random sampled dictionary of sufficient size is able to obtain performance as high as their sophisticated trained counterparts. Note that they do not compare with supervised dictionaries.

Locality-Constrained Sparse Coding. In standard sparse coding, descriptors are encoded independently. Since the L_1 norm in sparse coding is not smooth, with a huge over-completed dictionary, sparse codes might vary greatly for small perturbations in the descriptor vectors. This hurts generalization. To overcome this drawback, a locality constraint has been adopted to quantize descriptor vectors more robustly [30, 101, 93, 97].

Laplacian Sparse Coding (LSC)[30] adds a second similarity regularizer in the objective function using a Laplacian matrix that encourages similar descriptors to have similar sparse encodings in the coding space.

Local Coordinate Coding (LCC) [101] has been proposed to approximate a general nonlinear function by a global linear function with respect to local coordinate coding. It shows theoretically that approximation quality is bounded by both the reconstruction error and the locality of the coding, and thus locality is more essential than sparsity for nonlinear function learning. Sparse coding is a special case of *local coordinate coding* with no locality constraints. Based on the same reasoning of *Lipschitz smooth* function approximation, Zhou *et al.* [104] extend hard vector quantization to a super-vector

representation. Specifically, each binary bit is expanded to a $d+1$ vector $[s, (\mathbf{x}-\mathbf{d_k})^\top]^\top$ representing the residual between the input $\mathbf{x}$ to the cluster center $\mathbf{d_k}$.

Wang *et al.* [93] proposed *Locality-constrained Linear Coding (LLC)*, which is a fast approximation of LCC with L_2 regularization:

$$\mathbf{c}_j = \underset{\mathbf{c}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{x}-\mathbf{D}\mathbf{c}\|_2^2 + \lambda\|\mathbf{r}^\top \mathbf{c}\|_2^2$$

$$s.t. \quad \mathbf{1}^T\mathbf{c} = 1,$$

where $\mathbf{r} = \exp(\frac{dist(\mathbf{x},D)}{\sigma}))$ is a vector representing distance from descriptors $\mathbf{x}$ to each basis in $\mathbf{D}$.

The LLC optimization has a closed form solution. In practice, an even faster k nearest neighbor approximation is adopted, and it achieves state-of-art performance. Because of its efficiency and good performance, we choose LLC as a representative of state-of-art BOW model and evaluate its performance on both of our datasets in chapter 4.

Yang *et al.* [97] proposed another approximation to LCC by a mixture sparse coding model. It efficiently produces a huge dictionary with local constraints by preclustering over the descriptor space. A different sparse coding is learned within each cluster. The final feature representation is the concatenation of encoding the descriptor using each of the sparse codes.

The advantage of locality-constrained sparse coding for the classification task can also be explained by its effect of reducing overlap during the feature pooling process. We will discuss this in the next section.

2.6 Feature Pooling

The feature pooling process is the final step in generating a single feature vector for the entire image. In this step, the goal is to summarize the encoded *bag-of-descriptors* a robust and discriminative fixed-length representation of the image.

2.6.1 Basic Pooling Function

Traditionally, with a hard encoding scheme, a histogram representation of the image is generated by average pooling.

Given a bag of encoded descriptors $\{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_M\}$, the average pooling function is defined as

$$\mathbf{h} = \frac{1}{M} \sum_{j=1}^{M} \mathbf{c}_j,$$

where $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_M] \in \mathcal{R}^{K \times M}$ is the matrix form of the encoded *bag-of-descriptors* of an image. Each patch is now encoded by a column in $\mathbf{C}$ of length K.

The image level feature vector is a histogram of length K normalized by the number of local patches i.e. the *cardinality* of the image. It simply counts the frequency of individual visual codewords occurring in the image.

However, not all codewords are of equal importance. Some codewords that are ubiquitous over all categories carry no discriminative information. *Term frequency-inverse document frequency* (tf-idf) pooling [75] is proposed to reduce the impact of frequently-occurring codewords. Originally introduced for text retrieval, *tf-idf* adopts an *inverse document frequency (idf)* codeword weighting scheme that penalizes frequently-appearing codewords. The *tf-idf* pooling improves the robustness of learning algorithms when the distribution of codewords is significantly unbalanced as shown by [41]. The use of *tf-idf* pooling in visual categorization [103, 99, 81] shows systematic improvement over the plain histogram.

Although *tf-idf* reduces the effect of the unbalanced distribution of codewords across examples, it does not take into account the unbalanced occurrence of visual features within a single image. Jegou *et al.* [38] call this phenomenon *the burstiness of the visual elements*. That means that a visual word is more likely to appear in an image if it already appeared once in that image. This is because there are repetitive patterns in natural images and man-made objects. This suggests that local descriptors do not satisfy the assumption of statistical independence, and therefore the frequency-based histogram over-counts these burst of descriptors. The diversity of descriptor vectors is more important than the absolute number of such descriptors. For example, a pair of images sharing ten descriptors of the same type (assigned to the same codeword) might not be as similar as a pair of images that have five different types of descriptors of

different types.

This hypothesis give rise to the presence/absence pooling scheme, which argues that the presence or absence of a visual word is more discriminative than its frequency. Presence/absence pooling is defined as

$$h_k = \begin{cases} 1 & \exists\, c_{kj} = 1 \quad 1 \le j \le M \\ 0 & otherwise \end{cases}, \quad for\ k = 1, ..., K,$$

A binary feature vector $\mathbf{h}$ is generated, where h_k is 1 if and only if there is at least one descriptor vector mapped to the kth codeword. Superior performance has been reported both in text domains [67] and image domains [99] using presence/absence pooling.

The above mentioned pooling schemes originated in text analysis, where each descriptor is hard encoded as a 1-of-K codewords. A max pooling strategy has been developed [96, 93, 12, 10] in the context of soft encoding, where the codes are in continuous K dimensional space. The max pooling function is defined by

$$h_k = \max_{1 \le j \le M} |c_{kj}|, \quad for\ k = 1, ..., K,$$

For every codeword k, the max pooling function outputs h_k as the maximum absolute value over the corresponding coefficients in the bag of codes of the image.

Inspired by biophysical research on modelling the visual cortex (V1) [46], Yang *et al.*[96] first adopt max spatial pooling over the sparse codewords and achieve excellent performance using linear SVMs for classification. They suggests there is a special compatibility between max pooling and linear SVMs with sparse coding. Max pooling suppresses small coefficients in sparse coding and prevents smearing out of prominent signals.

LeCun *et al.* [10] perform extensive experiments of different combinations of encoding, pooling, and classification methods. One of their conclusions is that max pooling almost always improves over average pooling regardless of the encoding approaches. Contrary to Yang's results [96], linear SVMs does not outperform kernel SVMs with max pooling. But max pooling does narrow the performance gap between linear and kernel SVMs. For linear classification, the most significant performance boost is switching from average to max pooling. With a linear classifier, even simple hard vector quantization

with max pooling outperforms sparse coding paired with average pooling. Note that max pooling over hard vector quantization generates binary feature vectors, which is equivalent to presence/absence pooling.

Boureau *et al.* [12] give a theoretical analysis of different feature pooling methods. They show that the max pooling strategy is well suited to sparse features that produce fewer collisions during pooling.

2.6.2 Spatial Pooling

Up to this point in our review, all pooling is performed over the entire image, which gives a completely orderless *BOW* representation, without taking the spatial layout of local patches into consideration. Spatial pooling methods pool encoded patches corresponding to local image regions. Koenderink *et al.*[44] first proposed the concept of locally orderless images. Similar local spatial pooling processes have been adopted in the design of local patch descriptors [54] [16], where gradient orientation histograms are computed in local regions giving robustness to small transformations of local patches.

Lazebnik *et al.* [76] first introduce the spatial pyramid pooling method for BOW representations. The original pyramid matching kernel [35] partitions the high-dimensional descriptor space into bins, which allows efficient and precise matching of two unordered feature sets. In contrast, Lazebnik *et al.* [76] partition in the two-dimensional image coordinates into bins at multiple spatial resolutions. The image is partitioned into finer spatial bins as the level in the pyramid increases. Average pooling is performed over local features within each spatial bin, and the pooled vectors from multiple bins are concatenated together to obtain a feature vector for the entire image. The pyramid structure coarsely preserves the spatial information and achieves invariance (within each bin) to local translations. A spatial pyramid matching kernel is employed to perform classification. The formula of the spatial pyramid kernel is presented in the next section.

Recent work [96, 10, 93] also adopts spatial pooling. Local encoded descriptors are max pooled within each spatial bin, and linear classification is performed afterwards. All the experiments confirm the superiority of spatial pooling over an orderless representation.

In general, spatial pooling can be defined as

$$\mathbf{h}_s = g_{(\mathbf{s}_j \in \mathcal{R}_s)}(\mathbf{C}),$$

where $g(\mathbf{C})$ is one of the basic pooling functions and $\mathcal{R}_s$ are local image regions in which pooling is performed. The local regions sometimes overlap when a spatial pyramid structure is applied. The final image representation $\mathbf{h}$ is the concatenation of all pooled feature vectors $\mathbf{h}_s$.

2.6.3 Local Neighborhood Pooling

Boureau *et al.* [11] address the often neglected loss of information during the pooling process. They argue that pooling of local descriptor vectors that are far apart in descriptor space will smear out individual signals, so pooling should only be performed on patches with similar descriptor vectors. While this condition is naturally satisfied by hard vector quantization, it is not the case for sparse coding. In standard sparse coding, descriptor vectors are encoded independently without a locality constraint. Sparse codes might vary greatly for similar descriptor vectors, and each dictionary basis might contribute to represent very different descriptor vectors. This overlap of encoded descriptors leads to loss of information when pooling in the sparse signal space.

Therefore, Boureau *et al.* proposed a multi-way local pooling by partitioning the 2D image space and the high dimensional descriptor space jointly. The image space is partitioned using a spatial pyramid structure, while descriptor space bins are produced by unsupervised clustering of the descriptor vectors after encoding. Multi-way local pooling can be formulated as

$$\mathbf{h}_{(s,p)} = g_{(\mathbf{s}_j \in \mathcal{R}_s, \mathbf{c}_j \in \mathcal{C}_p)}(\mathbf{C}),$$

where $\mathcal{R}_s$ is a local region in image coordinates (spatial bins in spatial pooling), $\mathcal{C}_p$ stands for local regions in descriptor space (high dimensional Voronoi cells), and $\mathbf{h}_{(s,p)}$ is generated by pooling within each cell (s, p). The final image representation $\mathbf{h}$ is the concatenation of all pooled feature vectors $\mathbf{h}_{(s,p)}$. Multi-way local pooling can boost the performance of the dictionary considerably, given a fine enough descriptor space partition (P=64).

Yang *et al.* [97] proposed to the learn a huge over-complete dictionary. To make the

learning process efficient, they perform a similar preclustering step. Local descriptors belonging to different clusters are encoded with different sparse coding dictionaries. This can be viewed as an approximation to *local coordinate coding* and inherently leads to pooling that is more local.

2.7 Classifier Design

By pooling the *bag-of-descriptors* into a single feature vector, we transform the image categorization problem into the standard multi-class classification problem. Then we can use a set of images with class labels to train the classifier. After training, the classifier can be applied to predict the visual categories of unlabelled images.

Different types of classifiers have been employed in image classification. As in general machine learning problems, the classifiers can be categorized into two types: generative models and discriminative models.

Note that there are other approaches based on different feature construction methods. Several methods circumvent the construction of image level feature representation and operate directly on the desciptors. Eichhorn *et al.* and Grauman *et al.* [21, 35] use SVM-based similarity matching kernels on two sets of local descriptors, and Boiman [7] adopts a nearest neighor classifier based on an image-to-class distance measure. Feifei and Fergus [26, 28] propose probabilistic generative part-based models. In this survey we restrict our discussion in classifiers built on *bag-of-words* features.

2.7.1 Generative models

In a generative model, the classifier learns a model of the joint probability $P(\mathbf{X}, Y)$ of input data and output target. Several generative model-based approaches have been proposed [15, 27].

Naive Bayes. The Naive Bayes classifer is often used in text classification. Csurka *et al.* [15] applied it to the image classification problem. It employs the naive assumption that the encoded descriptors of local patches are independent given the class label. As shown in figure 2.2, the generative process can be described as follows: (a) The image class label y is drawn from a prior distribution. (b) Each encoded descriptor w_j is then chosen independently from a multinomial distribution $p(w_j|y)$ given the class label.

The joint probability is defined as

$$p(\mathbf{w}, y) = p(y) \prod_{j=1}^{M} p(w_j|y)$$

The parameters of Naive Bayes can be learned by maximizing the log-likelihood with Laplace smoothing. Inference is also straight forward by maximizing the posterior probability $p(y|\mathbf{w}) \propto p(\mathbf{w}, y)$.

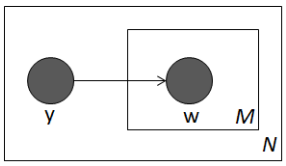

Figure 2.2: Naive Bayes model

Supervised LDA. Just as words in common phrases are conditionally dependent of each other, in images there are shared causal factors (e.g., pose, appearance of an object) that influence the correlations among patches. Hence, the assumption of independent local patches is not valid. Topic models introduce an intermediate topic level representation. Topics are soft clusters of visual words. An image can have multiple topics, and each topic is defined as a distribution over the visual words. The standard topic models are unsupervised and are learned using an algorithm called Latent Dirichlet Allocation (LDA) [5]. Discriminative classifiers have been trained using the per-image topic distribution as a feature vector [72, 82, 71, 48, 80].

Feifei *et al.* [27] introduce a supervised LDA model by adding a class label node y as shown in figure 2.3 The Dirichlet prior distribution $\boldsymbol{\theta}$ over topic probabilities is drawn from a conditional distribution $p(\boldsymbol{\theta}|y, \boldsymbol{\alpha})$ given class label y.

The joint probability is defined as

$$p(\mathbf{w}, \mathbf{z}, \boldsymbol{\theta}, y|\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\eta}) = p(y|\boldsymbol{\eta}) p(\boldsymbol{\theta}|y, \boldsymbol{\alpha}) \prod_{j=1}^{N} p(z_j|\boldsymbol{\theta}) p(w_j|z_j, \boldsymbol{\beta}).$$

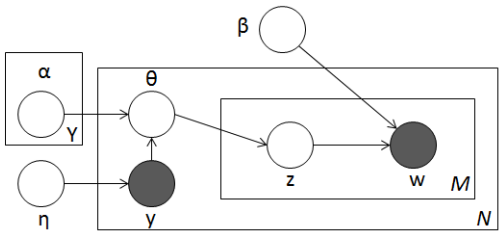

Figure 2.3: Supervised Latent Dirichlet Allocation model

Prediction is made by maximizing the posterior probability of the class label:

$$p(y|\mathbf{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\eta}) \propto p(\mathbf{w}|y, \boldsymbol{\alpha}, \boldsymbol{\beta}) p(y|\boldsymbol{\eta}) \propto p(\mathbf{w}|y, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

Assuming a uniform class prior, this is proportional to maximizing the probability of $\mathbf{w}$ given class label y:

$$\hat{y} = \underset{y}{\operatorname{argmax}}\, p(\mathbf{w}|y, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \int p(\boldsymbol{\theta}|\boldsymbol{\alpha}, y) \prod_{j=1}^{N} \sum_{z_j} p(z_j|\boldsymbol{\theta}) p(w_j|z_j, \boldsymbol{\beta}).$$

This probability can be computed by approximate inference algorithms such as *variational message passing*.

In general, the performance of generative models is not as good as discriminative models [76, 48]. However the generative topic model provides a compact and more robust image representation and outperforms discriminative classifiers when the training sample is very small [72, 62].

2.7.2 Discriminative models

In discriminative models, the classifier directly models the posterior target probability $P(Y|\mathbf{X})$ conditioned on the input data.

Because of the large variability in images and the use of very long feature vectors in *bag-of-words* representations, there are three requirements for the classifier: a) good generalization capacity to unseen examples; b) the ability to handle long feature vectors without overfitting; and c) efficiency in both training and testing in order to deal with very large datasets [18].

The **k-nearest-neighbor classifier (KNN)** has suboptimal performance because the standard distance measures (e.g. Euclidean distance) tend to lose their meaning for high-dimensional descriptor space. It also suffers from high-variance caused by finite training samples. Zhang *et al.* proposed SVM-KNN [102], a hybrid method of KNN and SVM. It performs *k-nearest-neighbor* search globally and learns a more smooth local boundary using SVM with a distance preserving kernel matrix on the collection of neighbors.

The **Support Vector Machine (SVM)** is the most popular classifier for the *bag-of-words* representation. It has been widely used in the literature [15, 96, 9, 76, 35].

The linear SVM was first applied to *bag-of-words* features by Csurka, et al. [15]. People empirically found that particular types of non-linear kernel SVM can achieve better performance [9, 76, 57].

Because the traditional *bag-of-words* representation is a histogram-based feature vector, the Euclidean distance based RBF kernel is not a suitable choice. Other kernels specifically designed for comparing distributions, such as the *intersection kernel*, *Earth mover's distance(EMD) kernel*, and χ^2 *kernel* have been broadly applied.

The *histogram intersection kernel* measures the similarity between two histograms defined as

$$k_{HI}(h_a, h_b) = \sum_{i=1}^{n} min(h_a(i), h_b(i)).$$

The χ^2 distance is another measure of distance between histograms [9]. The χ^2 kernel is defined as

$$k_{\chi^2}(h_a, h_b) = 1 - \sum_{i=1}^{n} \frac{(h_a(i) - h_b(i))^2}{\frac{1}{2}(h_a(i) + h_b(i))}.$$

As discussed in section 2.6.2, a *spatial pyramid matching kernel* (SPM) is proposed [76] to preserve spatial information. The original *pyramid matching kernel* [35] places a set of increasingly coarser grids over the feature space and takes a weighted sum of the number of matches that occur at each level of resolution. The formula is defined as

$$k^L(h_a, h_b) = I^L + \sum_{l=1}^{L-1} \frac{1}{2^{L-l+1}}(I^l - I^{l+1}), \quad l = 0, ..., L,$$

where l is the level in the pyramid with L being the finest level; I^l is the abbreviation

of $k_{HI}(h_a^l, h_b^l)$, the histogram intersection kernel function of features at level l. Matches at finer level have larger weights. The matches at level l include all matches at finer level $l+1$, so the new matches found at level l can be computed as $k^l - k^{l+1}$.

The *spatial pyramid kernel*[76] takes an orthogonal approach to perform pyramid matching in the two-dimensional image space. Therefore, l is the level of grid resolution in 2D image space and h^l is the encoded image-level histogram in each level l.

The use of non-linear kernels produces good performance at the cost of complexity. A non-linear SVM has training complexity $O(N^2d)$ and test complexity $O(Nd)$. The speed becomes a major concern when handling thousands of training images in large scale datasets [18]. Maji *et al.* [57] propose a fast computation of the *intersection kernel* that reduces the test complexity to $O(d \log N)$, but the training complexity remains high.

The linear SVM enjoys low complexity in both training and testing. The training complexity is $O(Nd)$, and the test complexity is $O(d)$. As a consequence, it has regained popularity in recent work [96, 10] . Explicit feature embedding that directly maps the feature vector into a new space in which the data is more linearly separable is adopted by Perronnin *et al.* [69]. They show that the square-root of the *bag-of-words* histogram is equivalent to the use of the Bhattacharyya kernel.

Other methods such as SVM-KNN [102] and Random Forests [8] are also more efficient in training and prediction than kernel SVMs. A comparison of the time complexity of different classifiers is shown in table 2.2.

Classifier	**Linear SVM**	**Kernel SVM**	**Random Forest**	**SVM-KNN**
Training Cost	$O(Nd)$	$O(N^2d)$	$O(N(\log N)^2\sqrt{d}\|T\|)$	NA
Test Cost	$O(d)$	$O(Nd)$	$O(\log N\|T\|)$	$O(dN + dk^2)$

Table 2.2: The train and test time complexity comparison of different classifiers. N is the number of images; d is the dimensionality of the feature vectors, T is the depth of the decision trees, and k is the number of nearest neighbors.

Ensemble Classifiers. Tree ensemble methods such as *random forests* [13] and *boosted decision trees* usually give performance comparable to SVMs. Those classifiers have also been explored in image classification [103, 8, 37].

Widely used in objection detection [92, 66, 86], the *boosted decision stump* is not suitable for BOW features. Boosted decision stumps perform feature selection over

the large set of potential features hoping to find a small set of discriminative features. However, in the *bag-of-word* representation, each visual word carries a small amount of discriminative information [7], so to achieve high performance, it is necessary to combine information from a large number of words.

The *boosted decision tree* is another choice. Zhang *et al.* [103] use the boosted decision tree as the base classifier. In each iteration, the dictionary size K is usually small, which means short feature vectors. However, for the large dictionaries adopted in state-of-the-art systems, the training of boosted decision tree will be slow. Therefore, the random forest classifier is preferred in such cases for its speed.

Bosch *et al.* [8] propose to use *random forests* over the spatial pyramid pooled appearance [76] and shape features [9]. To combine the features, at each node, the test feature is randomly selected from the types of features and levels of the pyramid. A linear classifier with random weights is adopted as the node test. It achieves state-of-the-art performance on the Caltech-101 [25] and Caltech-256 [36] datasets.

There are three advantages of *random forests* over SVMs: (a) compared with kernel SVMs, the parameters in *random forests*, number of trees $|T|$, maximum tree depth D and node test size R, are not very sensitive, which makes training easier. (b) Multiple kinds of features can be fused easily by adding them at the node tests, whereas for SVMs, complicated multiple kernel learning [3, 31, 91] is required to fuse multiple feature types. (c) The *random forest classifier* is inherently a multi-way classifier and can handle multiple classes with ease. When the number of image categories becomes large, the complexity of multi-class SVM increases either linearly for the *one-versus-all* strategy, or quadratically for the *one-versus-one* method.

2.8 Summary

A large volume of research in the *bag-of-word* representation for image classification has been presented in this survey. Different approaches explore various design dimensions of the system with diverse objectives and assumptions. A few general conclusions can be reached.

The aforementioned research provides a unified view of encoding and pooling steps. This sheds some light on the general design guidelines of the BOW representation. The goal of designing a single image-level feature vector is to summarize and maintain the

discriminative information of the local descriptors.

During the feature construction process, both encoding and pooling steps can lead to information loss. Sparse coding improves over hard vector quantization by decreasing the information loss in the encoding step. However, sparse codes can lead to the dilution of the signal during the pooling step. Recent work addresses this problem by taking into account locality in descriptor space. *LLC* [93] and *LSC* [30] both add locality constraints during sparse coding that forces codes to be local and therefore preserve neighorhood relationships, while multi-way pooling [11] performs a more selective local pooling over the standard sparse codes. This can be viewed as a joint effort to minimize the information loss in the pooling step.

In summary, the *state-of-the-art BOW* image representation favors a form of *super-vector* representation [104] as shown in figure 2.4, where different clusters of visually similar descriptors are mapped to separate local code blocks. Within each local code block, descriptors are represented by a similar linear combination of codes, this gives a rich and pooling-robust representation. Accordingly, 1-of-K hard vector quantization, which shrinks each code block into a single binary feature, and sparse coding, which overlaps different local code blocks, can both be viewed as special cases of this *super-vector* representation.

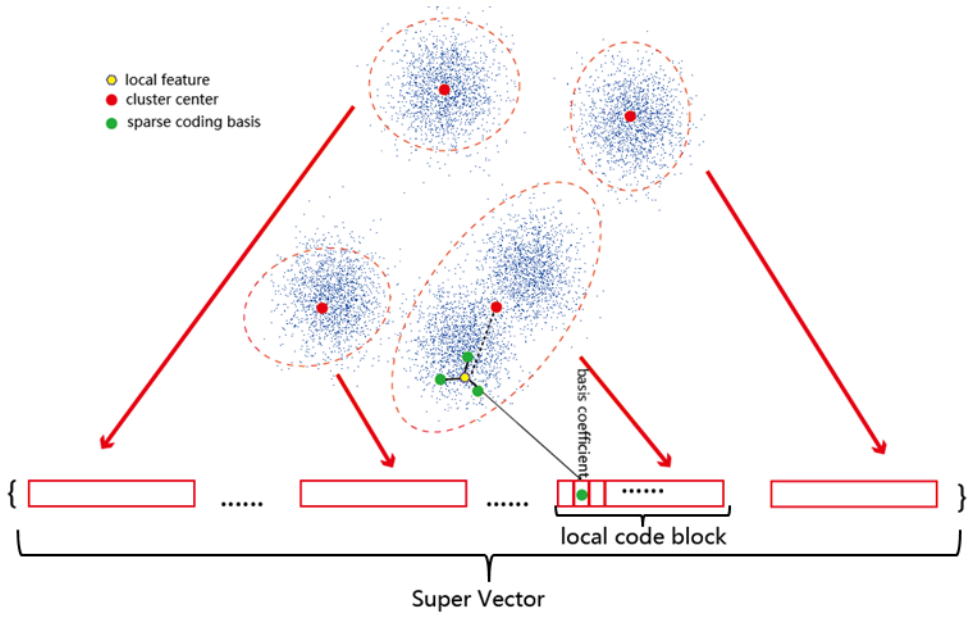

Figure 2.4: The illustration of super-vector representation. Different clusters of visually similar descriptors are mapped to separate local code blocks. Within each local code block, descriptors are represented by a similar linear combination of codes, this gives a rich and pooling-robust representation.

Chapter 3: Stacked Evidence Trees

3.1 Overview

The stacked evidence trees approach is yet another tree structured model. But unlike other models [61, 79] that treat the leaves of the trees as dictionary words, we view the trees as a way of discriminatively structuring the information in the training set. A random forest is trained as a first level classifier that predicts the class label for the entire image based on individual descriptors. Each leaf of the trees then stores a class distribution, a histogram of the number of training examples that reached that leaf. The class distributions of individual descriptors are accumulated for each image, and a second level stacked classifier is trained based on the aggregated image level features to make the final category prediction.

This elegant method is dictionary-free. Therefore, it avoids hand-engineered decisions about dictionary size as well as the loss of information when descriptors are encoded using a dictionary. It also employs descriptor level supervision information while training the random forest, which gives it another advantage over unsupervised dictionary construction in BOW model. Our experiments show that the stacked evidence trees gain a big performance improvement over the baseline *BOW* model. On STONEFLY9, they achieve 93.6% accuracy while the baseline BOW method scores 83.9%; on EPT54, they attain 77.4% versus 48.5%.

3.2 Classification Architecture

Figure 3.1 shows the overall architecture of the system. The input to the system is a set of different bags of detector/descriptor combinations extracted from the image. The descriptors can be SIFT, HOG, edge descriptors, etc. We learn an individual random forest for each bag of descriptors. The evidence histograms from different descriptors are fused at the second-level stacked classifier.

In classification, a set of (detector, descriptor) pairs $c = 1, ..., C$ are extracted from

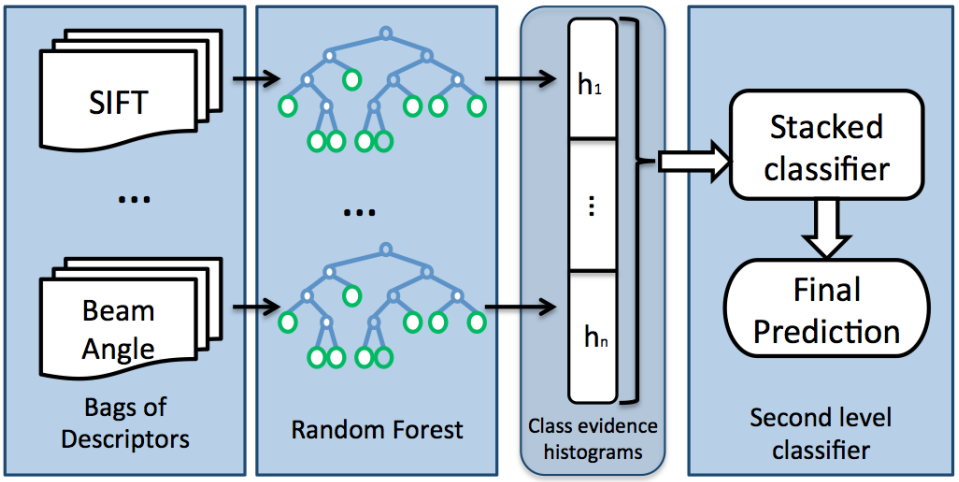

Figure 3.1: The overall classification architecture of stacked evidence trees. Descriptors are directly drop through the random forests without computing a visual dictionary. Each leaf node in the forest stores a histogram of class labels of the training examples. The class histogram at all such leaves are accumulated to produce a single image level class histogram which is then fed to a stacked classifier to make the final prediction.

the image. For each combination c, each descriptor $\mathbf{x}_j^c$ from the bag $\mathbf{X}^c = [\mathbf{x}_1^c, \mathbf{x}_2^c, ..., \mathbf{x}_M^c]$ is dropped through each tree in the previously-learned random forest RF^c until it reaches a leaf l. That leaf l stores evidence as a class distribution histogram h_l^c collected from the training examples. We then perform two-way of accumulation of the evidence. First we sum over all trees in the RF^c for every descriptor $\mathbf{x}_j^c$ to obtain h_j^c, then we sum over all descriptors to obtain image-level histogram h_c. Each h_c is normalized to 1 and then all C histograms are concatenated together to fuse information and form the second level feature vector. This image-level vector is then processed by the stacked classifier to produce the final prediction.

The learning process consists two steps: (a) learning the random forests and (b) learning the stacked classifier.

Learning a random forest. A random forest is an ensemble of randomized decision trees. This randomized ensemble structure captures the diversity and richness of high-dimensional descriptors. Each tree is constructed in the usual top-down way as in C4.5 [73]. However, two kinds of randomness are applied to add diversity among trees. First,

the training data for each tree is obtained by bootstrap resampling [20] of the training set. Second, at each node, only a subset of the attributes are evaluated to find the most discriminative combination of an attribute and a threshold. Specifically, at each node, only a subset of size $1 + |\log A|$ is chosen (where A is the number of attributes; 128 for SIFT).

We learn the random forest at the descriptor level. The image class label Y is assigned to each descriptor $\mathbf{x}_j$ computed from the image. A training example is created for each descriptor: $(\mathbf{x}_j, Y)$. But the bootstrap samples are generated at the image level by drawing images with replacement from the training set. We control the depth of the trees by limiting the minimum number of training examples in each leaf node. In each leaf, we store the class distribution histogram h_l as the number of training examples belonging to each class.

In our experiment, we set the minimum number of training examples per leaf node to 20 and trained a random forest containing 100 evidence trees for each detector/descriptor combination.

Learning the stacked classifier. We first constructed a second-level training set for the stacked image-level classifier. Since each tree in the random forest is grown on a bootstrap sample, for each tree there exists a set of "out-of-bag" images that were not used to grow that tree because they were not selected by the bootstrap resampling. Then for each image I, we only use the subset of trees in the forest that regard I as an "out-of-bag" image. The bag of descriptors of image I are dropped only through that subset of trees. The leaf histograms are summed only over those trees to obtain h_j^c and over all descriptors j to obtain h_c. Each h_c is normalized and concatenated to form the image-level feature vector for the stacking example. The class label of image I is assigned to the class label of this stacking example.

We employ a boosted decision tree ensemble as the stacked classifier. The number of boosting iterations is set to 200. There are only three parameters in the system: (a) the minimum number of training examples in each leaf node, (b) the number of trees in each random forest, and (c) the number of boosting iterations for the stacked classifier. Our experiments have shown that the results are insensitive to all of these parameters.

3.3 Relationship to BOW Approaches

Although the stacked evidence trees are somewhat different from the *BOW* model, the two types of approaches are closely related. Both methods start with bags of descriptors from the image, and both try to predict a single class label from this information. Since classifiers usually take a single fixed-length feature vector as input, pooling over the bag of descriptors is applied in both approaches. Both methods try to process the information in the bags of descriptors into a discriminative feature vector of moderate length, as the proper input to standard classifiers. But they are different in the sequence of steps.

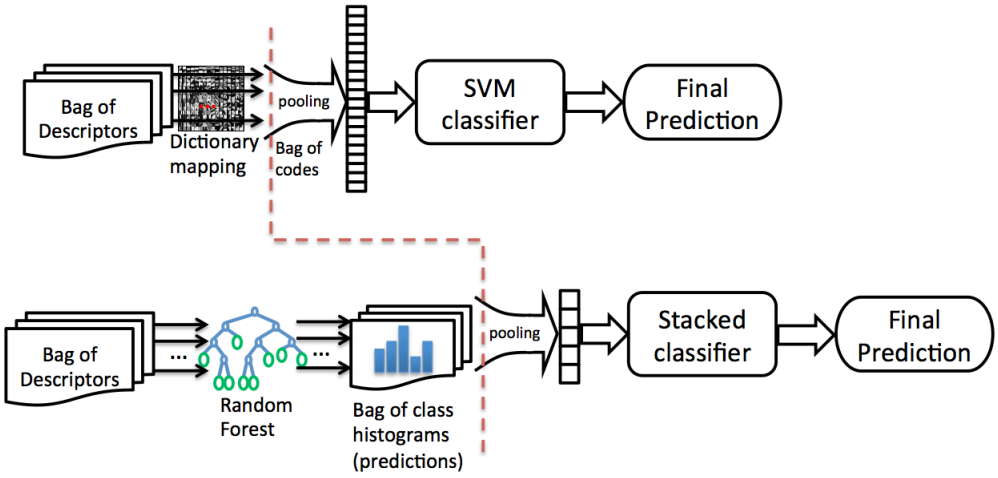

Figure 3.2: The comparison between the *BOW* model and the stacked evidence trees model. For the *BOW* model, the pooling happens before classification, while the evidence trees model first makes predictions on the individual descriptors and then pools the class distributions before the second image level classification.

As shown in figure 3.2, while the *BOW* model first maps each descriptor to a visual dictionary, pools the encoded *bag-of-descriptors* together, and then feeds the image-level feature vector to the classifier, our stacked evidence tree model makes prediction directly on each descriptor, pools the class distribution evidence from the individual descriptors together, and then feeds the aggregated class distribution evidence to the final classifier. Since the random forest is learned at the descriptor level, there is no need for a dictionary mapping process.

Interestingly, for linear pooling and linear classifier combination, the pooling and

classification steps are interchangeable for BOW models. Linear pooling can be defined as the linear weighted combination of individual descriptors

$$\mathbf{h} = \sum_{j=1}^{M} v_i \mathbf{x}_j,$$

and the linear classifier can be defined as the linear weighted combination of the image-level feature vector $\mathbf{h}$:

$$f(\mathbf{h}) = \mathbf{w}^T \mathbf{h}.$$

Then the *BOW* model can be written as the combination of the linear pooling and the linear classification:

$$f(\mathbf{h}) = \mathbf{w}^T \mathbf{h} = \sum_{j=1}^{M} v_i \mathbf{w}^T \mathbf{x}_j = \sum_{j=1}^{M} v_i f(\mathbf{x}_j),$$

where the linear pooling and linear classifier can commute. When the order of linear pooling and linear classification is reversed, the *BOW* model can be regarded as computing a discriminant function on the individual descriptors first and then pooling the discriminant function values together to reach the final decision. This resembles the stacked evidence trees approach.

3.4 Hypotheses Concerning the Requirements for Successful Fine-grained Categorization

The large intra-class variation and small inter-class differences make fine-grained categorization much more challenging than generic object categorization.

Is there any fundamental difference in designing systems to solve these two tasks? What are the requirements that must be satisfied to obtain good performance in fine-grained classification? We propose two hypotheses concerning fine grained categorization: a) high resolution images are needed to capture more detailed information from the objects; b) more aggressive information extraction, such as finer descriptor encoding with large dictionaries or classifiers based on raw descriptors, is required to extract more information from the bag-of-descriptors. We test both hypotheses in the following

chapter.

Chapter 4: Benchmarks and Experimental Tests of Our Hypotheses

In this chapter, we evaluate various fine-grained classification methods on the STONEFLY9 and the EPT54 datasets. To test our hypotheses, we analyze the effect of image resolution and dictionary size on the performance of these fine-grained classification methods.

Image capture and pre-processing. Images are captured using a semi-automated apparatus as described in Section 1.2. The specimens are photographed against a blue background. To segment each specimen from the background, we apply Bayesian matting and morphological operations. This results in a binary mask covering the specimen. For STONEFLY9, we apply three keypoint detectors: Hessian [60], Kadir-Brady [42], PCBR [17] and points falling on a regular grid on the mask. For EPT54, since each of the detectors fails on some of the species, we only compute descriptors using a regular grid.

For both datatsets, we employed a 3-fold cross validation process. Since each specimen is photographed multiple times, to ensure unbiased evaluation, all images of a single specimen are kept together in the same fold. We report average image classification accuracy and standard deviation over 3-fold cross validation.

4.1 Benchmarks

We first benchmarked the STONEFLY9 and EPT54 databases using three approaches: the baseline *Bag-of-Words* model, the *Locality-constrained Linear Coding (LLC)* [93] method, and the *Stacked Evidence Trees* model. For the baseline *Bag-of-Words* model, the dictionary size was set to 3000; for the *LLC* method, the dictionary was set to 3000 as well and the k in number of nearest neighbors was set to 5. The dictionaries are generated by a fast streaming k-means algorithm [78] based on the descriptors extracted from one training fold. For the *Stacked Evidence Trees*, the three parameters of the system were set as described in the previous chapter.

Baseline	Evidence Trees	LLC
71.9 ± 1.0	86.7 ± 1.8	90.5 ± 1.2
83.9 ± 1.2	93.6 ± 1.8	96.2 ± 1.1

Table 4.1: Performance on the Stonefly9 dataset. The top row shows the results using only SIFT descriptors on a regularly sampled grid, and the bottom rows shows the results of combining SIFT descriptors applied to patches found by four different detectors. The error bars are computed as one standard deviation of accuracy over 3-fold cross validation.

Baseline	Evidence Trees	LLC
48.5 ± 2.6	77.4 ± 1.8	80.9 ± 2.3

Table 4.2: Performance on the EPT54 dataset. The error bars are computed as one standard deviation of accuracy over 3-fold cross validation.

	PCBR	Regular	Salient	Hesaff	All Detectors
Evidence Trees	88.6 ± 1.9	88.8 ± 1.8	88.9 ± 1.1	84.5 ± 2.8	93.6 ± 1.8
LLC	86.3 ± 1.7	90.1 ± 1.3	92.4 ± 1.1	84.0 ± 1.8	96.2 ± 1.1

Table 4.3: Classification accuracy of four Detector/descriptor combinations on STONEFLY9. The four detectors are the PCBR detector[17], the regular grid, the Kadir-Brady salient region detector [42], and the Hessian affine detector [60], We also show the result of combining all four detectors. The error bars are computed as one standard deviation of accuracy over 3-fold cross validation.

As shown in table 4.1 and table 4.2, both *Stacked Evidence Trees* and *LLC* achieve good performance on both datasets. Both methods beat the baseline by a large margin. The *LLC* method has the best performance on both datasets. It achieves an outstanding 96.2% accuracy on STONEFLY9 using all four detector types and 80.5% on EPT54. *Stacked Evidence Trees* achieve 93.6% accuracy on STONEFLY9 and 77.4% on EPT54.

Table 4.3 shows that combining the patches found by different detectors can improve the performance of both approaches. For *Stacked Evidence Trees*, random forests are

trained separately for the four detector/descriptor combinations, and the resulting class label histograms are concatenated before feeding them to the final stacked classifier. For LLC, one dictionary is generated for each of the four detector/desecriptor combinations, and the pooled image-level feature vectors are concatenated to form the feature vector for a linear SVM classifier.

4.2 Tests of the Image Resolution Hypothesis

In this section, we analyze how the resolution of images affects fine-grained classification accuracy. For generic objection categorization tasks, the images are usually of low resolution (300×200 for Caltech 101, 500×400 for PASCAL). However, our databases contain images of high resolution (2560×1920). In order to simulate low resolution images, we first build a scale pyramid using the original 2560×1920 images following the standard cascade Gaussian blurring and downsampling procedure [77]. Each image is downsampled to 50%, 25%, 12.5%, and 6.25% of the original size. At 12.5%, the size of the image is 320×240, which is about the same size as the images in popular generic object categorization databases. In order to ensure that we extract patches at exactly the same locations on the object at the different image resolutions, we then generate a downsized regular sampled grid of patches for each scale. The patch diameter is also shrunk accordingly. The SIFT descriptors are computed from the extracted patches.

There are two ways that image resolution could affect classification accuracy. First, the SIFT descriptors computed from lower-resolution images could be degraded in some way. Second, the classifier learned from those descriptors could be less accurate.

4.2.1 The Effect of Reduced Resolution on SIFT Descriptors

To explore the first issue, we evaluated the sensitivity of the SIFT descriptors to change of image resolution. Specifically, we compute a city block distance between the SIFT descriptors of the original images and the resolution-reduced images centered at the same locations:

$$dist = \sum_{j=1}^{D} |x(j) - y(j)|.$$

We compute the block distance over all SIFT descriptors of STONEFLY9 and report

the median distances.

Gaussian σ	0	2	4	8	16
Resolution	2560	1280	640	320	160
Median Block Distance	0	32	102	315	896

Table 4.4: A quantitative measure of the image downsizing effect on SIFT descriptors on STONEFLY9. The median city block distance of the SIFT descriptors at the same location in different scales is computed.

The SIFT sensitivity results are shown in table 4.4. The SIFT descriptor is quite robust to resolution reduction. The median block distance between the original descriptors and the 25% downsampled ones is only 102 for the 128 dimension descriptor vectors. Recall that each dimension of the SIFT vector is an integer between 0 and 255. Hence, at 25% downsampling, the SIFT descriptor values have changed by less than 1 part in 255 on average. This measure confirms that the orientation histogram of a SIFT descriptor is robust to scale changes. Occasionally (in less than 5% of the locations), we observe substantial change in SIFT vectors when they are located on abrupt edges of objects.

4.2.2 The Effect of Reduced Image Resolution on Classification Accuracy

To explore the second issue – inferior classifiers – we repeated the full classification experiments on the low resolution versions of both databases using *LLC* and *Stacked Evidence Trees*. The parameters and the evaluation metrics are the same as in the previous section.

The classification accuracy results are shown in figure 4.1 and figure 4.2. On the Stonefly9 dataset, we observe a dramatic drop of performance at extremely low resolution 160×120 (6.25% of the original size). But from the original size down to the 320×240 resolution (12.5%), we only observe mild performance degradation: from 90.5% to 89.6% correct for *LLC* and from 86.7% to 85.7% correct for *Stacked Evidence Trees*. However, on EPT54, there is a steady drop in accuracy as the resolution is reduced. This is probably because of the larger number of categories and greater similarity of taxa in EPT54.

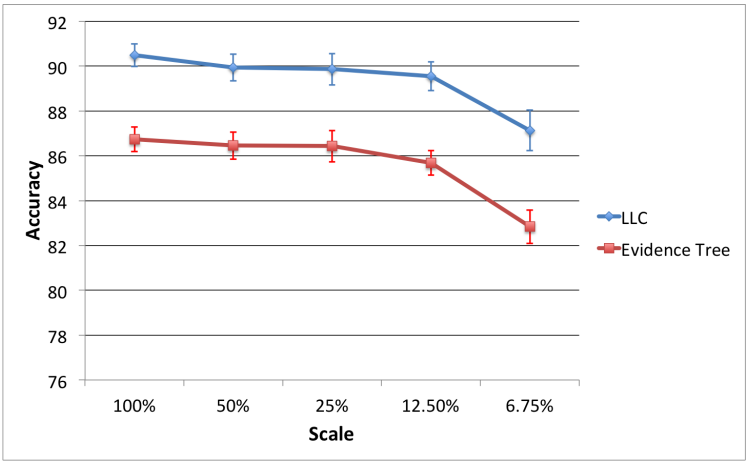

Figure 4.1: Performance under resolution reduction on STONEFLY9 dataset

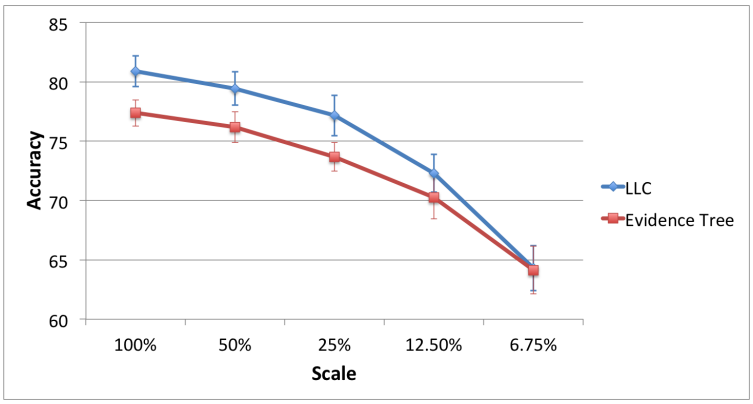

Figure 4.2: Performance under resolution reduction on EPT54 dataset.

4.3 Tests of the Information Extraction Hypothesis

According to the information extraction hypothesis, fine-grained classification requires extracting more information from the image than coarse-grained classification. For *Bag-of-Words* methods, the amount of information extracted is determined by (a) the number of image patches and (b) the size of the dictionary.

In our experiments, we held the number of patches constant and varied the size of the dictionary. We also varied the difficulty of the classification problem from coarse to fine by selecting pairs of arthropod taxa that have different "tree distance" in the

phylogenetic tree of life. Of course poor methods for encoding and pooling can cause loss of information, so we performed our experiments using *LLC*, which employs state-of-the-art methods for these steps.

4.3.1 Varying the Size of Dictionaries

To test our second hypothesis, we ran experiments by varying the size of the dictionary from 100 to 10,000 on both of our datasets and on a standard generic object categorization dataset, Caltech-101 [25]. Caltech-101 contains 9144 images in 101 generic object categories. Most images are centered on the object and have low resolution (300×200).

For STONEFLY9 and EPT54, we follow the same 3-fold cross validation process and report average image classification accuracy and standard deviation; for Caltech-101, we followed the standard experiment setup: train on 30 images per category and test on the rest. The evaluation metric is average over per-class classification accuracy and the standard deviation is computed via 10-fold cross validation.

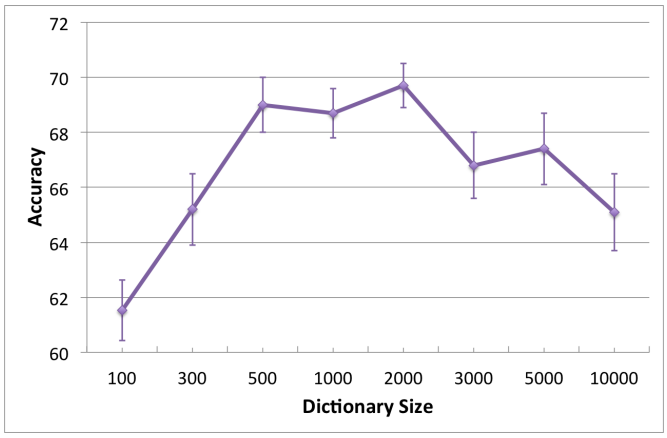

Figure 4.3: Performance of *LLC* under different dictionary sizes on the Caltech-101 dataset. The error bars are computed as one standard deviation of accuracy over 10-fold cross validation.

Figure 4.3 shows the result on Caltech-101. The performance peaks with a dictionary size of 2000 and then goes on decrease. This is likely the result of increased variance that results from unstable nearest neighbor assignment during the encoding step and overfitting of the classifier (due to very long pooled feature vectors) in the classification

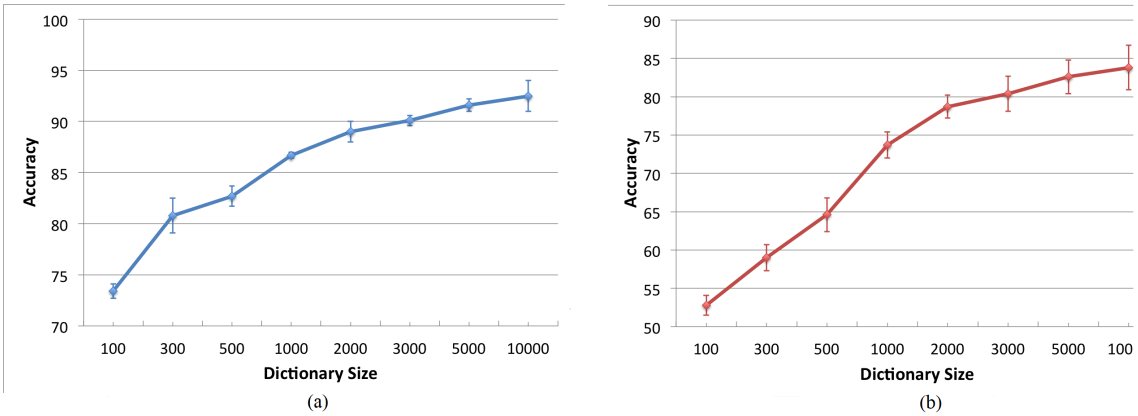

Figure 4.4: Performance of *LLC* under different dictionary sizes on (a) STONEFLY9 and (b) EPT54. The error bars are computed as one standard deviation of accuracy over 3-fold cross validation.

step.

Figure 4.4 shows the results on the STONEFLY9 and EPT54 datasets. Contrary to the previous experiment, the accuracy continues increase as the dictionary size grows for both datasets. This result supports our second hypothesis that larger dictionaries are desired for fine-grained object classification.

4.3.2 Varying the Difficulty of Classification Problems

We designed an experiment in which we varied the difficulty of the classification task from coarse-grained to fine-grained. In EPT54, the categories are divided into three Orders (Ephemeroptera, Plecoptera, and Trichoptera) in the tree of life. Each Order in turn is divided into Families, and each Family is divided into Genera. Each category in our classification tasks is a Genus. We can measure the "tree of life" distance between two categories in terms of the tree distance in the taxonomic tree. Two genera belonging to the same family have a tree distance of 2. Two genera belonging to the same order have a tree distance of 4. And two genera belonging to different orders have a tree distance of 6. We repeated the dictionary size experiments using pairs of categories separated by these different tree distances.

Visually, arthropods from different orders are very different and reflect the kinds of

gross differences that are typical of Caltech-101 and other generic object recognition databases, whereas arthropods from the same family can be very difficult to distinguish. According to our information extraction hypothesis, achieving high accuracy on categories at tree distance 6 will require much smaller dictionaries than achieving high accuracy on categories at tree distance 4, and those in turn will need smaller dictionaries than are required for categories at tree distance 2.

To make a fair comparison, we control the number of training examples in each class to be roughly around 250. Table 4.5 shows the exact number of training examples for each class. We performed the same kind of experiment as before by varying the dictionary size from 100 to 10,000 for all the selected taxonomic pairs. We report average image classification accuracy (with one-standard-deviation error bars) over 3-fold cross validation.

Code	**Order**	**Family**	**Genus**	**Training Size**
Calib	Ephemeroptera	Baetidae	Callibaetis	299
Fallc	Ephemeroptera	Baetidae	Fallceon	224
Amelt	Ephemeroptera	Ameletidae	Ameletus	297
Isogn	Plecoptera	Perlodidae	Isogenoides	231
Cerat	Trichoptera	Hydropsychidae	Ceratopsyche	299
Prpsy	Trichoptera	Hydropsychidae	Parapsyche elis	264
Micras	Trichoptera	Brachycentridae	Micrasema	295
Hydro	Trichoptera	Hydropsychidae	Hydropsyche	226

Table 4.5: The number of training examples for each class (at genus level) and its taxonomic information in the pairwise difficulty controlled experiments.

Figures 4.5, 4.6, and 4.7 show the results of two pairs at each tree distance. As we can see, the dictionary size required for top performance decreases along with the difficulty of the classification tasks. For the genus-level tasks, a dictionary size of 10,000 has the best performance; for the family-level tasks, only a moderate size of dictionary (2000-3000) is required; and for the order-level tasks, a small dictionary size of 500 already gives perfect results. This experiment further validates our second hypothesis that the harder the classification task is, or the more subtle distinctions among object categories are, the

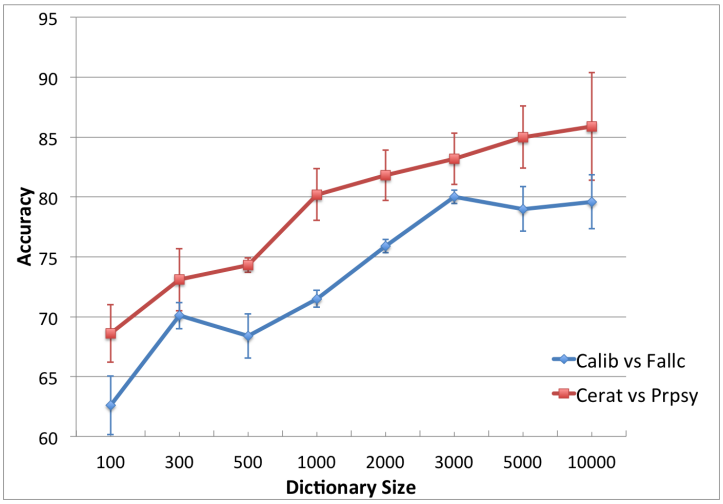

Figure 4.5: Performance under different dictionary sizes at genus level with phylogenetic tree distance of 2.

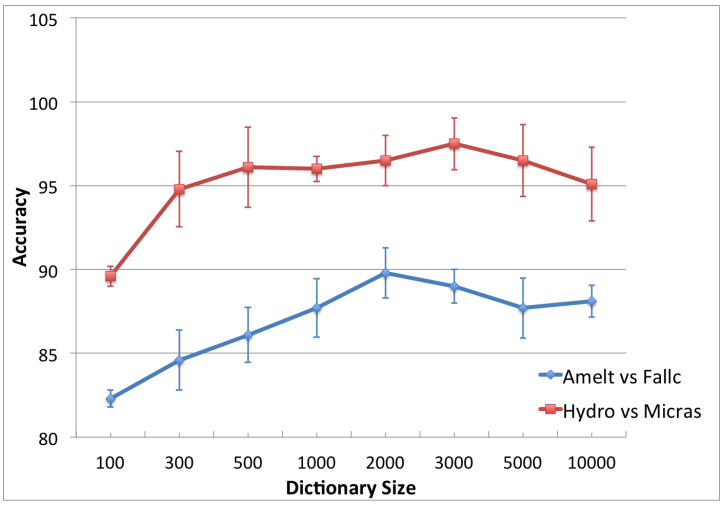

Figure 4.6: Performance under different dictionary sizes at family level with phylogenetic tree distance of 4.

larger dictionary needs to be.

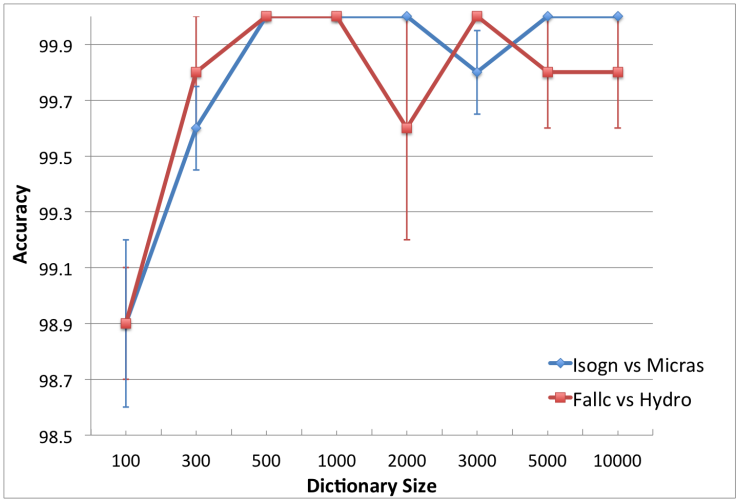

Figure 4.7: Performance under different dictionary sizes at order level with phylogenetic tree distance of 6.

4.4 Discussion

In this chapter, we tested both of our hypotheses on the requirements for successful fine-grained categorization. The effect of low resolution images is less significant than we expected. On STONEFLY9, the 12.5% resolution version of the original images only decreases the accuracy by less than one percent. However, the resolution reduction has much more effect on EPT54. We observe a steady drop in accuracy as the resolution is reduced. This shows the importance of high resolution images for larger fine-grained databases with greater similarity of taxa.

In the *BOW* model, the size of the dictionary is a critical parameter. Previous work [89] has shown that as the size of the dictionary grows, the classification accuracy first increases, then levels off, and finally decreases. This phenomenon suggests a bias-variance tradeoff. A small dictionary would have high bias (because of information loss in both the encoding and pooling steps) but good generalization. A large dictionary would generate long feature vectors which leads to high variance and hence poor generalization.

Our second experiment further shows that the optimal dictionary size that achieves the best performance is related to the difficulty of the classification task. A moderate-sized dictionary can perform well on a coarse-grained categorization problem, while a

much larger dictionary is required to achieve state-of-art accuracy on more difficult fine-grained classification problems.

From the experiments, we can draw some general conclusions about the design of fine-grained classification systems. The success of the system requires that a) the image capture process collects relatively high resolution images and b) the entire feature construction process extracts a large amount of information from the local descriptors while keeping good generalization capability for classification.

Chapter 5: Conclusions and Future Directions

This thesis studied the design, evaluation and analysis of learning algorithms for fine-grained object classification. Two fine-grained image databases of arthropods were introduced. A literature review on the development of *Bag-of-words* (BOW) approaches to object classification and our *stacked evidence tree approach* was presented. Benchmark results on both datasets were given, and further experiments were conducted to test our two hypotheses that a) high resolution images and b) more aggressive information extraction, such as finer descriptor encoding with larger dictionaries or classifiers based on raw descriptors, is required to achieve good performance in fine-grained categorization.

There are several important issues yet to be explored. First, empirical study has shown the tradeoff between dictionary size and generalization ability. However no systematical analysis in the bias-variance framework has been studied. Such an analysis would provide an estimate of the optimal dictionary size for a particular categorization problem.

Second, it has been shown that discriminative dictionaries outperform generic ones of the same size. However, since no efficient large scale discriminative dictionary learning approach has been developed, a randomly sampled dictionary of larger size can easily outperform a learned discriminative dictionary in state-of-the-art systems.

Finally, our *Stacked Evidence Trees* model is another way to extract information and construct features from the *bag-of-descriptors*. It provides an elegant dictionary-free approach to the image classification problem. However, its performance is inferior to the state-of-art *Bag-of-Words* methods. We can improve it in several ways: a) a more sophisticated pooling process can be developed to aggregate descriptor-level predictions into image-level feature vectors; b) an adaptive stopping criterion can be developed to control the depth of the trees, i.e. the amount of information extracted from descriptors, according to the difficulty of the classification problem.

Bibliography

[1] Ankur Agarwal and Bill Triggs. Hyperfeatures - multilevel local coding for visual recognition. In *ECCV*, pages 30–43, 2006.

[2] M. Aharon, M. Elad, and A. M. Bruckstein. The k-SVD: An algorithm for designing of overcomplete dictionaries for sparse representations. In *IEEE Transactions on Signal Processing*, volume 54(11), pages 4311–4322, 2006.

[3] Francis R. Bach and Gert R. G. Lanckriet. Multiple kernel learning, conic duality, and the SMO algorithm. In *ICML*, pages 6–13, 2004.

[4] Ron Bekkerman, Ran El-Yaniv, Naftali Tishby, Yoad Winter, Isabelle Guyon, and Andre Elisseeff. Distributional word clusters vs. words for text categorization. In *Journal of Machine Learning Research*, volume 3, pages 1183–1208, 2003.

[5] David M. Blei, Andrew Y. Ng, Michael I. Jordan, and John Lafferty. Latent Dirichlet allocation. In *Journal of Machine Learning Research*, volume 3, pages 993–1022, 2003.

[6] Hendrik Blockeel, Luc De Raedt, and Jan Ramong. Top-down induction of clustering trees. In *Proceedings of the 15th International Conference on Machine Learning*, pages 55–63, 1998.

[7] Oren Boiman, Eli Shechtman, and Michal Irani. In defense of nearest-neighbor based image classification. In *CVPR*, pages 1–8, 2008.

[8] A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. In *IEEE International Conference on Computer Vision*, pages 1–8, 2007.

[9] A Bosch, A Zisserman, and X Munoz. Representing shape with a spatial pyramid kernel. In *ACM ICVR*, pages 401–408, 2007.

[10] Y-Lan Boureau, Francis Bach, Yann LeCun, and Jean Ponce. Learning mid-level features for recognition. In *CVPR*, pages 2559–2566, 2010.

[11] Y-Lan Boureau, Nicolas Le Roux, Francis Bach, Jean Ponce, and Yann LeCun. Ask the locals: multi-way local pooling for image recognition. In *ICCV 11*, pages 2651–2658, 2011.

[12] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *ICML*, pages 111–118, 2010.

[13] Leo Breiman. Random forests. In *Machine Learning*, volume 45(1), pages 5–32, 2001.

[14] Adam Coates and Andrew Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *ICML*, pages 921–928, 2011.

[15] Gabriella Csurka, Christopher R. Dance, LixFan, Jutta Willamowski, and Cedric Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning Computer Vision, ECCV*, pages 59–74, 2004.

[16] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages 886–893, 2005.

[17] H. Deng, W. Zhang, E. Mortensen, T. Dietterich, and L. Shapiro. Principal curvature-based region detector for object recognition. In *CVPR*, pages 1–8, 2007.

[18] J. Deng, K. Li A. Berg, and L. Fei-Fei. What does classifying more than 10,000 image categories tell us? In *ECCV*, pages 143–156, 2010.

[19] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. In *Annals of Statistics*, volume 32, pages 407–499, 2004.

[20] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, New York, NY, 1993.

[21] Jan Eichhorn and Olivier Chapelle. Object categorization with SVM: kernels for local features. In *Technical Technical report, Max Planck Institute for Biological Cybernetics*, 2004.

[22] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. In *IEEE Transactions on Image Processing*, volume 15(12), pages 3736–3745, 2006.

[23] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results http://www.pascal-network.org/challenges/voc/voc2010/workshop/index.html. 2010.

[24] J. Farquhar, S. Szedmak, H. Meng, and J. Shawe-Taylor. Improving bag-of-keypoints image categorisation: Generative models and pdf-kernels. In *Technical report, University of Southampton*, 2005.

[25] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. In *CVPR Workshop on Generative-Model Based Vision*, 2004.

[26] Li Fei-fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 28, pages 594–611, 2006.

[27] Li. Fei-Fei and P. Perona. A Bayesian hierarchical model for learning natural scene categories. In *CVPR*, pages 524–531, 2005.

[28] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *In CVPR*, pages 264–271, 2003.

[29] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. Localizing objects with smart dictionaries. In *Proceedings of European Conference on Computer Vision*, pages 179–192, 2008.

[30] Shenghua Gao, Ivor Waihung Tsang, Liangtien Chia, and Peilin Zhao. Local features are not lonely – Laplacian sparse coding for image classification. In *CVPR*, pages 3555–3561, 2010.

[31] Peter Gehler and Sebastian Nowozin. On feature combination for multiclass object classification. In *ICCV*, pages 2169–2178, 2009.

[32] Bogdan Georgescu, Ilan Shimshoni, and Peter Meer. Mean shift based clustering in high dimensions: A texture classification example. In *ICCV*, pages 349–358, 2003.

[33] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. In *Machine Learning*, volume 63(1), pages 3–42, 2006.

[34] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Int. Conf. on Very Large Data Bases*, pages 518–529, 1997.

[35] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*, pages 1458–1465, 2005.

[36] G. Griffand, A. Holub, and P. Perona. Caltech-256 object category dataset. In *Technical Report UCB/CSD-04-1366, California Institute of Technology*, 2007.

[37] Gall J. and Lempitsky V. Class-specific Hough forests for object detection. In *IEEE International Conference on Computer Vision*, pages 1022–1029, 2009.

[38] Herve Jegou, Matthijs Douze, and Cordelia Schmid. On the burstiness of visual elements. In *CVPR*, pages 1169–1176, 2009.

[39] Yugang Jiang, Chongwah Ngo, and Jun Yang. Towards optimal bag-of-features for object categorization and semantic video retrieval. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 165–172, 2007.

[40] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *ECML*, pages 137–142, 1998.

[41] Frédéric Jurie and Bill Triggs. Creating efficient codebooks for visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 604–610, 2005.

[42] T. Kadir and M. Brady. Scale, saliency and image description. In *IJCV*, volume 45(2), pages 83–105, 2001.

[43] Koichi Kise, Kazuto Noguchi, and Masakazu Iwamura. Simple representation and approximate search of feature vectors for large-scale object recognition. In *BMVC*, pages 1–10, 2008.

[44] J. Koenderink and A. V. Doorn. The structure of locally orderless images. In *IJCV*, volume 31, pages 159–168, 1999.

[45] Josip Krapac, Jakob Verbeek, and Frédéric Jurie. Learning tree-structured descriptor quantizers for image categorization. In *British Machine Vision Conference*, pages 1–11, 2011.

[46] I. Lampl, D. Ferster, T. Poggio, and M. Riesenhuber. Intracellular measurements of spatial integration and the max operation in complex cells of the cat primary visual cortex. In *Journal of Neurophysiology*, volume 92, pages 2704–2713, 2004.

[47] N. Larios, H. Deng, W. Zhang, M. Sarpola, J. Yuen, R. Paasch, A. Moldenke, D. Lytle, S. Ruiz Correa, E. Mortensen, L. Shapiro, and T. Dietterich. Automated insect identification through concatenated histograms of local appearance features. In *Machine Vision and Applications*, volume 19(2), pages 105–123, 2008.

[48] Diane Larlus and Frédéric Jurie. Latent mixture vocabularies for object categorization. In *British Machine Vision Conference*, pages 5228–5235, 2006.

[49] Svetlana Lazebnik and Maxim Raginsky. Supervised learning of quantizer codebooks by information loss minimization. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 31(7), pages 1294–1309, 2009.

[50] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng. Efficient sparse coding algorithms. In *NIPS*, pages 801–808, 2007.

[51] B. Leibe, K. Mikolajczyk, and B. Schiele. Efficient clustering and matching for object class recognition. In *BMVC*, pages 1–10, 2006.

[52] Xiao-Chen Lian, Zhiwei Li, Bao-Liang Lu, and Lei Zhang. Max-margin dictionary learning for multiclass image categorization. In *ECCV*, pages 157–170, 2010.

[53] Bing Liu, Yiyuan Xia, and Philip S. Yu. Clustering through decision tree construction. In *SIGMOD-00*, pages 20–29, 2000.

[54] David G. Lowe. Distinctive image features from scale-invariant keypoints. In *IJCV*, volume 60(2), pages 91–110, 2004.

[55] Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Discriminative learned dictionaries for local image analysis. In *CVPR*, pages 1–8, 2008.

[56] Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Supervised dictionary learning. In *NIPS*, pages 1033–1040, 2008.

[57] Subhransu Maji, Alexander C. Berg, and Jitendra Malik. Classification using intersection kernel support vector machines is efficient. In *CVPR*, pages 1–8, 2008.

[58] S. Mallat and Z. Zhang. Matching pursuit in a time-frequency dictionary. In *IEEE Transactions on Signal Processing*, volume 41, pages 3397–3415, 1993.

[59] G. Martinez-Munoz, W. Zhang, N. Payet, S. Todorovic, N. Larios, A. Yamamuro, D. Lytle, A. Moldenke, E. Mortensen, R. Paasch, L. Shapiro, and T Dietterich. Dictionary-free categorization of very similar objects via stacked evidence trees. In *CVPR*, pages 549–556, 2009.

[60] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *ECCV*, pages 128–142, 2002.

[61] Frank Moosmann, Eric Nowak, and Frédéric Jurie. Randomized clustering forests for image classification. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 30(9), pages 1632–1646, September 2008.

[62] Andrew Y. Ng and Michael I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *NIPS 14*, pages 605–610, 2002.

[63] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, pages 2161–2168, 2006.

[64] Eric Nowak, Frédéric Jurie, and Bill Triggs. Sampling strategies for bag-of-features image classification. In *European Conference on Computer Vision*, pages 490–503, 2006.

[65] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? In *Vision Research*, volume 37, pages 3311–3325, 1997.

[66] Andreas Opelt, Axel Pinz, Michael Fussenegger, and Peter Auer. Generic object recognition with boosting. In *PAMI*, volume 28, pages 416–431, 2006.

[67] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques. In *EMNLP*, pages 79–86, 2002.

[68] Florent Perronnin. Universal and adapted vocabularies for generic visual categorization. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 30(7), pages 1243–1256, 2008.

[69] Florent Perronnin, Jorge Sanchez, and Yan Liu. Large-scale image categorization with explicit data embedding. In *CVPR*, pages 2297–2304, 2010.

[70] James Philband, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Lost quantization: Improving particular object retrieval large scale image databases. In *CVPR*, pages 1–8, 2008.

[71] P. Quelhas, F. Monay, J. Odobez, D. Gatica-perez, T. Tuytelaars, and L. Van Gool. Modeling scenes with local descriptors and latent aspects. In *ICCV*, pages 883–890, 2005.

[72] Pedro Quelhas, Florent Monay, Jean-Marc Odobez, Daniel Gatica-Perez, and Tinne Tuytelaars. A thousand words a scene. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1575–1589, 2005.

[73] J.R. Quinlan. *C4.5: Programs for Empirical Learning*. Morgan Kaufmann, San Francisco, CA, 1993.

[74] Rajat Raina, Alexis Battle, Honglak Lee, Benjam Packer, and Andrew Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the Twenty-fourth International Conference on Machine Learning*, pages 759–766, 2007.

[75] Gerard Salton and Christopher Buckley. Term-weighting approaches automatic text retrieval. In *Information Processing and Management*, pages 513–523, 1988.

[76] Cordelia Schmid. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, pages 2169–2178, 2006.

[77] Linda Shapiro and George Stockman. *Computer Vision*. Prentice Hall, 2001.

[78] Michael Shindler, Alex Wong, and Adam W. Meyerson. Fast and accurate k-means for large datasets. In *NIPS 24*, pages 2375–2383, 2011.

[79] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, pages 1–8, 2008.

[80] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering object categories image collections. In *Technical Report MIT-CSAIL-TR-2005-012, Massachusetts Institute of Technology*, 2005.

[81] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching videos. In *ICCV*, pages 1470–1477, 2003.

[82] Josef Sivic, Bryan C. Russell, Alexei A. Efros, Andrew Zisserman, and William T. Freeman. Discovering objects and their location images. In *ICCV*, pages 370–377, 2005.

[83] Noam Slonim and Naftali Tishby. Agglomerative information bottleneck. In *NIPS*, pages 617–623, 1999.

[84] Noam Slonim and Naftali Tishby. The power of word clusters for text classification. In *23rd European Colloquium on Information Retrieval Research*, pages 1–12, 2001.

[85] Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. In *ACM SIGIR*, pages 368–377, 2000.

[86] Antonio Torralba, Kevin P. Murphy, and William T. Freeman. Sharing features: Efficient boosting procedures for multiclass object detection. In *CVPR*, pages 762–769, 2004.

[87] Tinne Tuytelaars. Vector quantizing feature space with a regular lattice. In *ICCV*, pages 1–8, 2007.

[88] van Gemert J. C., J. M. Geusebroek, C. J. Veenman, and A. W. M. Smeulders. Kernel codebooks for scene categorization. In *ECCV*, pages 696–709, 2008.

[89] van Gemert J. C., C. J. Veenman, A. W. M. Smeulders, and J. M. Geusebroek. Visual word ambiguity. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 32(7), pages 1271–1283, 2010.

[90] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. In *IJCV*, volume 62, pages 61–81, 2005.

[91] Andrea Vedaldi, Varun Gulshan, Manik Varma, and Andrew Zisserman. Multiple kernels for object detection. In *ICCV*, pages 886–893, 2009.

[92] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, pages 511–518, 2001.

[93] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *CVPR*, pages 3360–3367, 2010.

[94] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *ICCV*, pages 1800–1807, 2005.

[95] J. Wright, A.Y. Yang, A. Ganesh, S.S. Sastry, and Y. Ma. Robust face recognition via sparse representation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 31, pages 210–227, 2009.

[96] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1794–1801, 2009.

[97] Jianchao Yang, Kai Yu, and Thomas Huang. Efficient highly over-complete sparse coding using a mixture model. In *ECCV*, pages 113–126, 2010.

[98] Jianchao Yang, Kai Yu, and Thomas Huang. Supervised translation-invariant sparse coding. In *CVPR*, pages 3517–3524, 2010.

[99] Jun Yang, Alexander G. Hauptmann, Yugang Jiang, and Chongwah Ngo. Evaluating bag-of-visual-words representations scene. In *Workshop on multimedia information retrieval*, 2007.

[100] Liu Yang, Rong Jin, Rahul Sukthankar, and Frédéric Jurie. Unifying discriminative visual codebook generation with classifier training for object category reorganization. In *CVPR*, pages 1–8, 2008.

[101] Kai Yu, Tong Zhang, and Yihong Gong. Nonlinear learning using local coordinate coding. In *NIPS*, pages 689–696, 2009.

[102] Hao Zhang, Alexander C. Berg, Michael Maire, and Jitendra Malik. SVM-knn: Discriminative nearest neighbor classification for visual category recognition. In *CVPR*, pages 2126–2136, 2006.

[103] Wei Zhang, Akshat Surve, Xiaoli Fern, and Thomas Dietterich. Learning non-redundant codebooks for classifying complex objects. In *ICML*, pages 1241–1248, 2008.

[104] Xi Zhou, Kai Yu, Tong Zhang, and Thomas S. Huang. Image classification using super-vector coding of local image descriptors. In *ECCV*, pages 141–154, 2010.

APPENDICES

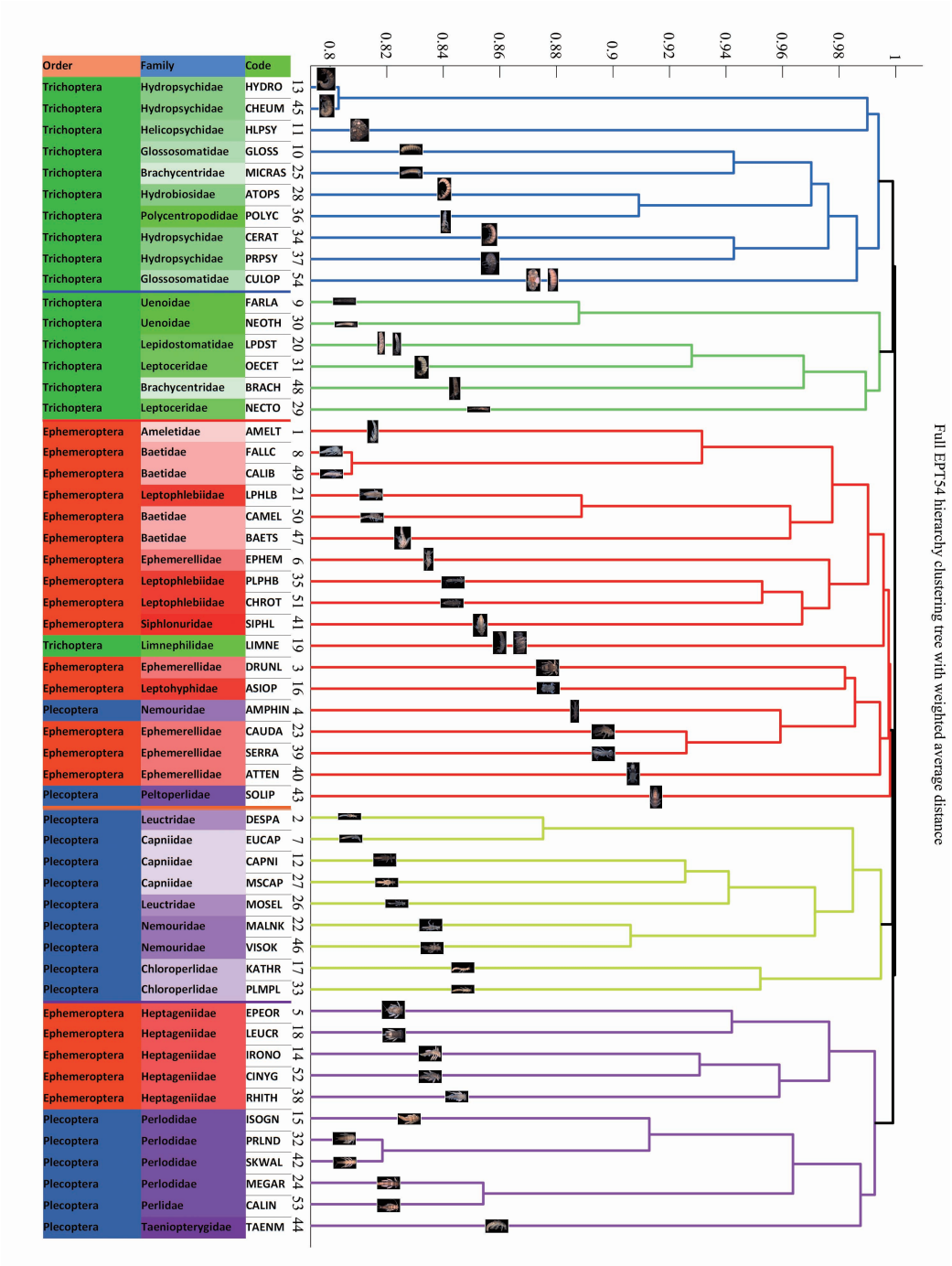

Figure 1: Visualization of confusion matrix of EPT54 dataset. The figure shows a hierarchical clustering tree based on the class-wise distances. The distances are computed as the reciprocal of the off-diagonal values of the confusion matrix. The table at left shows the hierarchy of biological classification for each category in the dataset. The three orders in the datasets are color coded: Ephemeroptera (Mayflies) in red, Plecoptera (Stoneflies) in blue, and Trichoptera (Caddisflies) in green.