Enhancing Bag of Words Image Representations

Thesis submitted in partial fulfillment
of the requirements for the degree of

Ms by Research
in
Computer Science

by

Vinay Garg
200702053
vinay.garg@research.iiit.ac.in

Center for Visual Information Technology
International Institute of Information Technology
Hyderabad - 500 032, INDIA
July 2013

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Enhancing Bag of Words Image Representations" by Vinay Garg, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date

Adviser: Prof. C. V. Jawahar

To My Family.

Acknowledgments

First and foremost I would like to express my sincere gratitude to my advisor Prof. C. V. Jawahar for his continuous guidance, support, patience and motivation. His immense knowledge of this area of research and guidance has helped me throughout the course of this research. The best thing about him is that he knows how to bring the best in a person. He always taught me to go for that extra bit which actually made the difference in the end. I could not have imagined a better advisor and mentor for my MS study.

Besides my advisor, I was fortunate enough to have friends and colleagues at CVIT who have played different roles at different times. They were always there with me both in my good and bad times. In case I had any doubt or was stuck at some point they were always there to discuss and help me out. I would like to thank - Siddhartha Chandra, Abhinav, Mayank Juneja, Yashaswi, Siddharth Kherada, Shirkant Baronia, Rohit Nigam, Harshit Sureka, Rohit Gautam, Shashank Mujumdar, Srijan, Nisarg, Nagender, Anand, Omkar, Jay, Ankit, Chandrashekar and Harshit Agrawal. I am also grateful to Mr.R. S. Satyanarayana for his administrative support. I would like to thank Phani whose presence at CVIT has helped students in more ways than one. Many thanks to Prof. P.J. Narayanan, Prof. Jayanthi and Prof. Anoop for their encouraging presence and for providing an environment conducive to learning of the finest quality at CVIT.

I would like to thank my friends Manish, Karan, Dhwaj, Vinit, Sourabh, Gaurav, Akshat, Ankit, Piyush, Ammar, Rahul, Rakshit, Ankur, Sankalp for making this time worth remembering.

I am thankful to my brother, my senior and my friend Aditya Singal for being there with me from the very first day of my college life. He has inspired me a lot, not only professionally but personally as well. His calm and composed attitude towards the problems which life imposes on us, has helped me a lot not only during my research but also during my graduation.

Finally, I would like to acknowledge the support of my parents in my academic and research pursuits. Both of them were, are and will always be a continuous source of motivation and inspiration in all the ventures of my life. Their belief in my abilities have always act like a catalyst for me to move forward in the direction of my goals and shaping my career and life.

Many thanks to everyone else who affected my life in any way, and wasn't acknowledged personally above.

Abstract

Bag of visual words inspired from text classification has been used extensively for solving various computer vision tasks such as image classification, image retrieval, image recognition, etc. In text classification the vocabulary set is a fixed finite set of words present in a particular language, which is not the case in visual domain. There is no fixed set of visual words in visual domain. Infact there is no concept of words in images. This is due to the fact that complexity in visual domain is so high as even a small change like rotation, translation, change of view angle, lightning, etc. will have a huge impact on the information perceived by the machines for these images. Even though for us these changes will not make any difference but for machines, these all will be different images. So to overcome this problem, vision community has defined the concept of visual words, which are analogous to textual words. But the visual words are not very well defined due to vast domain of the visual data as compared to textual data which have finite number of words. Using these visual words we create the image representations as the frequency of these visual words in that image and in turn use these representations to do various vision tasks.

In this thesis we aim at improving these image representations, as the accuracy and performance of various vision models depends directly on quality of image representations given to them as input. We started with the traditional bag of visual words, study various practical issues and drawbacks in that approach, tried refining one of the various steps of pipeline at a time. Doing so we devised novel strategies to overcome some of the issues which we faced while studying the traditional approaches. In the approaches which we applied to solve the issues, we used various parameters which needed fine tuning and we have discussed the effect of each parameter in detail with the empirical results to support our hypothesis and finally conclude that our representations were better as compared to the various traditional approaches presently used.

To solve the problem of information loss due to hard assignments in traditional bag of words, we analyzed various soft assignment techniques. On replacing the hard assignments with soft assignments, we found that classification results improved drastically. Even while comparing different soft assignment techniques among themselves, we found that absolute soft assignments are better as compared to relative soft assignments. We demonstrate the superiority of our approaches on various popular datasets.

Recently vision community showed that Fisher vector image representations outperform Bag of words representations. This boost in performance is because, firstly Fisher vectors use soft assignments and secondly, they reduce the information loss by capturing the deviations of each visual feature

from the mean. However, like any other approach they also have their share of drawbacks. Size of Fisher vector image representations is huge and they are not inherently discriminative. So, we introduced sparseness to reduce the effective size of the representations, and added some class information to make them discriminative. These additions reduces the high storage requirements, but at the same time adding class information also increases the performance. To demonstrate these findings, we tested it on various datasets which supported our claim.

Driven from the hypothesis that improving individual steps of various image representations pipelines will improve the final image representation, we have tried various techniques to refine these steps, which in turn will improve the performance of our model. After improving the final step of creating image representations from the visual words, we tried improving the set of visual words (or vocabulary) itself. We found that most of the visual words which we use for building the image representations are not actually useful and there are a lot of redundant words present in the visual vocabulary. So, to further improve our representations, we devised a novel technique to combine various visual words from different types of vocabularies (which will capture different type of information from a given set of images), and combine the best of them to get the final global vocabulary, which will be used to get the final image representations. Again, we used benchmark datasets to demonstrate that our hypothesis is correct.

In this thesis, we have tried our hands to solve the classification task in a better way. Although we are not able to solve the research tasks perfectly (i.e. reaching the perfect score), but we hope that our findings will atleast give a starting point for various new directions which will lead us to our ultimate goal of replicating the human vision.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

With advance in the modern day technology, the amount of visual data has increased manifold. This increase is so much that processing and analyzing it manually is practically not possible. E.g. If we want to extract some information manually from a surveillance camera video which has information for past 7 days, then one has to sit for another 7 days to analyze and process that video. In order to make our lives simpler researchers have tried their hands on automating the process of analyzing this huge visual data and extracting useful information from it by trying to duplicate the abilities of human vision with the help of machines. But in order to use machines for doing our job, we need to convert this data from the format which humans understand to a format which machines can understand. This branch of science which deals with methods for acquiring, processing, analyzing and understanding visual information and in general high dimensional data from real world in order to produce numerical or symbolic information is known as "Computer Vision". In computer vision cameras serve the purpose of eyes, which will receive information from the outer world, high configuration processors serve the purpose of our brain, which will process the huge information, and various computer vision algorithms try to replicate the complex processing happening in our brain to analyze the information which our brain does effortlessly.

Replicating human brain as such is not a trivial task. The various things which our brain can do, say for example recognizing people in their social media images, is so easy for us that we will not even realize the complexity behind it. Even if they are in different poses, standing or sitting, front pose or side pose, partially or fully visible, or even occluded by some other object, we will not face any issues in that. For a machine doing similar things is not that trivial as all of the cases mentioned above are different problems all together for a machine's perspective.

Because of this limitation of machines, where they can understand information only in terms of numerical data, various problems are defined in the field of computer vision. Some of the popular ones are defined in the next section.

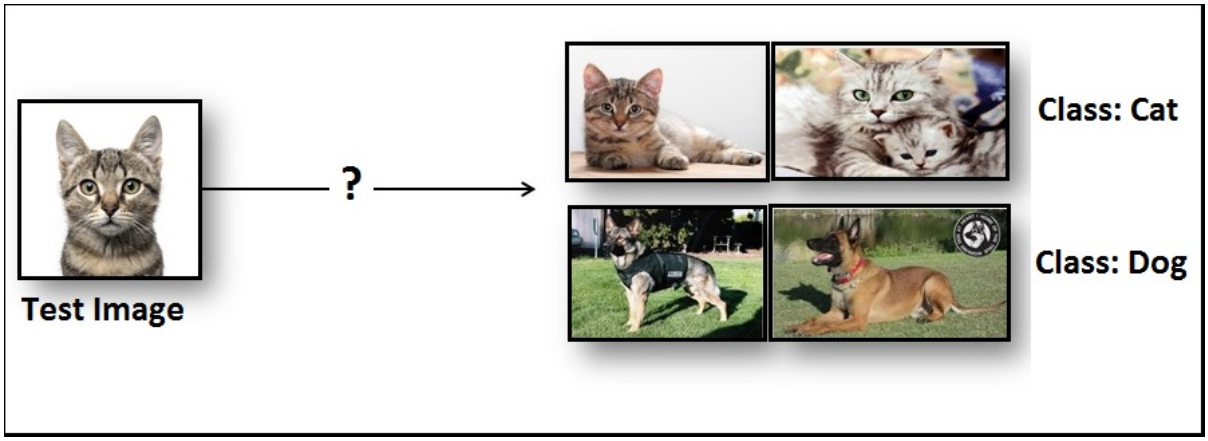


Figure 1.1 Example of Image Classification: Here we try to classify the test image into one of the two classes (cat or dog) using a previously trained model.

1.2 Computer Vision Tasks

Computer Vision tasks mainly deal with understanding the visual data which can be of various forms like images clicked from a digital camera, videos, high dimensional data from medical scanner or views from multiple camers. But understanding the visual data is a very broad concept as it depends on the problem in hand, which defines how correctly our system is able to understand the information in given visual data.

1.2.1 Image Classification

Classification refers to the process of categorizing different things based on some relationships defined by the user. Following the same definition, image classification is the process of categorizing similar images into a group or a category or a class. The similarity can be defined differently depending on the problem we want to solve. And the definition of similarity will define the complexity of our problem. E.g. Classification between say animals and landscapes is easier than classification between a dog and a bird which in turn will be easier than the classification between a dog of one kind and a dog of another kind. Various applications of image classification are scene classification, action classification, object detection, image retrieval, etc.

1.2.2 Object Detection

Another popular task of computer vision is “Object Detection”. Object detection refers to the process of detecting a particular object like humans, buildings, cars or animals, in a given digital image or video source. Classification can be considered as a related problem of object detection as we can say that classification is a image detection with the similarity criteria that a particular object is present in given set of images or not. In object detection we not only verify that a particular object is present in the given

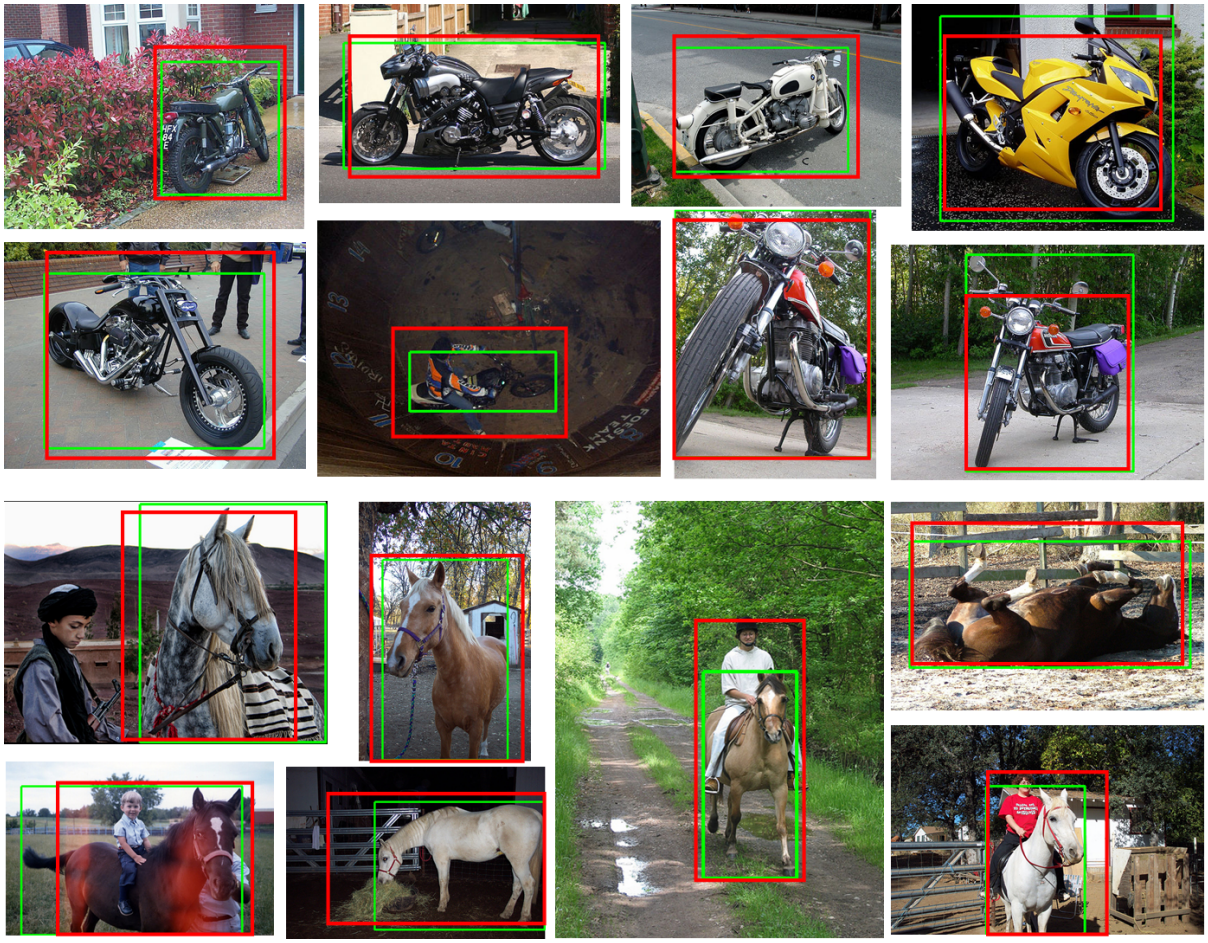


Figure 1.2 Object Detection: Example of object detection where we try to localize the position of objects of interest in a given image. The objects of interest in these example images are bikes and horses.

image or not, but we also localize its position in the image. Various applications of object detection includes face detection, pedestrian detection, etc.

1.2.3 Image Retrieval

Image retrieval refers to the system for browsing, querying, and retrieving images of similar kind from a large database of digital images. In image retrieval user will query for a particular image or a part of an image and the system will return a set of similar images from the database based on a similarity criteria ranked on the basis of how similar are the retrieved images to the query image. This is widely used in the field of image search engines, where we are not sure of the exact image but if we give somewhat similar image as a query we will be able to find the image we were looking for. Another application of image retrieval is automating annotation of images. If we know annotation for a particular image, we can query for the similar images in the database and give them the annotation based

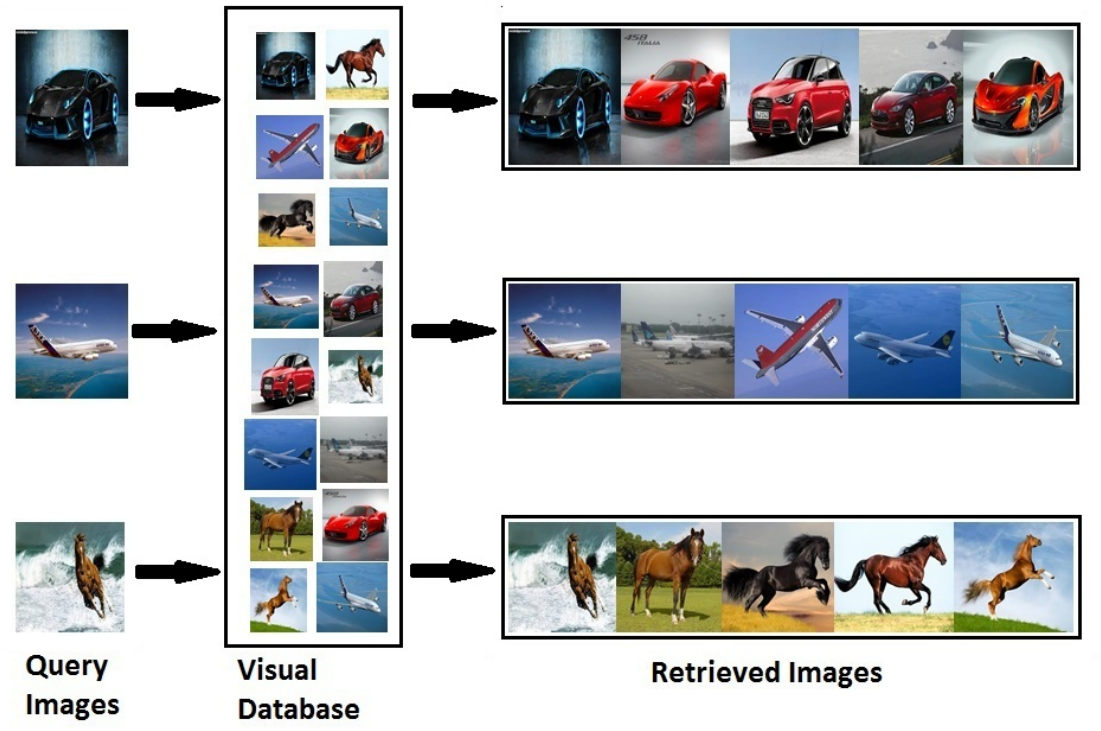


Figure 1.3 Example of Image Retrieval: Here when we query the database using the query image, images which are similar to the query image are retrieved and ranked according to the similarity between the query image and the retrieved images.

on their similarity with the original image. This is very useful as with time the amount of visual data is increasing exponentially and manual annotation of all the images is almost practically impossible.

1.3 Bag of Visual Words Representations

Bag of visual words model is being used extensively by the vision community to solve various problems in hand. BOW representation is inspired from the word-document representations of images. The first step of the bag of visual words based approach is the computation of local feature descriptors like SIFT [26] for a set of image patches. These patches can be either at the key-point locations or densely sampled on a regular grid of the image. These set of local feature descriptors are quantized using a clustering technique. This step is referred to as vocabulary building step. The resultant set of cluster centers is referred to as visual vocabulary or codebook, and each cluster center is individually called a “visual word” or “codeword”. The generated visual vocabulary is then used for assigning the nearest visual word for each of the local feature descriptors in a given image. This step is referred to as

assignment step. The histogram of visual words in a given image is used as its representation for image classification, retrieval or recognition tasks.

In chapter 2 we will start with the detailed explanation of BOW model, followed by various practical issues. Further we will discuss various advances in this field and how the vision community have addressed some of the practical issues.

1.4 Focus of the Thesis

We can do all these tasks mentioned above without any efforts and are so normal for us that we wont even realize the complexity behind them. Not only ease but the accuracy with which human brain does these things is astonishing. Till date we are not able to replicate these complex operations perfectly. This is the reason that performance of machines in all of the problems mentioned above has never reached a perfect score of cent percent.

All the problems in computer vision whether it is classification, recognition, detection, or any other problem boils down to following smaller problems:

1. Extract information from the visual data in such a manner that it gives us better insights about the underlying patterns

2. Based on the information extracted in the previous step, learn/ train a model which will help us make better decision in future for the new/unseen data.

Both are separate research fields in themselves. In this thesis, we have tried to address the problem of extracting information from the visual data in a better way, which in turn will help us in making better decisions. Below is the overview of the contributions we have made in this field which are described in detail in the subsequent chapters.

A Soft Clustering Based Exposition of BOW: Traditionally, Bag of visual words uses describes an image as a bag of discrete visual codewords i.e they use hard assignments for building the image representations. The drawback of hard assignments is that every visual feature in an image is assigned to single codeword, which leads to a loss of information regarding the other relevant codewords that can represent the same feature. This loss of information can be useful for understanding the underlying structure of images. So, we replaced it with soft assignments and based on that constructed new image representations which are better since they preserve more information.

Sparse Discriminative Fisher Vectors: Our findings in chapter 3, show that soft assignments have superior performance over hard assignments. Fisher vector representations have been shown to outperform other global representations on most benchmark datasets. However, the Fisher vector representations are huge and the representation size increases linearly with the vocabulary size. Recent findings report that the classification performance is proportional to the vocabulary size. Computational and storage requirements on the other hand discourage the use of arbitrarily large vocabularies. Also

Fisher vectors are not inherently discriminative. We devise a novel approach compute sparse discriminative Fisher representations.

Discriminative Visual Words: Recent literature describes a variety of codebook building strategies, but there is no clear consensus on how to learn good, discriminative vocabularies. Researchers have often pondered if there is a universal vocabulary building approach. While the search for a universal vocabulary continues, we address the problem of ascertaining the discriminative powers of visual words in a codebook. This knowledge can be used to enhance a vocabulary by rejecting visual words with little discriminative power, or select an optimal set of vocabularies. It may be pointed out that while this form of subset selection is an NP hard problem per se due to the combinatorial explosion, our approach is greedy and linear in the number of visual words.

1.5 Organization of the Thesis

The organization of the this thesis is as follows. In Chapter 2, we explain the traditional bag of visual words approach for image classification. We also explain about the related work in the same area and various practical issues with the traditional approach, which guided the whole research done in this thesis. In Chapter 3, we address the problem of information loss due to hard assignments in traditional bag of words and how we can overcome it upto a certain extent using soft assignments. In Chapter 4, we explain the Fisher vector image representations, which have revolutionized the research in the field of computer vision. We also address some of the issues with the Fisher vectors and introduce a novel approach to compute sparse discriminative Fisher vectors, with a little computational overhead. In Chapter 5, we introduce an approach to select a subset of visual words from which are more discriminative for a particular class by rejecting the ones which are having very little discriminative power. Finally we conclude in Chapter 6 with all our findings throughout the course of this research.

Chapter 2

Bag of Visual Words: Background and Related Work

2.1 Bag of Visual Words

Bag of Words representation (also known as bag-of-features) was first proposed for solving problems related to Natural Language Processing(NLP) and Information Retrieval. In NLP domain, a text document using this model is represented as an unordered collection of words, disregarding the grammar rules and order of words. For example, "this is a document" and "document this a is" are considered to be same under this model as in both the documents, each word

$$\{\text{“}this\text{”}, \text{“}is\text{”}, \text{“}a\text{”}, \text{“}document\text{”}\}$$

is occurring the same number of time which is once.

In computer vision, BoW representation is based on the analogy of considering "images" as "documents" and "image patches" as "words in the documents". These image patches are specifically represented by using visual words, which are formed by vector quantizing the visual features (color, texture, etc.,) like local region descriptors. Construction of "Bag-of-Words" representation from the images mainly involves the following steps:

- Visual Features Extraction
- Vocabulary Construction / Vector Quantization
- Image Representation

Figure 2.1, shows the complete pipeline with various steps involved in it.

2.1.1 Visual Feature Extraction

Nowadays with advancement in the technology, visual data what we are getting is huge in terms of information that needs to be processed. These days, a 5MP camera is normally used for capturing images, which imply that image taken by this camera will have 5 million pixels in it. Now if we want

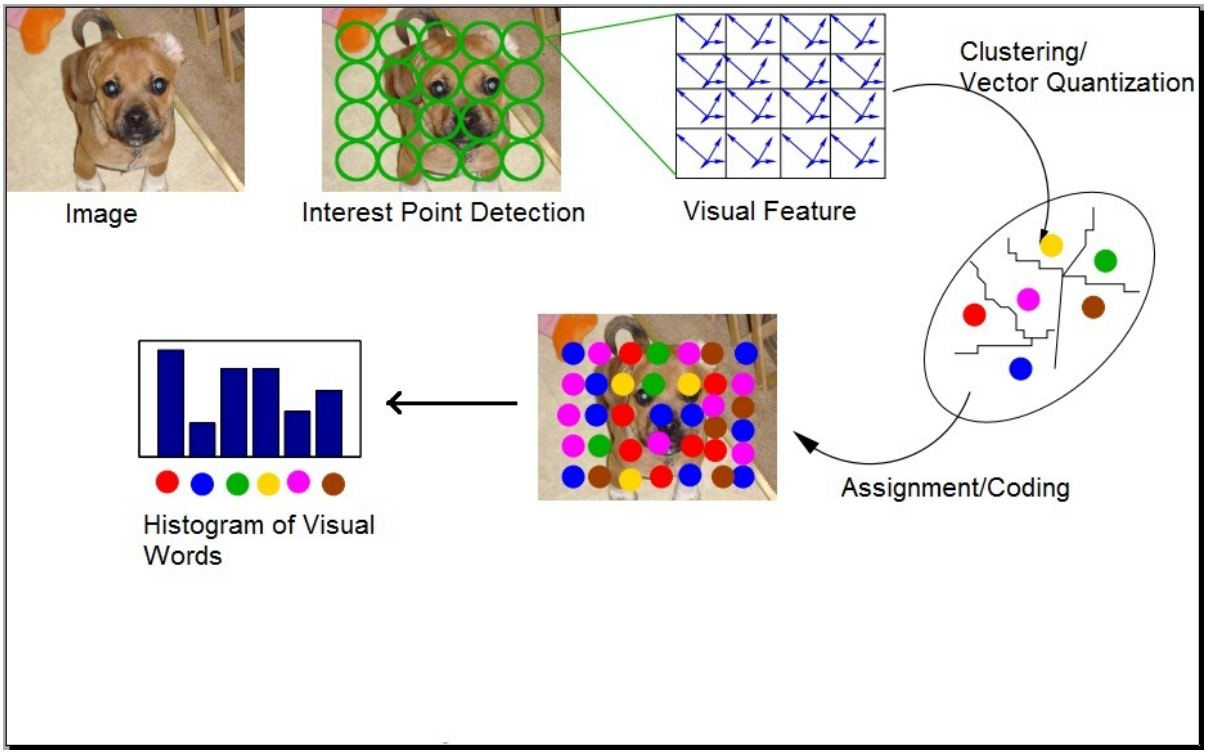


Figure 2.1 Figure showing the various steps involved in the complete "Bag of Visual Words" pipeline. (a) First we extract interest points from an image on which we will extract features. (b) Then these features are clustered together to form the visual vocabulary. (c) Using this visual vocabulary, we create the final image representations.

to work on each pixel individually, not only it is computationally very expensive, but also redundant as only a minor portion of this data have useful information rest all is irrelevant to us. So to counter this problem we compute "Visual Features" which will represent a image instead of these pixels. This process of extracting visual features is known as "Visual Feature Extraction" or simply "Feature Extraction". Hence by computing visual features we reduce our problem from pixel level to these visual features level. Depending upon whether these features are representing an image as whole or only a local region of an image, we can classify them into *global* or *local* visual features respectively. While global feature extraction gives a single representation for a particular image, local feature extraction on the other hand can be further divided into following two steps

- **Interest Point Detection:** Interest points are pixels of an image around which visual features (local) will be computed. In this process we basically eliminate the redundant/irrelevant pixels and keep only those which we think captures more useful information e.g. in an image corners, edges, shapes are the regions of interest which should contain most of the information, while the flat patches are not of much importance. Today in literature there exist several methodologies for computing these interest points which range from (a) choosing random pixels (random sam-

pling), (b) choosing every pixel at a fixed interval (dense sampling), or (c) by applying complex interest point detection algorithms like corner detection, shape detection, etc. Each of the above mentioned methods have their pros and cons, and depending upon the problem we can chose any of these methods for detecting interest points in an image. Researchers have shown that dense sampling generally outperforms the others in most of the problems, hence it is most commonly used.

- **Feature Descriptor:** Once we have computed the interest points by the most suitable method for interest point detection, next step is to compute the visual features on these interest points. These visual features should have various properties which should be satisfied in order to serve our purpose. Firstly, these visual features should be compact as computation complexity is one of the main reasons for moving from pixel space to visual feature space. Secondly, these features should be invariant to various external factors like illumination, orientation, scale, translation, etc. Some of the best performing local features used today are SIFT, SURF, etc. For computing traditional bag of words representations, we need local features to be computed at various points in an image. But depending upon the requirements we might also use global features on a patch (which will serve as the pseudo image) around the interest point. Like local features some of the global features which are used extensively in various computer vision tasks are HOG, GIST, etc.

2.1.2 Vocabulary Construction / Vector Quantization

Visual features gives us a way to represent images in a compact manner without losing any relevant information. Now this image which is represented by the visual features is analogous to a document in text classification. Advantage in text classification over image classification is that the vocabulary is defined precisely in text classification. In image classification we do not have a fixed representation for visual words which will serve the purpose of words in text classification. We can take each visual feature as a visual word, but this method would still be inefficient. After a lot of research, people have come up with idea that visual features which are similar to each other in one way or the other should be grouped together and this group/cluster will serve as the visual word.

2.1.2.1 Clustering

Clustering refers to a task of grouping similar objects together. This group is known as a cluster of similar objects and the process is known as clustering. The problem here is that the notion of cluster is not defined properly. It can vary depending upon the problem in hand e.g. in the set of natural numbers we can have clusters of numbers based on the distance from a particular number, or a cluster of even numbers, or a cluster of prime numbers and so on. Similarly in objects, say cars, we can have clusters of cars either on the basis of their model, year, color, brand, or all of them together. Depending upon how we want to define our clusters we can be broadly classified the existing clustering algorithms into, (a)

connectivity based clustering, (b) centroid based clustering, (c) distribution based clustering, (d) density based clustering, (e) subspace based clustering.

Other than this we can also have supervised, semi-supervised, unsupervised clustering depending upon the degree of information given by the experts in clustering. In supervised clustering we have complete external information regarding the clusters, where as in semi-supervised clustering this information is limited to only a subset of the data and for the rest of the data we have to learn and in unsupervised clustering we dont have any external information and everything i.e. relation between the objects has to be learn on our own. Once we have a clusters using algorithm of our choice, we need a cluster representative, which will represent the cluster as whole and define all the essential properties of that cluster.

2.1.2.2 K-Means Clustering Algorithm

Due to its simplicity, K-Means clustering algorithm is the most popular clustering algorithm. It takes as an input N data points and outputs K disjoint subsets(clusters) of the input data. Each cluster is represented by a single data point also known as the centroid of the cluster, which is the mean of all the data points which belong to that particular subset. A data point belongs to a particular cluster if it is closer to centroid of that particular cluster as compared to centroids of other clusters. It tries to minimize distortion, which is defined as the sum of squared distances between each data point and its dominating centroid. Mathematically, we can define K-means algorithm as follows.

Given $X = \{x_1, x_2, ..., x_N\}$, data points each of dimension d and we want to divide them into K clusters $C = \{c_1, c_2, ...c_K\}(K \leq N)$ so as to minimize the distortion i.e. sum of squared distances between each data point and the centroid to which it belongs by minimizing the following objective function

$$J = \sum_{i=1}^{K} \sum_{x_j \in c_i} \|x_j - c_i\|^2 \tag{2.1}$$

Minimizing the above mentioned objective function is an iterative process, where in each iteration of the k-means algorithm we refine the choices of centroids to reduce distortion. The change in distortion is used as the stopping criteria i.e. when the change in distortion between the current iteration and the previous iteration is less than a particular threshold, we will stop. We can also restrict the maximum number of iterations allowed in this process, in which case algorithm will terminate after the maximum number of iterations irrespective of the change.

Traditionally in bag of words model while creating the image representation, we cluster the visual features using K-means to form K-clusters, where each cluster's representative is the centroid or the mean of that cluster. Now this cluster representative will serve the purpose of visual word. Though these visual words are still not defined as precisely as the textual words but these are the closest we can

get to them. Now this set of K-cluster centroid represents the visual vocabulary or the codebook or the dictionary of the dataset.

2.1.3 Image Representation

Once we have constructed the vocabulary, next step is to get the image representations using this vocabulary. Since this vocabulary represents the collection of visual words present in the dataset, we basically represent the image as the frequency of the these visual words without considering their spatial context as it is done in text classification. Using the visual features and the visual vocabulary, we build the histograms/ image representations. This can be divided into following steps

- **Assignment/Coding Step:** In this step each visual feature is assigned to the nearest visual word from the vocabulary. Traditionally, these assignments are done in the hard manner i.e. visual feature is assigned to the nearest visual word only, which implies that it will be completely represented by that particular word and its relevance to other visual words is dropped. So, visual features assigned to the same visual word are considered exactly similar and the visual features assigned to separate visual words are considered dissimilar. In general, for each visual feature we create a $K - Dimensional$ code vector or assignment vector which denotes the visual word to which a particular feature is assigned to by keeping value of a index as 1 (which is the index of the visual word to which this feature is assigned) and rest all 0's.

- **Pooling Step:** Once we have got the code/assignment vectors for all the visual features present in a particular image, we have to club them together to get the final image representation. Advantage of pooling is that number of visual features can vary for different images, but pooling all the code vectors ensures that final image representation for all the image is of the same dimension k.

2.2 Practical Issues in Building Effective Representations

Although Bag of visual Words model, after its introduction has gained a lot of fame in the field of computer vision due to its fast, efficient and best performance, it still has various practical issues associated with it. Some of them are discussed below.

2.2.1 Feature extraction

This step of the pipeline is the most stable step and researchers unanimously but silently had agreed that any local features can be used depending upon the problem setting. The only practical issue with this step is the interest point detection. As mentioned in Section 2.1.1, we can chose interest points using various techniques. Although dense sampling is the most preferable one but it is not good for all problem settings. So, we might have to decide the method of choosing interest points empirically in some cases.

2.2.2 Vocabulary construction

This step is very crucial in deciding the performance of the complete BOW pipeline. There are various types of practical issues which are associated with this step. Following are the issues along with their explanation:

1. For building vocabulary, we have to apply a clustering algorithm on the visual features to group them together. Although the number of visual features are very less as compared to number of pixels in an image but, still their number is quite large for performing the clustering on all of them. Consider for example a dataset with 10K images in it with each image having 2K visual features (SIFT) each of 128 dimension. So total number of feature on which we have to perform clustering will be 2 million each of dimension 128. So performing a simple K-means clustering on such a huge data is computationally very expensive. So, to tackle with this problem we perform clustering by sampling a subset of the visual features and then cluster this subset to get the codebook. This subset can be chosen in various ways like, (a) randomly sampling features from the complete set, (b) fixed number of features from each image, (c) fixed number of features from each class, and so on.

2. *Vocabulary Type:* We have to cluster the visual features to group them together on the basis of some criteria which is decided by the algorithm which is chosen to do the clustering. Mostly we use K-means clustering, but actually choosing the best algorithm for this grouping is a separate field of research in itself. In every type of clustering there is a loss of information since clustering is also a sort of dimensionality reduction process. The amount and type of loss of information depends on the clustering algorithm we chose to build clusters. Hence this step affects a lot in the final performance of the pipeline.

3. *Supervised or Unsupervised:* Another issue related to the vocabulary construction is the use of external information (present as the labels for each training image) for performing the grouping. If we use all the labels and build a vocabulary for each class separately, that somehow reduces generality of our codebook. On the other hand if we build completely general vocabulary, that is not useful for various tasks where we need high level of discrimination. So, this issue is mostly problem dependent but a lot of research is going in this field to see if we can come up with a general solution which is independent of the problem.

4. *Vocabulary Size:* Vocabulary size has the direct impact on the performance and computation complexity of the model as the final image representation is a vector of length equal to the size of vocabulary. There is a trade-off between the computation complexity and representation power of the final image representation. If we have a small vocabulary, final representation of image will be small which implies less computations to be done but in this case visual vocabulary will be too general and the visual words will not be able to capture the minute details present in the image. As we increase the vocabulary size, we are increasing the representation power of the

final image histogram/representation along with the computation complexity. But this increase in performance can be achieved only upto a certain point (sometimes called as saturation point), beyond which the performance starts decreasing. This decrease is caused by the fact that when we have very large vocabulary size, visual words are too specific for the training set, which leads to over-fitting. Since there is no exact solution for deciding the apt vocabulary size for a given problem, so it is generally decided by carrying out extensive experimentation.

2.2.3 Image Representation

This step as explained in section 2.1.3, involves the assignment/coding step in which for each visual feature an assignment code is generated, which are pooled to get the final image representation. In the assignment step, since each visual feature is assigned to the nearest visual word, its relevance to the other visual words is lost. This leads to two problems of codeword uncertainty and codeword plausibility. Codeword uncertainty refers to the situation when we have two or more equally competing visual words for a particular feature, but as we have to assign a feature to single visual word, so it is assigned randomly to any one of these and its relevance to the other visual words is lost. So, by doing this we are losing an important information. In codeword plausibility, we assign a non deserving visual feature (e.g. an outlier) to the nearest visual word. Both of these problems are addressed and solved in chapter 3.

2.2.4 Spatial Context

Since while constructing the final image representation, we only consider the frequency of the each visual word present in that image, without considering their spatial arrangement. So this leads to the loss in spatial context. As a result, if there are two different images with a particular object lying in the different part of the images and our usecase is to distinguish such images and put them into separate sets, BOW model will fail to do so since for this model only frequency of visual words matter which will be same for both the images.

2.2.5 Non-Linear Classifiers

Now once we have got the image representations, we need to train the classifier which will serve as the model for distinguishing the future unseen images into various classes/sets. The image representations we get from bag of visual words model are not linearly separable. As a result to classify them properly, we need to train non-linear classifiers, which are hard to train due to the complexity associated with them. This complexity increases with the increase of the training data. As a result, we can't work with large training data which might lead to insufficiently trained classifier.

2.3 Recent Advances in Bag of Words

With the success of Bag of visual words in the field of computer vision, researchers now started exploring various advances in this area to further improve the BOW approach by addressing various issues mentioned in the previous section.

2.3.1 Soft Assignments

As explained in section 2.2.3, hard assignments in traditional bag of words model leads to problem of codeword uncertainty and codeword plausibility [43]. The main reason due to which these issues arise is that information related to other relevant words is discarded when we assign a visual feature to its nearest visual word. To overcome this issue, instead of assigning the visual feature to the nearest visual word, we assign it to multiple visual words in a soft manner. These soft assignments can be done in various ways. In depth analysis of these various techniques have been discussed in Chapter 3.

2.3.2 Sparse Coding (SC)

As addressed in section 2.2.5, working with non-linear classifiers implicitly puts a cap on the training data which can be used to train the model. So, to improve the scalability, researchers aim at obtaining nonlinear feature representations that work better with linear classiers. In particular, Yang et al. [56] proposed the "Sparse Coding Spatial Pyramid Matching (ScSPM)" method where sparse coding (SC) was used instead of vector quantization (VQ) to obtain nonlinear codes.

Sparse coding provides a class of algorithms for finding succinct representations of input data. Given only unlabeled input data, it learns basis functions that capture higher-level features in the data. Unlike some other unsupervised learning techniques such as PCA, sparse coding can be applied to learning overcomplete basis sets, in which the number of bases is greater than the input dimension. Sparse coding can also model inhibition between the bases by sparsifying their activations. Despite the rich promise of sparse coding models, their development has been hampered by their expensive computational cost. In particular, learning large, highly overcomplete representations is extremely expensive.

The goal of sparse coding is to represent the input vectors approximately as a weighted linear combination of small number of "basis vectors". These basis vectors thus capture high level patters in the input data. Concretely each input vector $\vec{x} \in R^d$ is succinctly represented using basis vectors $\vec{v}_1, ..., \vec{v}_K \in R^d$ and a sparse vector of weights or coefficients $\vec{u} \in R^K$ such that $\vec{x} \approx \sum_j \vec{v}_j u_j$. The basis set is overcomplete (K >d) and can thus capture a large number of patters in the input data.

Sparse coding aims at minimizing the objective function given by equation Eq. 2.2 where we try to minimize the reconstruction error with L_1 regularization on the weights/coefficients.

$$J_{SC} = min_{u,V} \sum_{j=1}^{N} ||x_j - u_j V||^2 + \lambda |u_j| \tag{2.2}$$

subject to

$$||v_k|| \leq 1, \forall k = 1, 2, ..., K \tag{2.3}$$

The optimization problem is convex in V (while holding U fixed) and convex in U (while holding V fixed) but not convex in both simultaneously. To solve this we iteratively optimize the above objective by alternatingly optimizing with respect to V (bases) and U (coefficients) while holding the other fixed.

For learning the bases V, the optimization problem is a least square problem with quadratic constraints. For learning the coefficients U, the optimization problem is equivalent to a regularized least square problem.

2.3.3 Locally constrained Linear Coding (LLC)

It was found empirically that SC results tend to be local i.e. nonzero coefficients are often assigned to bases nearby to the encoded data which suggested a modification to SC, called Local Coordinate Coding (LCC) [51], which explicitly encourages the coding to be local, and theoretically pointed out that under certain assumptions locality is more essential than sparsity, for successful nonlinear function learning using the obtained codes. Similar to SC, LCC requires to solve L1- norm optimization problem Eq. 2.4, which is however computationally expensive.

$$J_{LLC} = min_c \sum_{i=1}^{N} ||x_i - uiV||^2 + \lambda ||d_i . * u_i||^2 \tag{2.4}$$

where $.*$ denotes element wise multiplication and d_i is given by

$$d_i = e^{\frac{dist(x_i, V)}{\sigma}} \tag{2.5}$$

The significance of d_i is that it controls the decay speed of the relevance of a codeword for a feature. Farther codewords will be given smaller weights to minimize the second term of the objective function.

Properties of LLC

1. **Better Reconstruction:** Since in VQ, feature is assigned to a single codeword the reconstruction error is quite high. Hence non-linear kernels are required for better performance to make up the loss in quantization. In LLC, each feature is formed by linear combination of local features which induces sparsity and locality and hence gives better reconstruction of the input data.

2. **Smooth sparsity:** In SC tough the reconstruction error is less but a feature can be a linear combination of any few basis vectors. As a result due to over-completeness similar features can have different representations/ codes. But in LLC there is a constraint that a feature can only be a linear combination of local features which gives similar representation for similar types of features.

2.3.4 Fisher Vector Representations (FV)

Further extending the problem of computing non-linear image representations, so that we can exploit the benefits of working with linear classifiers, researchers thought of a new addition to the image representations. Traditional bag of words representations, count the number of occurrences of a particular visual word in an image. Thus, they capture the $0 - order$ statistics of the distribution of the visual descriptors. Fisher vector representations extend the traditional bag of words by encoding higher-order statistics into the final image representations.

In Fisher vector framework [32], we try to fit each of the visual descriptors x_i to a generative probability distribution model (GMM) $P(x_i|\theta)$, θ being the set of model parameters. This is done by moving in the direction of the gradients wrt to the parameters of the model, so that it can best represent the given input data. Based on these gradients, a Fisher score is assigned to each visual descriptor which is used to compute the final image representation. Complete details of the method are explained in chapter 4.

Fisher vector image representations are huge in size. For a vocabulary of size K, and visual features of dimension D, if we want to capture both first and second order statistics, our final image representation will be of size $K(2D + 1)$.

2.3.5 Vector of Locally Aggregated Descriptors (VLAD)

VLAD[16], is the non-probabilistic Fisher vector image representation. We replace the generative probabilistic model (GMM) in Fisher vector representations with K-means. Then each visual descriptor is associated to its nearest visual word in the vocabulary. For each of the visual word c_j, we compute the aggregation of the differences $x_i - c_j$ of the visual descriptors x_i assigned to c_j using the following equation:

$$v_j = \sum_{x_i \in c_j} x_i - c_j \tag{2.6}$$

Each of this v_j (which is the aggregation vector for visual word c_j), is of dimension D, where D is the dimension of each visual descriptor. So now VLAD is the concatenation of the D-dimensional vectors to get the final representation of size KD.

To summarize, VLAD is a simplified non-probabilistic version of FV or the VLAD is to the FV what K-means is to GMM clustering. Perronnin *et. al.* [17], have derived this relation between FV and VLAD using mathematical proofs as well.

2.4 Insights into the datasets and evaluation methods

2.4.1 Evaluation Method

For evaluating our methods, we divided the datasets into two sets, training and testing. Using the training set as name suggests, we followed the complete training pipeline, i.e. feature extraction, vocab-

ulary construction, building the image representations and finally using these image representation, we trained a classifier. After this we tested our model by classifying the images present in the testing set.

There are many different types of classifiers which can be used for different problem settings, e.g. K-Nearest Neighbor classifier, Decision Trees, Support Vector Machines, etc. For our evaluation purposes, we have used SVM's [7] extensively. Since, SVM can only classify two classes, so to make SVM work in a multi-class setting (say N classes), we can either train a $1vs1$ or $1vsRest$ classifier. And later we will combine the results we get from these individual classifiers to get the final result. In $1v1$, we will train a separate classifier for each pair of classes, so we will have a total of $C(N, 2)$ classifiers. On the other hand in $1vsRest$ setting, we will train a separate classifier between a particular class and rest of the classes. So in this case we will have a total of N classifiers. In all our evaluations we have used $1vsRest$.

For evaluation also we can different techniques to report performance. e.g. mAP, classification accuracy, etc. We have used these two techniques extensively throughout the course of this thesis, so we thought its worth explaining about them in brief.

1. **Mean Average Precision:** It represent the mean of average precision scores for a set of queries. It is the area under the precision-recall curve (or in short PR-Curve). Its value signifies how precisely are we able to retrieve information from a given dataset. Higher the value better is the performance.

2. **Classification Accuracy:** Accuracy literally means how accurately we are able to predict and unknown data using our model. We compare the label which we get for each of the test image and compare it with its original label. So, this is the measure of how many predicted and actual labels match.

2.4.2 Datasets

The main attribute for a dataset to be a good dataset for computer vision tasks is that it should have varied images for a particular class, so that it covers most of the usecases of the problem in hand. Vision community have various benchmark datasets out of which we have used following four datasets for our evaluation tasks.

2.4.2.1 Scene-15

Figure 2.2, shows the sample images from Scene-15 dataset [39].

- **Specifications of Images:** It has 4485 images divided into 15 different scene categories. Each image is in gray-scale. Average image size is 300 x 250 pixels. The major sources of the pictures include the COREL collection, personal pictures and Google search images. This is one of the complete scene category dataset used in literature.

Figure 2.2 Sample image for Scene 15 Dataset. All the images are spanned over 15 different scene categories. This dataset is widely used for scene classification tasks.

- **Example Categories:** Bedroom, Kitchen, MITForest, MITForest, MITBuilding, CALSuburb, LivingRoom, MITHighway, MITOpencountry, PAROffice, Industrial, MITCoast, MITInsideCity, MITStreet, Store.
- **Train/Test Images:** From the total of 4485 images, 100 image per category were selected at random for training and rest 2985 images were kept as the part of testing set.
- **Evaluation Method:** We reported Mean Average Precision, averaged over all the 15 categories as our performance measure for this dataset.
- **Download Link:** `http://www-cvr.ai.uiuc.edu/ponce_grp/data/`.

2.4.2.2 Scene-67

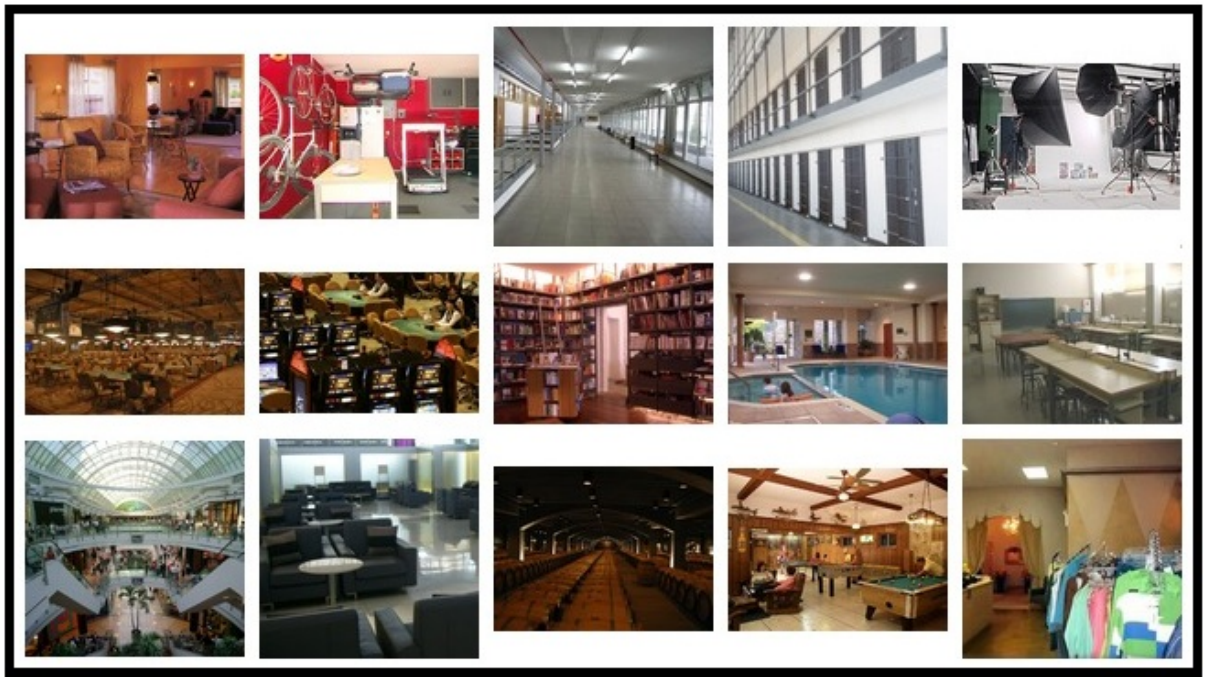


Figure 2.3 Sample Images from Scene 67 Dataset. Images in it belong to one of 67 different categories. This dataset mainly is used for indoor scene recognition.

Figure 2.3, shows the sample images from Scene-67 dataset [36]. The database contains 67 indoor categories.

- **Specifications of Images:** These 67 categories span over 15620 images. The number of images for each category varies, but there are atleast 100 images in each of the category. Each image is a colored image.
- **Example Categories:** Bakery, GroceryStore, BookStore, Bedroom, Nursery, Closet, PrisonCell, Library, Church, Buffet, FastFood, Bar, MeetingRoom, Warehouse, StudioMusic, etc.
- **Train/Test Images:** Since there are atleast 100 images per category, we chose 100 images per category from the complete dataset and out of these 100, 80 images per category (a total of 5360 images) were kept for training and 20 images per category (a total of 1340 images) were used for testing.
- **Evaluation Method:** We reported classification accuracy as the performance measure for this dataset.
- **Download link:** `http://web.mit.edu/torralba/www/indoor.html`

2.4.2.3 Pascal VOC 2007

Figure 2.4 Sample images from Pascal VOC 2007 Dataset. This is the most popular dataset in vision community. It has 20 categories for which every year there is a competition, where the participants have to increase performance of various computer vision models using it as the benchmark dataset. This particular dataset is the version which was used for this competition in the year 2007.

Figure 2.4, shows the sample images from Pascal VOC 2007 dataset [10].

- **Specifications of Images:** It contains 9963 images which are divided into 20 categories. Each image present have atleast 1 occurrence of a particular class. Many images even have multiple classes present in it. E.g. an image of class bicycle can have persons in it as well. So this is one of the toughest dataset for various computer vision tasks. All the images are colored and average image size is 350 x 500 pixels.
- **Example Categories:** Aeroplane, Bicycle, Bird, Bus, Boat, Bottle, Car, Cat, Chair, Cow, DiningTable, Dog, Horse, MotorBike, Person, PottedPlant, Sheep, Sofa, Train, TvMonitor.
- **Train/Test Images:** Total image are divided into training and testing set with training set consisting of 5011 images and testing set having 4953 images.
- **Evaluation Method:** For evaluation just like scene 15 we reported mAP averaged for all the 20 categories as performance measure for this dataset.
- **Download Link:** `http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/index.html`

2.4.2.4 Caltech-101

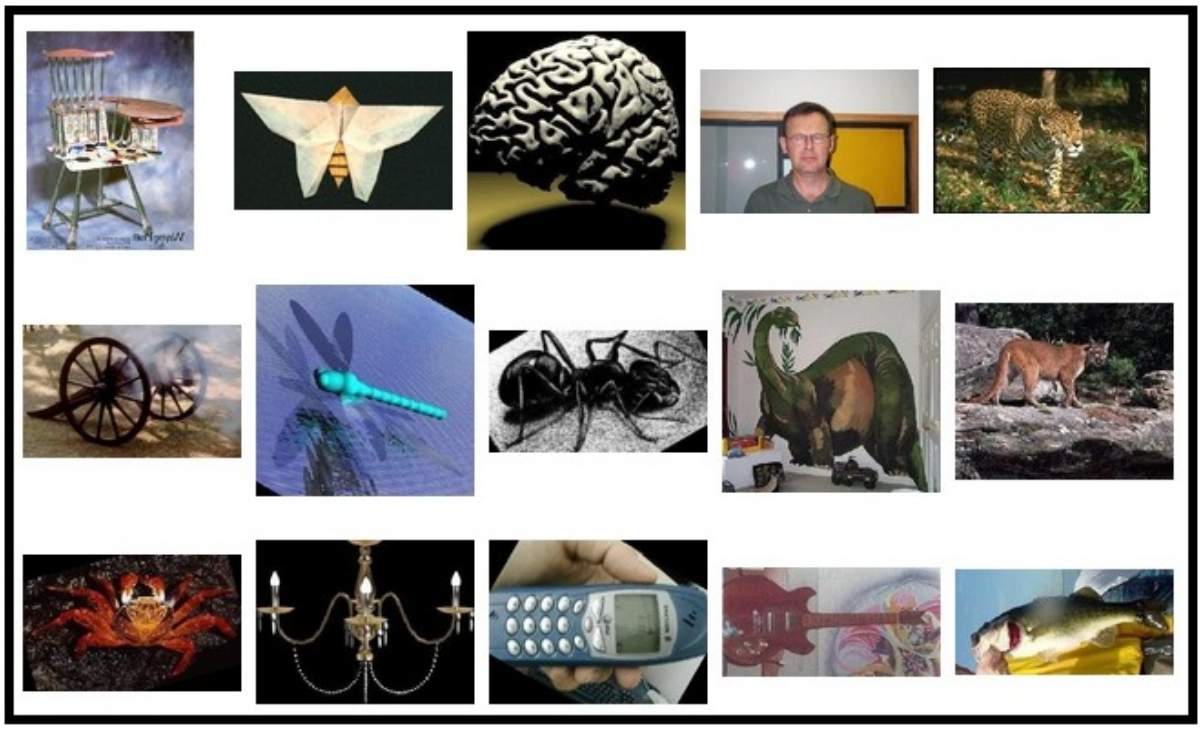


Figure 2.5 Sample Images from Caltech 101 Dataset. It has 101 different categories. This is also one of the benchmark datasets in vision community. This is relatively easier dataset to work with as compared to pascal.

Figure 2.5, shows the sample images from Caltech 101 dataset [11].

- **Specifications of Images:** Images in the dataset belongs to 101 different categories. Each category has about 40 to 800 images with most of them having 50 images with a total of 9146 images. All the images are colored and average image size is 300 x 200 pixels.
- **Example Categories:** Airplane, Anchor, Barrel, Binocular, Brain, Camera, Butterfly, Crab, Cup, Dolphin, Elephant, ElectricGuitar, Lamp, Lobster, Lotus, Mayfly, Saxophone, Strawberry, etc.
- **Train/Test Images:** Popular number of training images vary from 1, 3, 5, 10, 15, 20, 30. Popular testing images per category varies between 30, 50. Following the standards in [39], we have restricted the number of testing images to 50 and picked 30 images per category for training. Thus reducing our total images set to 5975 images.
- **Evaluation Method:** We reported classification accuracy averaged over all the classes as performance measure for this dataset.
- **Download Link:** `http://www.vision.caltech.edu/Image_Datasets/Caltech101/`

Chapter 3

Bag of visual words: A soft clustering based exposition

3.1 Introduction

In the recent past, advances in the storage, capturing device and Internet has led to the rapid growth in the number of digital image collections. Automatic classification of images based on the semantic category can be helpful in efficient search and management of these large collections of images. For example, a collection of photos needs to be categorized into semantic categories like "bedroom", "mountain", "night time", etc., to support efficient browsing and search. Recent research shows the success of Bag of Words representation for images in automatic classification and search tasks [39, 4, 12, 41].

Bag of visual word representation is inspired from the word-document representations of images. The first step of the bag of visual words based approach is the computation of local feature descriptors like SIFT [26] for a set of image patches. These patches can be either at the key-point locations or densely sampled on a regular grid of the image. These set of local feature descriptors are quantized using a clustering technique. This step is referred to as vocabulary building step. The resultant set of cluster centers is referred to as visual vocabulary or codebook, and each cluster center is individually called a "visual word" or "codeword". The generated visual vocabulary is then used for assigning the nearest visual word for each of the local feature descriptors in a given image. This step is referred to as assignment step. The histogram of visual words in a given image is used as its representation for image classification, retrieval or recognition tasks. The main drawback of the traditional bag of words approach is the hard assignment of the visual code words to the local image features [35]. As only a single visual word is assigned to a given feature descriptor, the relevance of the feature descriptor to other visual words is lost which leads to poor results in classification tasks. In order to overcome this problem, we use fuzzy set theoretic notions in the traditional bag of words approach. Fuzzy logic allows an object to belong to multiple classes with varying degrees of membership. This helps in modeling the ambiguity of assigning a visual codeword to a local feature descriptor. Introducing fuzziness, leads to better characterization of an image in terms of the distribution of visual words, which in turn helps in the better classification of the images.

In the section 2, we discuss the related work. We describe the traditional Bag of words approach, its shortcomings and fuzzy bag of words approach along with experimental results in section 3. Section 4 contains an alternative approach to Possibilistic BoW by eliminating fuzziness parameter, from which we will derive the equation for soft assignment which is used frequently [35] and finally conclude in section 5.

Bag of Words representation, motivated from field of documents search, was first used for the problem of texture recognition [23]. It was then popularly used for content based image search and classification tasks [39, 4, 12, 41]. Spatial layout is lost by representing an image as histograms of visual words for image classification tasks. In order to capture spatial layout, Lazebnik *et. al* [39] proposed pyramid histogram of visual words.

Other improvements in bag of words approaches mainly focus on the vocabulary generation. Vogel *et. al* [49] present a semantic vocabulary for scene classification tasks where in each image patch is labeled with a semantic label like sky, water, grass, etc., Winn *et. al* [52] proposed universal codebook vocabulary for object recognition. Perronnin *et. al* [31] presented a class-specific vocabularies for generic visual recognition. Jurie and Triggs [18] compare different clustering techniques for generation of vocabulary. Lazebnik *et. al* [22] have proposed learning of quantization code-books by information loss minimization.

Philbin *et. al* [35] have presented the use of assignment of a single descriptor to multiple nearest visual words for the problem of particular object retrieval. Our work is in the similar direction of these works, which aim for obtaining better representation of the bag of words by taking care of the multiple relevant visual words.

3.2 Fuzzy Bag of Visual-Words

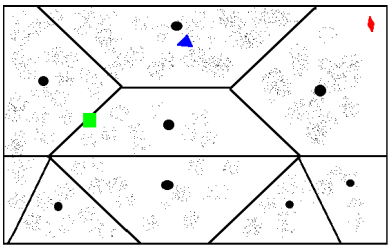


Figure 3.1 An example showing the problems of codeword ambiguity. The small dots are image features, the circles are codewords found by hard clustering. Data sample that is well suited to the codebook approach is shown by blue triangle. Problem of codeword uncertainty is shown by the green square, and the problem of codeword plausibility by the red diamond.

Traditional bag of words approach assigns a single visual word to each of the features descriptors in an image. This hard assignment gives rise to two issues: codeword uncertainty and codeword plausibility [43]. Codeword uncertainty refers to the problem of selecting the correct codeword out of two

or more relevant candidates. The traditional bag of words approach merely selects the best representing codeword, ignoring the relevance of other candidates. Codeword plausibility denotes the problem of selecting a codeword without a suitable candidate in the vocabulary. Traditional bag of words approach assigns the best fitting codeword, regardless of the fact that this codeword is not a proper representative. Both these problems are illustrated in Fig. 3.1.

We now show how these problems can be handled by introducing fuzziness in the vocabulary building and assignment steps. In general, hard clustering schemes like k-means clustering is used for vocabulary building. Given a set of N visual feature descriptors, k-means algorithm tries to find an optimal set S, having C cluster centers, which minimizes the following objective function:

$$J_{kmeans}(S) = \sum_{i=1}^{C} \sum_{j=1}^{N} ||x_j^{(i)} - c_i||^2 \tag{3.1}$$

where, x_j represents j^{th} feature, c_i represents the center of i^{th} cluster. The above equation assigns a feature descriptor to the single nearest cluster center without considering the other most nearest cluster centers.

In fuzzy vector quantization framework, instead of assigning each feature with a single codeword, we use an uncertainty term model [2] in which each feature is assigned to multiple codewords with some membership value, which represents its relevance to that codeword. This membership value can either be relative (as in *"Fuzzy C-Means (FCM)"*) or absolute (as in*"Possibilistic C-Means (PCM)"*) [2, 20].

3.2.1 Fuzzy/Probabilistic C-Means

In fuzzy c-means we use a membership function u_{ij} to modify the objective function as follows.

$$J_{fcm}(S) = \sum_{i=1}^{C} \sum_{j=1}^{N} u_{ij}^m ||x_j - c_i||^2 \tag{3.2}$$

subject to the condition

$$\sum_{i=1}^{C} u_{ij} = 1, \forall j \tag{3.3}$$

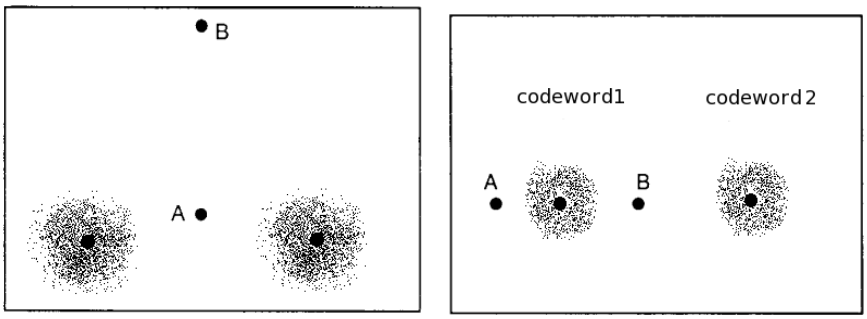
Here u_{ij} is the membership value of the j^{th} feature to the i^{th} codeword. The above equation is minimized by using iterative optimization using the following update equations

$$u_{ij} = \frac{d_{ij}^{\frac{-2}{m-1}}}{\sum_{l=1}^{c} d_{lj}^{\frac{-2}{m-1}}}, \tag{3.4}$$

$$c_i = \frac{\sum_{j=1}^{N} u_{ij}^m x_j}{\sum_{j=1}^{N} u_{ij}^m} \tag{3.5}$$

Here u_{ij} is the membership value and d_{ij} is distance of the j^{th} feature to the i^{th} codeword respectively and m, $(m > 1)$ is called the fuzzifier or the weighting exponent whose value determines the amount of fuzziness that is introduced in the assignments.
Assigning the features in this manner encodes the relevance of a feature to a particular codeword depending upon its distance from the other codewords. Eq.(3.3) ensures that the sum of the membership degrees for each feature to all the codewords equals 1. This means that each feature receives the same weight in comparison to all other data and, therefore, that all data are (equally) included into the cluster partition. The membership values resemble the probability of a particular feature belonging to a particular codeword, since sum of the membership values of a particular feature for all clusters is 1.

Although this probabilistic fuzzy assignment solves the problem of codeword uncertainty and plausibility but probabilistic fuzzy membership values can be misleading when there is some noise or outliers. Consider, for example,the simple case of two codewords shown in Fig. 3.2(a) . Feature A has the same distance to both the codewords and thus it is assigned a membership degree of about 0.5. However, the same degree of membership are assigned to feature B even though this feature is further away from both the codewords and should be considered less typical. Because of the normalization (Eq.2) however, the sum of the membership values has to be 1. Consequently B receives fairly high membership values of 0.5 to both the codewords. Also two features equidistant from a particular codeword, may be assigned



(a) (b)

Figure 3.2 (a) Example of a dataset with two features A and B in which the (Probabilistic) fuzzy membership of these features in both the codewords are equal, even though feature B is much less representative of either codeword. (b) Example of a dataset with two codewords in which the membership generated by the Probabilistic assignments for features A and B are different, even though they are equidistant from codeword 1.

with different membership values because the membership value not only depends on the distance of a feature from that particular codeword but also on its distance from other codewords. This is shown in Fig. 3.2(b), where features A and B, even though are equidistant from codeword 1 but will have different membership values. This problem arises from the constraint on the memberships, which forces feature B to give up some membership in codeword 1 in order to increase its membership in codeword 2.

3.2.2 Possibilistic C-Means

The problems caused by noise in the probabilistic assignments are mainly because of the normalization constraint (Eq.(3.3)). By dropping this constraint, we can achieve a more intuitive assignment of degrees of membership and avoid undesirable normalization effects. But dropping the normalization constraint can lead to the trivial solution where all u_{ij} are 0, which will lead to the minimization of the J_{fcm} [20]. This problem is overcome by adding another term in the objective function as follows

$$J_{pcm}(S) = \sum_{i=1}^{C}\sum_{j=1}^{N} u_{ij}^{m}||x_j - c_i||^2 + \sum_{i=1}^{C}\eta_i \sum_{j=1}^{N}(1-u_{ij})^m, \tag{3.6}$$

where $\eta_i > 0 (i = 1, \ldots, c)$. The first term leads to a minimization of the weighted distances, the second term suppresses the trivial solution since this sum rewards high membership (close to 1) that makes the expression $(1 - u_{ij})^m$ approximately 0. Thus the desire for the strong assignments of features to the clusters is expressed in the objective function J_{pcm}.
J_{pcm} is minimized by using the following update equation for membership values

$$u_{ij} = \frac{1}{1 + (\frac{d_{ij}^2}{\eta_i})^{\frac{1}{m-1}}} \tag{3.7}$$

The update equation for c_i remains the same as Eq. (3.5).

Here η_i is called the "bandwidth" or "resolution" or "scale" parameter. Considering the case m = 2 and substituting η_i for d_{ij}^2 yields u_{ij} = 0.5. It becomes obvious that η_i is a parameter that determines the distance to the cluster i at which the membership degree should be 0.5. The significance of this parameter can be seen like this, that it is distance beyond which a feature will not be of much relevance to a particular codeword. Its value can either be fixed for every codeword or can be estimated by the fuzzy intra-cluster distance using the fuzzy membership matrix.

A distinguishing characteristic of possibilistic assignment is that the membership values u_{ij} of feature j in codeword i is absolute as depends only on the distance of the feature from the codeword as compared to probabilistic assignments where they are relative.

3.2.3 Experimental Results

We present the scene classification results using the hard assignment as baseline results and compare them with classification results obtained using fuzzy framework (both fuzzy probabilistic and fuzzy possibilistic assignments). All the experiments are performed on Scene-15 dataset [39] and Pascal VOC 2007 [10].

Scene-15 dataset consists of 4485 images spread over 15 categories like mountains, forests, kitchen, etc. Pascal VOC 2007, dataset consists of 9963 images with 5011 training and 4952 testing images spanning over 20 categories, like cat, dog, car, bus, bicycle, etc. Mean Average Precision (mAP), calculated using different techniques is used as the evaluation measure for our experiments.

Initially, we extract dense SIFT feature descriptors from each of the training images and cluster a subset of them to obtain a visual vocabulary. Then the histogram of the visual words is build for all the images with hard, fuzzy probabilistic and fuzzy possibilistic assignments, which is used to represent an image. We use 33 percent of images for testing and 67 percent of images for training from the available images of each category. Then a 1-vs-all SVM classifier is trained for all the 15 classes. The overall mAP is used as the evaluation measure in our experiment, which is calculated as the mean of the mAP of all the classes. Since clusters were initialized randomly, so to ensure fair experimental evaluation each experiment was performed 10 times with different initialization.

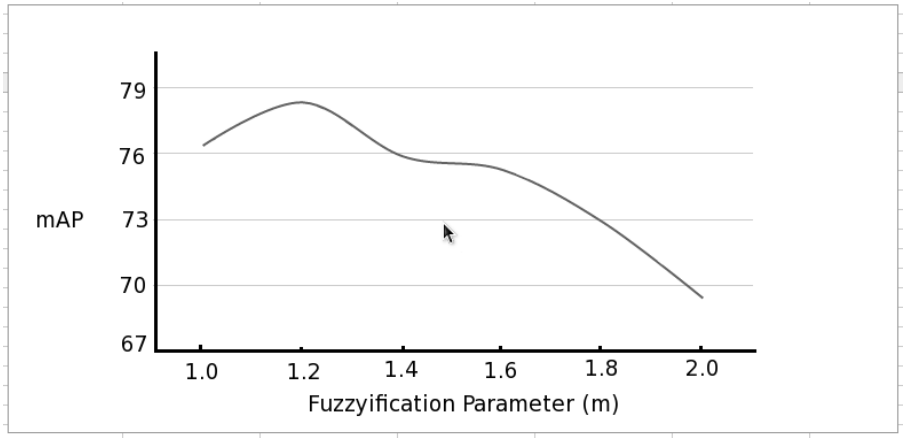
Initially we use, fuzzy approach both in the vocabulary construction and the assignment step but those results were not as good as compared to baseline results. The reason could be, that vocabulary built using the fuzzy probabilistic and possibilistic approaches were not that discriminative since the cluster centers were coming very close to each other. Therefore, we used vocabulary built using the hard k-means, but introduced fuzziness in the membership assignment step.

First three columns of Table I, shows the average of mAP calculated in 10 different experiments along with the variance, for different vocabulary sizes using histograms prepared by hard, fuzzy probabilistic and fuzzy possibilistic assignments. We can observe that both probabilistic assignment and possibilistic assignments give better results as compared to the baseline results of hard assignments. For a given vocabulary size, we choose the fuzzification parameter m (for probabilistic assignments) and scale parameter η (for possibilistic assignments) resulting in the best mAP by experimentation. $m = 1.2$ and $\eta = 0.05$ was giving the best results for our dataset.

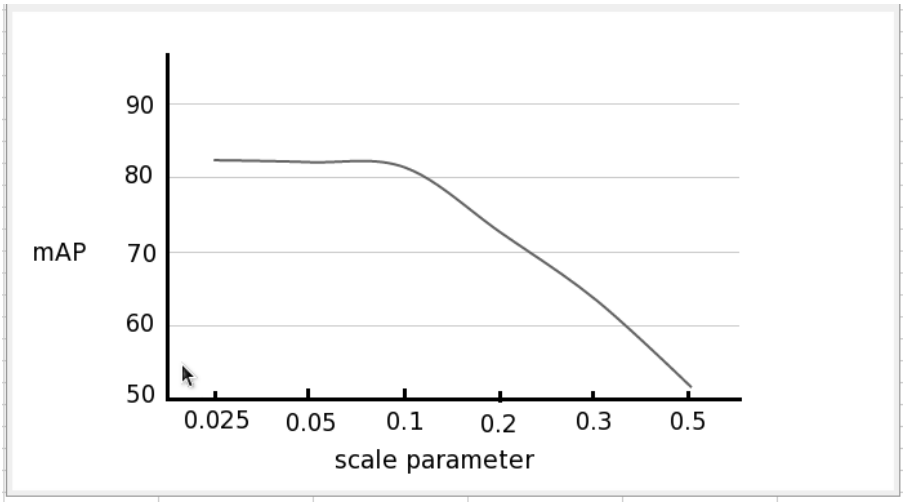
It can be seen from graph Fig. 3.3(a) that in case of probabilistic assignments the classification performance decreases with the fuzzification parameter m (here vocabulary size = 1000). As the fuzzification parameter m increases all the clusters gain equal membership value of $\frac{1}{c}$ for any feature descriptor. This results in less discriminative representations of the bag of words for images of all the classes, leading to decrease in the performance.

Similar behavior is seen in case of possibilistic assignments Fig. 3.3(b). The classification performance decreases as we increase the scale parameter η because as value of scale parameter approaches infinity, the membership value of each feature corresponding to every cluster approaches 1, which again results in less discriminative representations and aptly performance decreases.

Though fuzzy possibilistic assignments are performing better than the hard assignments, but are not as good as fuzzy probabilistic assignments. The possible reason could be that, the interpretation of m is different in the case of FCM and the PCM. In the FCM increasing values of m represent increased sharing of points among all the clusters, whereas in the PCM, increasing values of m represent increased possibility of all the points belonging to a given cluster. So it would be better to remove this m completely from the J_{pcm} [20], which will lead to better performance, as shown in the next section.



(a)



(b)

Figure 3.3 (a) Effect of fuzzification parameter m on mAP.(b) Effect of scale parameter η on mAP.

3.3 Possibilistic C-Means Another Approach

The objective function for PCM described above is a particular implementation of the possibilistic approach which is dependent on fuzzifier parameter m. We could eliminate m altogether by choosing alternative formulations of the PCM described below.

$$J'_{pcm}(S) = \sum_{i=1}^{C}\sum_{j=1}^{N} u_{ij}||x_j - c_i||^2 + \sum_{i=1}^{C}\eta_i \sum_{j=1}^{N}(u_{ij}\log_{ij} - u_{ij}) \tag{3.8}$$

The updating equations for the membership degrees can be derived from J'_{pcm} by setting its derivative to zeros as follows

$$\frac{\delta(J'_{pcm})}{\delta u_{ij}} = 0 \Rightarrow d_{ij}^2 + \eta_i \log u_{ij} + 1 - 1 = 0$$

$$\Rightarrow \log u_{ij} = \frac{-d_{ij}^2}{\eta_i} \Rightarrow u_{ij} = e^{\frac{-d_{ij}^2}{\eta_i}} \tag{3.9}$$

Eq. (3.9) defines the membership degrees of the j^{th} feature to the i^{th} cluster center which is similar to membership values of the "Descriptor-space soft assignment" used by Philbin *et. al* [35]. Experiments done using Eq. (3.9) as the membership assignment equation, outperforms all the techniques used above (even fuzzy assignments and possibilistic assignments described above). Table 3.1 and table 3.2 shows the result where mAP of each techniques mentioned in section 3 is compared with the "soft assignment" (a particular type of possibilistic assignment) on Scene15 and PASCAL VOC 2007 datasets respectively.

Table 3.1 Comparison of classification results using Hard, Probabilistic and Possibilistic Assignments on Scene-15 Dataset.

Vocabulary Size	Hard Assignment	Fuzzy Assignment	Possibilistic Assignment I	Possibilistic Assignment II
200	71.21	75.07	73.29	78.38
500	75.32	76.85	75.92	80.27
1000	75.99	77.43	76.23	81.25
2000	76.02	78.98	76.89	82.46
4000	76.20	79.80	77.02	84.13

Table 3.2 Comparison of classification results using Hard, Probabilistic and Possibilistic Assignments on PASCAL VOC 2007.

Vocabulary Size	Hard Assignment	Fuzzy Assignment	Possibilistic Assignment I	Possibilistic Assignment II
200	29.36	31.99	31.30	35.67
500	32.12	33.17	32.89	40.32
1000	33.08	34.95	34.15	44.13
2000	36.11	37.78	36.55	46.88
4000	38.97	40.05	39.23	49.73

3.4 Summary

Due to the hard assignment of visual features to codewords there is a loss of relevance of other equally relevant codewords in traditional BoW. We have shown that fuzziness in the assignment step when used with the vocabulary built by hard k-means, can result in better "Bag of Words" representation for image classification tasks. The fuzzification parameter m and the scale parameter η are data dependent parameters, and their value needs to be selected experimentally for a given data. Experimental results on Scene-15 dataset demonstrate superiority of Fuzzy BoW over traditional BoW. Also, absolute fuzziness performs better as compared to relative fuzziness, in which absolute fuzziness leading to exponential membership values gives the best result.

Though soft assignments have increased our performance manifold in visual image classification, still researchers have shown better encoding methods are present as compared to soft assignments and can be used to further enhance the performance. So, in next chapter we will introduce Fisher vector

image representations which have shown promising results when used in classification pipeline. Also we will address some of the inherent issues with this new technique and try to solve them with our novel strategies.

Chapter 4

Sparse Discriminative Fisher Vectors

4.1 Introduction

Recognition of visual artefacts is one of the most popular problems in the Computer Vision community. Common tasks such as visual object recognition, image retrieval and scene classification, all involve assigning class labels to images, either implicitly or explicitly. Researchers generally agree on a three fold approach (Figure 4.1) to tackle such problems [6]: (a) computing low level feature descriptors over small image parts, (b) pooling these local descriptors to arrive at a global image representation of some sort, and (c) using a classification method that learns the underlying distribution of the seen examples from their global representations and uses this knowledge to predict the class labels of the unseen examples. Over the years, much research has gone into improving the individual stages of the pipeline.

Feature learning methods have evolved steadily over the past few decades and is perhaps the most stable component of the pipeline. Dense SIFT has emerged as a popular local descriptor for most classification tasks [18]. Meanwhile, majority of the efforts in this field have been focused on improving the feature pooling stage [5, 6]. The Bag of Features approach, where local features in each image are vector quantized (using K-Means on the training set features) and counted to arrive at a histogram image representation has enjoyed reasonable success for years. Some recent approaches have used soft assignments, as opposed to the traditional hard quantization in K-Means by expressing features as linear combinations of visual words [43]. Other approaches have aimed at minimizing loss of information by encoding the quantization errors in the representation [32, 58]. Finally, classification methods have seen their share of innovations too [40].

Fisher Kernels, which were introduced in 1998 [15], have achieved huge success after their introduction to the image classification domain [32]. Fisher encoding assumes the data distribution is a Gaussian Mixture Model (GMM) and captures the deviations in the fit. GMM can be understood as a soft visual vocabulary. Most state of the art approaches on various datasets employ Fisher encodings [6]. However, this performance comes at a huge computational and storage price. A fisher vector for K Gaussian components has a final representation size of $K(2D+1)$, where D is the dimension of the local features, and requires computations of the same order. Remedies that involve compressing Fisher vectors [33] and

methods such as Product Quantization [17, 38] have been suggested to cope up with the high memory requirements. While most of these methods compress Fisher vectors to reduce storage requirements and uncompress these on the fly during classification, [48] demonstrates that the Product Quantization code itself can be used for classification. Recent advances into this field have involved fine tuning the model parameters to incorporate class-discriminative information [9].

Some of the recent literature [5, 13, 53] has emphasized the success of sparse representations in classification. Sparse representations are said to be more discriminative, better representative and since the introduction of certain data structures, come at little computational and storage price.

In this chapter, we devise a simple approach to introduce sparseness in the Fisher vector representation. This was motivated by two objectives (a) to reap the performance advantages of the sparse representation in classification, and (b) to reduce the representation size and computational cost of the feature encoding procedure. In Section 4.3.2, we describe an implementation trick that helps us achieve objective (b). Our approach was inspired by [51], where locality constraints were enforced to compute a sparse image representation. We however use a different scheme to introduce sparsity in our representation. Another difference is that while [51] computes the best describing visual words for each low level feature of an image, we find the best representative visual words for an image or class. We also devise a novel strategy for class-discriminative encoding of Fisher vectors that identifies the Gaussian components most representative of each class and weights them appropriately to arrive at a discriminative representation. We demonstrate the superior performance of our approach over the traditional Fisher vector representations on several datasets. On all these datasets, we either outperform the state of the art or produce results that are comparable to the corresponding state of the art approaches. Figure 4.1 describes a block diagram of the Fisher vector computation procedure. The loopy arrows mark the modifications we have made to the pipeline. These modifications have been elaborated in section 4.3.

4.2 Fisher Kernel

Pattern classification models can be broadly divided into *"Generative Models"* and *"Discriminative Models"*. While generative models randomly generate input data using given hidden variables, discriminative models predict models for the unseen samples based on the distribution learnt from the training examples. Fisher Kernel [15] combines the advantages of generative statistical models (like GMM) and those of discriminative methods (like SVM). Recently Fisher kernels have been extensively used for various computer vision and machine learning tasks like retrieval and classification [32]. We now describe the Fisher kernel formulation.

Let $P(x_n|\theta)$ denote the generative probability distribution model, θ being the model parameters. $X = \{x_n, n = 1, \ldots, N\}$ represents the data item set and x_n is a data item. Let ∇_θ be the gradient function with respect to θ and $\log p(x_n|\theta)$ be the log-likelihood of x_n with respect to the model given the set of model parameters θ. Now for each data item we can define the Fisher score, F_{x_n} as the gradient

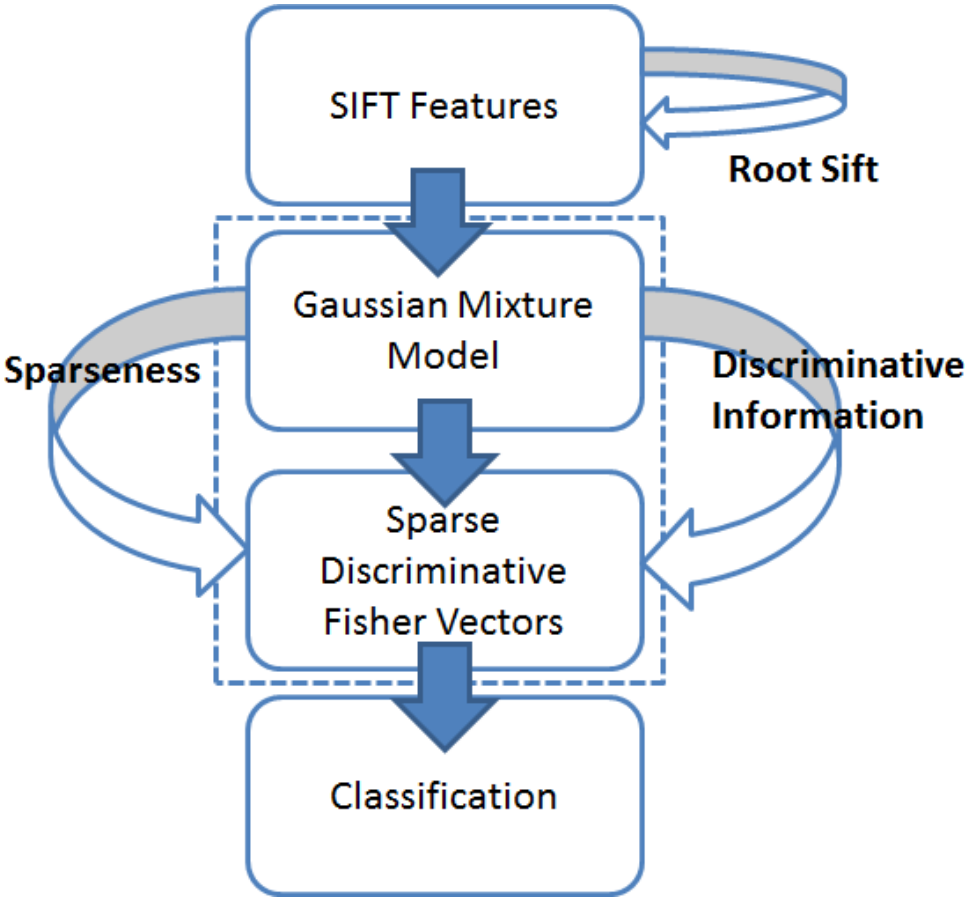


Figure 4.1 The Fisher vector computation pipeline. The loopy arrows mark our contributions.

of the log likelihood of x_n with respect to model paramters θ which is given by

$$F_{x_n} = \nabla_\theta \log p(x_n|\theta) \tag{4.1}$$

The Fisher score gives us the direction in which parameters should be modified to best fit the data. It transforms the variable length data into a N-dimensional feature space $\mathbb{R}^N$. Fisher kernel is defined by

$$K(x_i, x_j) = F_{x_i}^T I^{-1} F_{x_j} \tag{4.2}$$

where I is the Fisher information matrix, which is used for normalizing the gradient vectors. Because of its cost of computation and inversion, various approximations of I have been proposed in literature. We discuss some of these in section 4.2.1. Comparison of two data items is done by directly comparing their Fisher scores. If their Fisher scores are similar it implies that they would require similar adaptations to the model parameters and hence they are similar. Thus, we can directly classify these gradient vectors in place of the data items using any discriminative classifier such as SVM.

4.2.1 Fisher Vector Image Representation

Computing the gradient vector for each image involves computing a Gaussian Mixture model learnt over the low level features (eg. SIFT). $\theta = \{w_i, \mu_i, \Sigma_i, i = 1, \ldots, K\}$ denotes the model parameters of GMM with K components, where w_i, μ_i, and Σ_i represents the weight, mean and covariance matrix respectively, corresponding to the i^{th} Gaussian component. The mixture weights capture the relative frequency of each component of the Gaussian, thus, $\sum w_i = 1$. Various types of covariance matrices can be used in this formulation. We use the diagonal covariance matrix for simplicity.

Let $X = \{x_n, n = 1, \ldots, N\}$ denote the set of low level features for an image, each of dimension D, and $L(X|\theta) = \log p(X|\theta)$, where L is the log likelihood of the features with respect to the model. Since the features in the feature set X, are independent of each other, we can rewrite $L(X|\theta)$ as:

$$L(X|\theta) = \sum_{n=1}^{N} \log p(x_n|\theta) \tag{4.3}$$

The likelihood that a feature x_n is generated by GMM is given by:

$$p(x_n|\theta) = \sum_{i=1}^{K} w_i p(x_n|\theta_i) \tag{4.4}$$

where,

$$p(x_n|\theta_i) = \frac{\exp\{-\frac{1}{2}(x_n - \mu_i)'\Sigma_i^{-1}(x_n - \mu_i)\}}{(2\pi^{\frac{D}{2}})|\Sigma_i|^{\frac{1}{2}}} \tag{4.5}$$

Also let, q_{ni} be the probability that feature x_n is generated by i^{th} Gaussian component, which will be given by

$$q_{ni} = \frac{w_i p(x_n|\theta_i)}{\sum_{j=1}^{K} w_j p(x_n|\theta_j)} \tag{4.6}$$

For computing the Fisher vector representation for an image we calculate the gradient $\frac{\partial L(X|\theta)}{\partial \theta_i}$ for each parameter separately and then concatenate them to get the final gradient vector. Since the number of low level features can vary from image to image, to achieve invariance with respect to the number of features in the image, the final Fisher vector representation of an image is divided by the number of features. This is also referred to as average pooling in literature.

$$F_{Image} = \frac{1}{N}\frac{\partial L(X|\theta)}{\partial \theta_i} = \frac{1}{N}\sum_{n=1}^{N}\frac{\partial L(x_n|\theta)}{\partial \theta_i} \tag{4.7}$$

$$\frac{\partial L}{\partial w_i} = \sum_{n=1}^{N}[\frac{q_{ni}}{w_i} - \frac{q_{n1}}{w_1}], i \geq 2 \tag{4.8}$$

$$\frac{\partial L}{\partial \mu_i^d} = \sum_{n=1}^{N} q_{ni}[\frac{x_n^d - u_i^d}{\Sigma_i^d}] \tag{4.9}$$

$$\frac{\partial L}{\partial \Sigma_i^d} = \sum_{n=1}^{N} q_{ni}[\frac{(x_n^d - u_i^d)^2}{(\Sigma_i^d)^{3/2}} - \frac{1}{(\Sigma_i^d)^{1/2}}] \tag{4.10}$$

$$\frac{\partial L(x_n|\theta)}{\partial \theta_i} = [\frac{\partial L}{\partial w_i}, \frac{\partial L}{\partial \mu_i}, \frac{\partial L}{\partial \Sigma_i}] \tag{4.11}$$

$\frac{\partial L}{\partial w_i}$ is a K-dimensional vector of the 0^{th} order statistics, $\frac{\partial L}{\partial \mu_i}$ is a KD-dimensional vector of 1^{st} order statistics, and $\frac{\partial L}{\partial \Sigma_i}$ is a KD-dimensional vector of the 2^{nd} order statistics of the gradient for each x_n. Exact derivations of $\frac{\partial L(x_n|\theta)}{\partial \theta_i}$ can be found in the appendix section of [32].

Implementation Details: Essentially we used the same Fisher vector computation pipeline as described in [19]. We start by computing SIFT features at multiple scales, from which GMM components are computed using the *yael library* [1]. Fisher vectors representations are then computed for all the images as described above. We use the Fisher vector representations to compute the kernel matrix for SVM learning using Eq. 4.2. As described in [15], the information matrix I can be safely ignored, thus we replace I in Eq. 4.2, with an identity matrix. An empirical approximation to Fisher information matrix is shown in [19], that uses a diagonal approximation for I. It is shown that performing whitening normalization on the Fisher vectors approximates the effect of I well. Whitening normalization ensures that the final gradient vectors have zero-mean and unit-variance. As described in [34], we performed three kinds of normalizations, (a) whitening normalization, (b) power normalization, (c) L2-normalization in the same order to arrive at the final image representation.

Discussion: Computing the Fisher representation of an image involves computing K 0^{th} order terms, $K \times D$ 1^{st} order terms, and $K \times D$ 2^{nd} order terms which are calculated using Eqns. 4.8, 4.9, 4.10 respectively. This gives us a final representation of size $K(2D+1)$, where D is the size of each feature. For SIFT features of size $D = 128$ and $K = 50$ Gaussian components, the final image representation size is 12850. The representation size increases linearly with the number of Gaussian components. In such high dimensional spaces, SVM classifier training is nearly intractable. Hence precomputing the kernel matrix (using Eq. 4.2) is indispensable. Computing the kernel matrix requires $O(n^2)$ dot products and $O(n^2 \times (K(2D+1))$ computations. Recent works [6], further use 8 spatial regions which increases the representation size and computations 8 fold. In section 4.4.3, we demonstrate this is unnecessary; we achieve comparable performance without using spatial pyramids, which reduces our computational and storage requirements.

4.3 Enhancing Fisher Vector Representations

Computing the Fisher vector representation of images involves several independent steps. The various steps in this pipeline can be enumerated as (i) computing low level features (SIFT) for images, (ii) building a generative pdf model (GMM), (iii) building Fisher vector representation for images using GMMs and low level features for respective images, (iv) SVM Classification. We intend to introduce changes to the traditional methods of implementing these steps. Improving any one or more of these results in better performance of the complete system.

[1] https://gforge.inria.fr/projects/yael

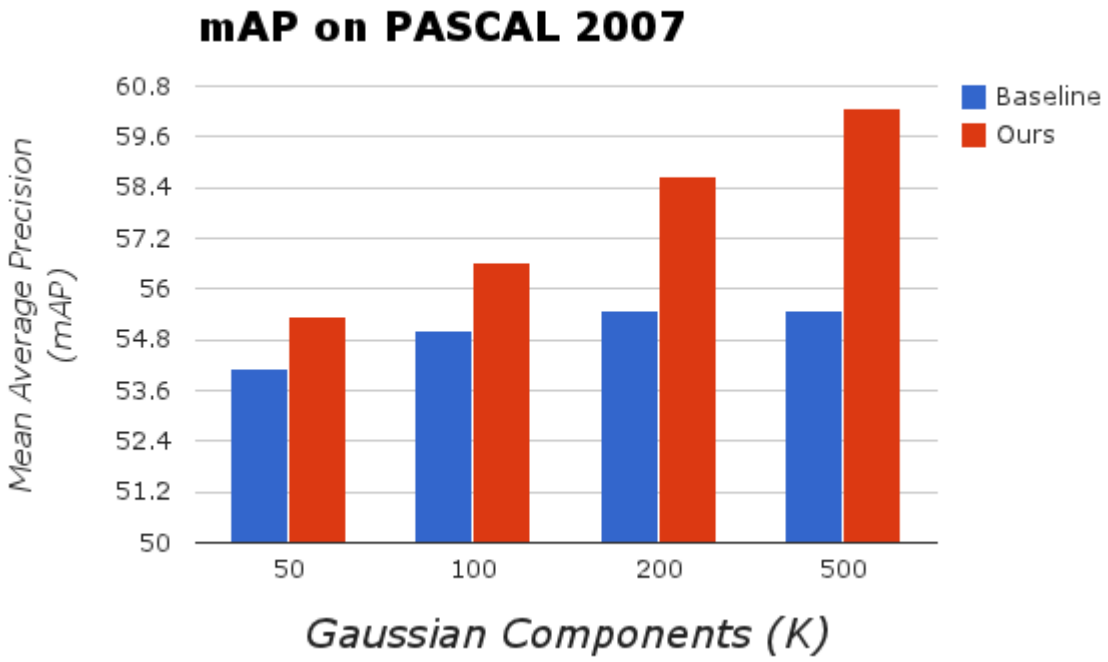


Figure 4.2 Baseline Results showing mAP on PASCAL VOC 2007 dataset based on our implementation of [19]

Baseline: In this chapter, we treat the traditional Fisher vector representation (as described in [19, 34]) as our baseline representation and describe several ways to improve it. Our baseline representation is described in section 4.2.1. The classification scheme described in section 4.4 remains unaffected. In the rest of this section, we describe modifications or additional steps that we introduce in the feature encoding procedure. Each of these supplements the classification performance of the baseline representation. We back this claim by producing superior results on four datasets in section 4.4. We argue that our final representations are both more compact and more discriminative, and come at little computational and storage price. Results on the Pascal VOC 2007 dataset using the baseline Fisher vector representations are shown in Fig. 4.2. We also show results using our improved Fisher representations (with compatible representation sizes) alongside. Our representations significantly outperform the traditional representations.

4.3.1 Root Sift

Recent works on classification [48] have endeavoured to find better distance metrics in the feature space. Inspired by [1], we experimented with various kernels in the SIFT feature space and found that the Intersection and Hellinger Kernels enhance the classification performance. This is in line with the findings of [1, 39, 48]. Table 4.1 shows the effect of variation of the distance metric/kernel on the

classification performance. The dataset here is Scene 15 and the number of Gaussian components are 25 and 50 in columns 2 and 3 respectively. This result motivated us to use a different distance metric in the SIFT feature space. Both the Intersection and Hellinger kernels can be approximated by computing corresponding explicit feature maps on the SIFT features. However, while the intersection kernel feature map increases the feature dimension 3 folds, the Hellinger kernel preserves the size of the input features. We therefore choose to map our SIFT features using the Hellinger feature map. As described in [48], this mapping enables us to use linear kernel to approximate the Hellinger kernel.

Table 4.1 Effects of different distance measures for SIFT Features on classification performance (Scene 15)

Distance Measure	$mAP(K = 25)$	$mAP(K = 50)$
Euclidean (Baseline)	85.03	86.30
χ^2	85.62	87.00
Jensen-Shannon	85.40	87.63
Intersection	86.04	87.51
Hellinger	85.96	88.18

As described in [1], this mapping consists of two simple steps: (i) L1 normalizing each SIFT feature, and (ii) Square rooting the individual values of 128 dimensional vector. Once this transformation is done, we can simply use this transformed SIFT feature namely **"RootSIFT"** [1] as it is in our pipeline.

4.3.2 Sparse Fisher Vector

We denote the N SIFT features in an image by the set $X = \{x_n, n = 1, \ldots, N\}$ and the GMM model parameters by $\theta = \{w_i, \mu_i, \Sigma_i, i = 1, \ldots, K\}$. Using Eq. 4.6, we compute a K dimensional feature assignment vector for each feature x_n.

$$\vec{q_n} = \{q_{n1}, q_{n2}, \ldots, q_{nK}\} \tag{4.12}$$

The k^{th} entry in $\vec{q_n}$ (q_{nk}) represents the soft assignment value of feature x_n to the k^{th} Gaussian component. We define the image assignment vector $\vec{Q}$ as the sum of all feature assignment vectors in the image.

$$\vec{Q} = \sum_{n=1}^{N} \vec{q_n} \tag{4.13}$$

The k^{th} value in $\vec{Q}$ represents the sum of assignment values of all the SIFT features in that image to the k^{th} Gaussian component. Having obtained $\vec{Q}$, computing the sparse image representation involves picking k' out of the K Gaussian components. In this chapter, the k' components we pick are the ones with the most population. This model assumes that the Gaussian components that occur the most in an image are the most representative of the image. This is implemented as picking the k' components

that have the highest values in $\vec{Q}$. The sparseness is introduced by ignoring $K - k'$ components. In other words, we compute the Fisher vector representation for the chosen k' Gaussian components only and values corresponding to rest $K - k'$ Gaussians are ignored (set to zero). Our final Fisher vector representation has a size of $k'(2D + 1)$.

Let G' be the set of these top k' Gaussians. Note that we have to update our Gaussian weights w_i to ensure they add up to 1 after this modification (as described in section 4.2.1). $\sum_{i \in G'} w_i = 1$. This is achieved using Eq. 4.14

$$w_i = \frac{Q_i}{\sum_{j \in G'} Q_j} \tag{4.14}$$

In the next section, we describe an alternate strategy to pick the k' Gaussian components out of K. The alternate strategy allows us to encode class-discriminative information in the Fisher vectors and hence is more useful.

Implementation Details: We now describe an implementation trick that enables us to reduce the storage requirements and computations for our sparse representation. Having computed the G', we compute the Fisher vector representations using Eq. 4.11. From Eqns. 4.9 and 4.10, it is clear that if $q_{ni} = 0$, the gradient value for that particular Gaussian is also zero. Since all but the values corresponding to the top k' Gaussians are zero, we do not have to explicitly compute all of them. Instead, for each image we save the indices of Gaussians for which $q_{ni} \neq 0$. Now we compute the Fisher vectors for G'. Our final image representation is thus of size $2k'[D + 1]$ (the $k'[2D + 1]$ values for Fisher vector representation corresponding to k' Gaussian components and an additional k' for storing indices of non-zero q_{ni}). However, implicitly our representation is that corresponding to K. By removing these least representative Gaussians for an image we gain two advantages:

1. Since our implicit representation for an image is sparse with only $k'(2D + 1)$ non-zero values, while computing kernel $k(F_i, F_j)$, we will have to make lesser computations (only for non-zero values). This can be achieved with a little implementation trick, where we first compute the intersection of the non-zero indices of F_i, F_j and then compute the dot products of the intersecting components since the other products all result in zeros.

2. Though our computations are done only on the non-zero values, the implicit length of Fisher vector still corresponds to the original K. By choosing appropriate the value of k' judiciously, we can achieve performance comparable to K by using only k' Gaussian components.

4.3.3 Discriminative Fisher Vectors

In this subsection, we describe a simple approach that encodes class-discriminative information in the Fisher Vector representation of images. Our approach involves computing pointwise mutual infor-

mation (P.M.I.) [2] between the class labels and the Gaussian components. Given C classes, we define the class assignment vector ($\phi_c, c = 1, \ldots, C$) as the distribution of the gaussian components in images corresponding to the class. Let $\vec{Q}_t, t = 1, \ldots, T$ be the image assignment vector (computed using Eq. 4.13) corresponding to an image t. Let $\mathbf{t}_c$ be indices t of images belonging to class c. ϕ_c is computed as

$$\phi_c = \sum_{t \in \mathbf{t}_c} \vec{Q}_t \tag{4.15}$$

Computing P.M.I. between class labels and the Gaussian components involves computing all $\phi_c, c = 1, \ldots, C$. Let ϕ_{ck} denote the entry in ϕ_c corresponding to the k^{th} Gaussian component. The P.M.I. I_{ck} between class c and the Gaussian component k is computed as:

$$I_{ck} = \phi_{ck} \log \left(\frac{\phi_{ck}}{\sum_{c=1}^{C} \phi_{ck} \sum_{k=1}^{K} \phi_{ck}} \right) \tag{4.16}$$

We use the P.M.I. values as multiplicative coefficients to the weights of Gaussian components. More precisely, we update each w_k by multiplying it with the corresponding I_{ck} to build the GMM for class c. This class specific GMM is used for computing Fisher Vectors for all images. Thus, intuitively, by computing I_{ck}, we are trying to gauge which Gaussian components are most representative of a class. Multiplying I_{ck} with the weights of the Gaussian components ensures that we use more information from the most representative components. This gives us class specific Fisher vector representation of all images without having to compute class specific vocabularies.

Discriminative Sparse Fisher Vectors: This strategy can further be used to improve our method of picking the most representative k' Gaussian components per image (section 4.3.2). Instead of choosing the most populated k' Gaussian components in each image, we use the P.M.I. values to decide which components are most representative of a class. More specifically, we pick the k' Gaussian components per class which share most mutual information with the class. The sparcity induced using this method promotes class level discrimination as opposed to image level discrimination. Note that while we picked the most representative Gaussian components for each image in section 4.3.2, here we pick the most representative Gaussian components corresponding to each class. Intuitively, this strategy makes more sense in a classification problem setting. We back this claim with results in section 4.4.1. In table 4.2 , we study the effects of introducing sparsity and discriminative information in isolation. The dataset used in these experiments is Scene 15. We show the effects on mAP for four vocabulary sizes (number of Gaussian components). In this set of experiments, we chose $k' = K/2$ and for sparse representations, the values of k' are those in column 1. It can be observed that both sparseness and discriminative information enhance classification performance in isolation. These gains are more marked at lower vocabularies.

[2] Pointwise Mutual Information between two random variables X (e.g. cluster labels from a clustering method) and Y (e.g. actual class labels) is defined as: $pmi(x, y) = \log \left(\frac{P(x,y)}{P(x)P(y)} \right)$ where $P(x, y)$ is the joint probability of x and y and $P(x)$ and $P(y)$ are the marginal probability distributions of X and Y respectively.

Table 4.2 Effects of introducing sparseness and discriminative information in Fisher vector representations on mAP (Scene 15).

Vocabulary	Baseline	Discriminative	Sparseness	Both
20	84.89	86.84	87.48	88.04
50	86.30	88.64	88.43	88.91
100	86.80	88.81	89.46	89.31
250	87.19	89.40	89.47	89.59
500	87.50	89.60	89.53	89.60

4.3.4 Discussion

We now discuss the implications of the three improvements we have introduced to the baseline approach. Root SIFT is a preprocessing step in our algorithm and does not affect the rest of the pipeline. Computing root sift involves little computational overhead, precisely $O(1)$ for each SIFT feature.

Introducing sparsity in Fisher vectors enables us to enjoy performance comparable to more Gaussian components while adhering to the storage and computational requirements of a smaller number of Gaussian components. The implementation strategy described in section 4.3.2 enables us to achieve this objective. Empirical results in section 4.4 indicate that we can achieve performance worth K components by choosing as less as $k' = K/2$ components. This is a significant improvement and reduces the memory requirements and computations by half.

Computing discriminative Fisher vectors involves computing the multiplicative coefficients. Computing $\mathbf{I}_{ck}$ involves $n' \times K \times D$ computations where n' is the total number of training features. We compute a different Fisher vector representation per class which are used to train a classifier for this class.

It should be noted that all these modifications affect only the Fisher vector computation process. The other stages like GMM learning and classification remain unaffected.

4.4 Experiments and Results

In this section, we evaluate our method on some standard visual classification datasets and provide empirical evidence to demonstrate the superior performance of our method over the traditional Fisher representation. Further, we also compare our performance with the state of the art classification approaches. We show results on 4 popular datasets (Figure 4.3): (a) two scene classification datasets

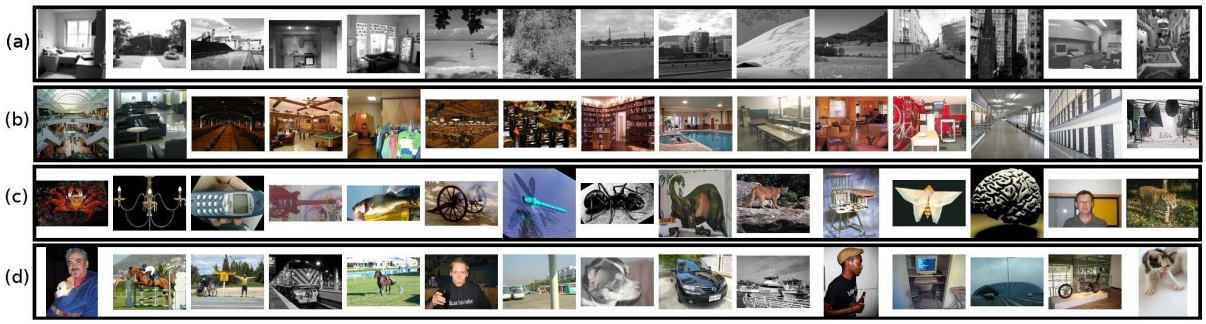


Figure 4.3 Random Images sampled from the four datasets: (a) Scene 15, (b) Scene 67, (c) Caltech 101 and (d) PASCAL 2007

Table 4.3 Specifications of the datasets we used.

Dataset	#Classes	#Images (Train/Test)	Evaluation
Scene15	15	4485 (1500/2985)	mAP
Scene67	67	6700 (5360/1340)	Accuracy
Caltech-101	101	5975 (3030/2945)	Accuracy
PASCAL VOC 2007	20	9963 (5011/4952)	mAP

namely Scene15 and Scene67 and (b) two object recognition datasets namely Pascal VOC 2007 and Calech 101. Table 4.3, describes the datasets we use in our evaluations.

Experimental Setup: All our SIFT features on Pascal VOC 2007 and Caltech 101 were computed on multiple scales of [4, 6, 8, 10] and a step size 3, as in [6]. For datasets Scene 15 and Scene 67, we computed SIFT on a single scale of 12 and a step size of 6. We used the vl_feat library [46] for SIFT computation. The number of Gaussian Components for building the Fisher Vectors were different and have been described in the following subsections. We used the *yael library* for GMM computation. For all the datasets, we trained a one-vs-rest svm [7] for each class. For Caltech101 and Scene 67, the test sample was assigned the label of the classifier that gave the maximum score and we report the

Table 4.4 Comparison with state of the art approaches on the 4 Datasets

Dataset	State of the art	Our
Scene 15	88.18 [55]	89.60
Scene 67	43.1 [29]	49.87
Caltech101	77.78[6]	76.16
PASCAL VOC 2007	61.69 [6]	61.09

classification accuracy while for the other datasets, we report the mean Average Precision of the retrieval task.

Experimental Details: Our baseline Fisher representation has been described in section 4.3. We use this baseline in all empiricial comparisions with our enhanced representations. In some of these experiments, we empirically compare the effect of varying k'. As described in section 4.3.2, by choosing the k' judiciously, we can achieve performance comparable to K with the storage and time complexity corresponding to k'. Since our final Fisher vector representation sizes depend linearly on K, this reduction in time and space complexity is also directly proportional to the difference between K and k'. For comparision with the state of the art approaches, we selected K and k' after extensive experimentation using a validation set.

4.4.1 Scene 15

Table 4.5 Comparison with the state of the art approaches on Scene 15

Method	mAP
SPM[39]	81.40
DSSIC [40]	85.50
Sun [55]	88.18
Sparse Discriminative FV	**89.60**

The Scene 15 dataset consists of 4485 images split over 15 different scene categories. As in [39], we randomly choose 100 images per category for training and the rest for testing. We repeated the experiments 5 times and report the mean Average Precision of the retrieval task.

Figure 4.4 demonstrates the effect of varying k' on the mAP. In these experiments we gradually increase k' from $20 - 100$ using a step size of 10 by fixing $K = 100$. It can be seen that our sparse representation with comparable explicit implicit dimension almost always beats the baseline representation. Another observation is, at implicit size $k = 60$, our method even outperforms that of $K = 100$. This indicates that choosing the implicit representation size wisely can produce results better than those obtained using a higher representation size. Table 4.5 compares our method with the recent state of the art approaches. We beat all published results on this dataset. The previous best state of the art approach described in [55] uses 14 low level features. Our best performance was achieved at $K = 750$, $k' = 400$.

4.4.2 Scene 67

The Scene 67 dataset [36] has 67 Indoor categories, and a total of 15620 images. However, as in [36], we choose a fixed set of 80 images per class for training and 20 for testing.

Figure 4.5, shows the effect of increasing k' on scene67 dataset. As compared to Scene15, the effect of varying k' is more significant on this dataset. The performance at $k' = 25$, is lower than that of

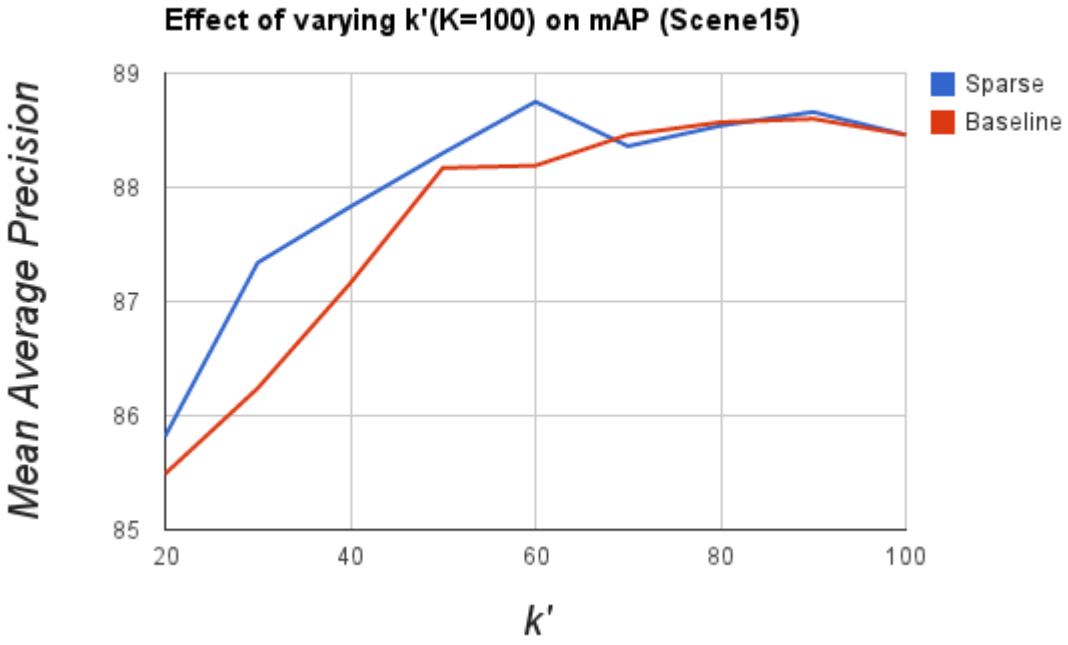


Figure 4.4 Effect of varying k' on the mAP on the Scene 15 dataset. K=100.

the baseline representation. The gain in performance happens as we move beyond $k' = 50$. Further increasing k' beyond 100 (note that at this poing, $k' = K/2$), gives significant improvement in accuracy and as we approach $k' = K = 200$, the accuracies begin to converge to the one corresponding to $K = 200$. After extensive experimentation on Scene 67 and other datasets, our findings indicate that in general, we should choose $k' >= K/2$.

Table 4.6 compares our method with the state of the art approaches. The parameters $k' = 650$ and $K = 1000$ give us our best performance quoted in the table.

4.4.3 Caltech 101

The Caltech 101 dataset has 9146 images coming from 101 distinct object categories. In our experiments, we randomly sample 30 images for training from each of the 101 categories, getting a total of 3030 training images; we test our method on the remaining images. We however limit the number of testing images per category to 50, as in [39]. We repeated the experiments on Caltech 101 with 5 such random subsamplings and report the mean classification accuracies over the five experiments.

Table 4.7 compares our method with the state of the art approaches. We achieve classification accuracy comparable to [6]. They have used the traditional Fisher kernel representation with 256 Gaussian components and spatial pyramids on top of the Fisher representation. It should be noted that while they

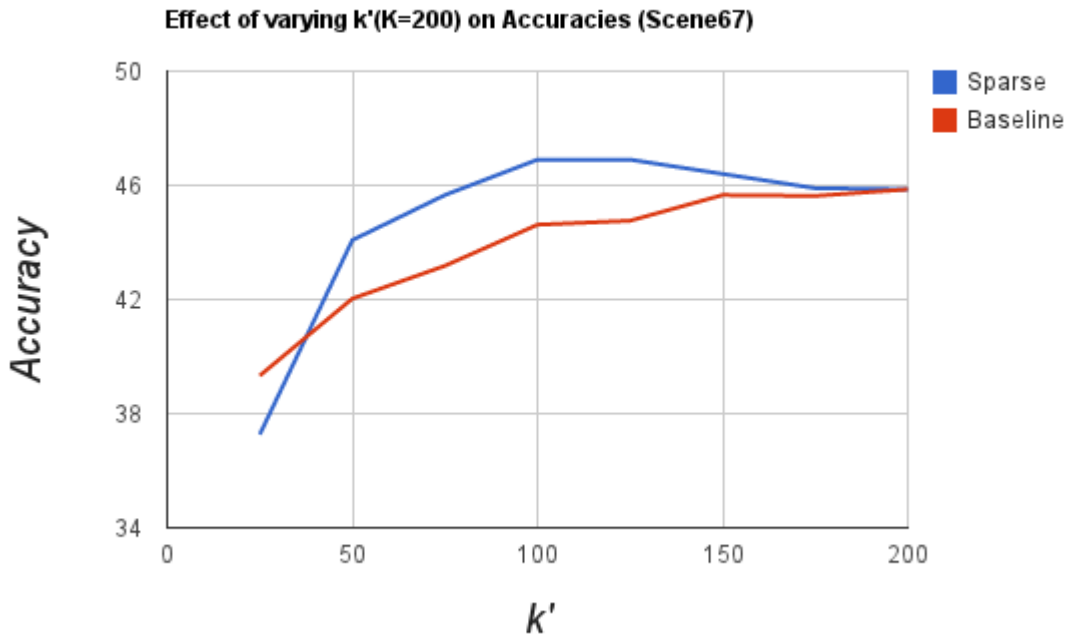


Figure 4.5 Effect of varying k' on the mAP on the Scene 15 dataset. K=100.

used 8 spatial regions, we did not use spatial pyramids. Our best results were achieved at $k' = 200$ and $K = 250$. Hence the representation size used in [6] is more than 8 times that of ours.

4.4.4 Pascal VOC 2007

PASCAL VOC 2007 data has 5011 training images and 4952 testing images in 20 classes. This is one of the hardest datasets used for visual classification as the objects in these images vary significantly in scale and orientation and some of the images have multiple objects present in them.

Table 4.8 reports the mAP achieved by the state of the art approaches on Pascal 2007. Our performance with $k' = 175$ and $K = 250$ is comparable to [6]. The results in [6] were achieved by using Fisher vectors with 256 Gaussian components and spatial pyramid with 8 spatial regions on top, which is around 12 times of our representation size.

4.4.5 Discussion

Empirical evaluations on 4 datasets indicate the utility of our approach. We outperform all previously published results on the scene classification datasets. On Caltech 101 and Pascal 2007, not only do we manage to match the state of the art classification rates (which they report were achieved using the

Table 4.6 Comparison with the state of the art approaches on Scene 67

Method	Accuracy
ROI+GIST [36]	26.5
Pandey*et.al* [29]	43.1
CENTRIST [54]	36.9
Object Bank [24]	37.6
Sparse Discriminative FV	**49.87**

Table 4.7 Comparison with the state of the art approaches on Caltech 101

Method	Accuracy
SPM [39]	64.60
LLC [51]	73.44
DD-FV [6]	**77.78**
Sparse Discriminative FV	76.16

traditional Fisher vector representations with spatial pyramids), our explicit representation sizes are much less. These results are highlighted in table 4.4.

Our experiments on all the datasets revealed a common observation: if we desire a representation that uses k Gaussian components, it is always better to pick these k components from a larger vocabulary than compute a vocabulary of k Gaussian components; the former always outperforms the latter. This gain in performance can be attributed to the implicit representation size which is always bigger than the explicit representation size. Like most sparse representations, the zero entries in our representation have as much useful information as that contained in the non zero terms.

Our primary motivation behind introducing sparseness in Fisher vectors was to counter the huge representation size. Product Quantization [17, 38] has been described in literature to reduce the storage requirements. These approaches generally encode the high dimensional features and store the compressed code on the disk. However, these codes are decoded on the fly when the original features are for classification. A recent work [48] describes a strategy that uses the Product Quantization codes itself in classification.

Table 4.8 Comparison with the state of the art approaches on Pascal

Method	mAP
KCB [43]	56.26
SV [58]	58.16
LLC [51]	59.74
DD-FV [6]	**61.69**
Sparse Discriminative FV	61.09

4.5 Summary

Most practical solutions require us to understand and resolve the tradeoff between performance and resources. Fisher vectors demonstrate this concept well. The representative power of Fisher vectors comes at a storage and computational cost. While most recent works in this direction have attempted to compress Fisher representations to save disk space, we have tried to address another important issue: reducing the representation size of Fisher vectors. We devised a novel sparse representation for Fisher vectors. Our sparse solution is desirable because it combines two elements: (a) the power of sparse representation, and (b) reduction in the explicit representation size. Further, we introduced a novel strategy for the discriminative computation of Fisher vectors. This further boosts our classification performance. In this chapter, we described two approaches of picking the most representative / discriminative Gaussian components from a pool of Gaussian components to introduce sparcity. Developing better schemes for picking these Gaussian components is a research problem in its own right. We evaluated our method on four popular visual classification datasets and our method consistently outperformed the baseline Fisher representation. The performance of our model is at par with the state of the art approaches on these datasets.

As shown in chapter 3 and chapter 4, by changing the encoding/assignment methods, we can get better image representations, which in turn will help improving our model for solving the problem in hand. In the next chapter, we will try to address the issue of building a proper vocabulary set. So, in the next chapter we will show how we can combine different vocabularies constructed from different features, sizes, in different spaces, can be combined together by choosing the best representing visual words from each of them. We will apply this new technique on top of enhancements done in the previous chapters to show that it further improves our learning model.

Chapter 5

Discriminative Visual Words

5.1 Introduction

In recent years, many supervised methods have employed Bag of Words (BoW) [8, 41] based models as simple but effective means of representation. Many of the state of the art approaches [6] on popular datasets and recent object recognition challenges employ BoW representations in some form or the other. BoW models approximate the distances in the feature space (128D SIFT space for instance) by a $0 - \infty$ metric, where the features corresponding to the same visual word are seen as exactly alike while features corresponding to different visual words are seen to be totally different. While this reduces the computations significantly, it also sacrifices some discriminative power. Good visual vocabularies are inherently discriminative and aim to compensate for this loss.

There have been many attempts at devising strategies for discovering visual codebooks. These include clustering approaches, class-specific vocabularies, supervised codebook building methods, and even approaches involving a manual labeling of image patches. However, despite the huge volume of literature on vocabulary construction methods, unsupervised clustering approaches, such as K-Means still seem to be the most popular ones [6].

The number of visual words in a codebook is a crucial parameter that dictates the complexity and quality (fineness) of the representation. If the number of visual words is less, the representation is coarse, and the visual words do not represnt all the keypoints (local features) properly. If the codebook is large, the representation is fine, and may cause quantization artefacts and overfitting. An optimal number of visual words ensures good representation of all keypoints; however, it does not ensure a discriminative representation. Visual words that may help describe one class of images may not be useful for other classes. Thus, in a discriminative task, it may be useful to discard irrelevant visual words. One might ponder, *which visual words are discriminative, and which are irrelevant?* In this chapter, we explore this gray area.

Recent advances into BoW methods have suggested the use of more features[55], huge vocabularies [6], pyramidal representations [39], and data kernels [32] for enhancing classification performance. All these methods increase the size of the representation. The old adage about the *curse of dimension-*

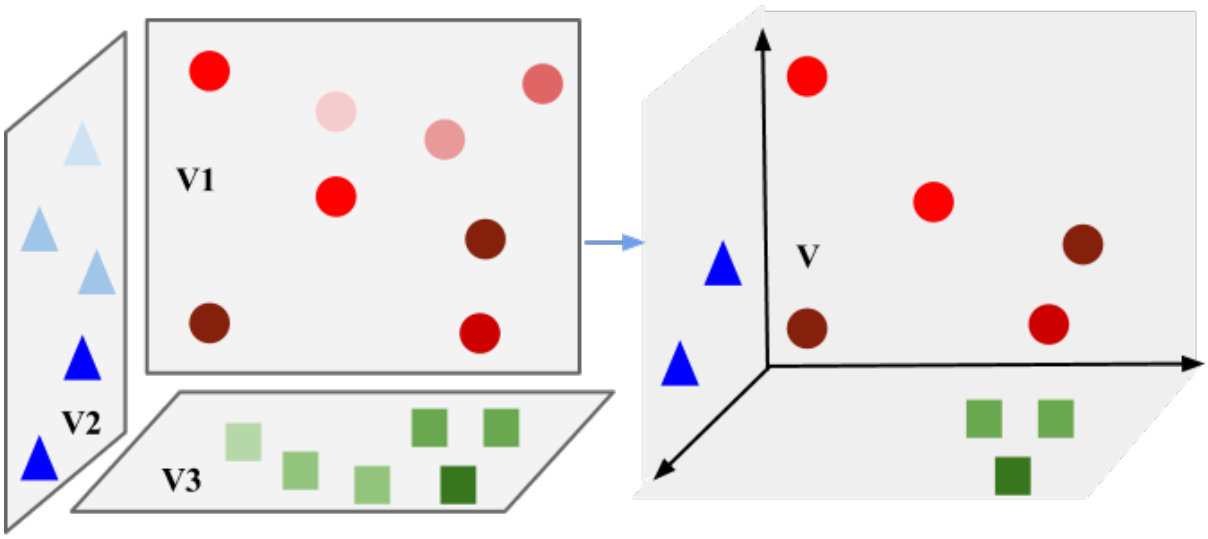


Figure 5.1 V1, V2 and V3 are three different visual vocabularies with visual words whose colour saturations reflect their discriminative power. V is a vocabulary constructed by combining V1,V2, and V3, and rejecting the words with less discriminative power.

ality dictates that the natural next step from this point on should be to reduce the feature size, without sacrificing the discriminative power of the representation. This chapter is intended to achieve this very goal.

Often it is feasible to compute multiple codebooks since the vocabulary construction phase is usually an offline step and done only once. These different vocabularies could come from (a) different types local features, (d) different initializations of K-Means, (c) different strategies of codebook construction from the same feature set, and so on. However, using all the vocabularies is often not desirable because (a) of the computational complexity involved and (b) some vocabularies may introduce noisy words which hamper the performance. In this chapter we describe a novel approach to choosing the class-wise, discriminative set of visual words/vocabularies drawn from a larger pool of vocabularies. Our primary contribution is proposing simple, intuitive solutions to two fundamental questions: (a) how good is a visual word?, and (b) how good is a codebook?

5.2 Words in the Bag

Visual Bag of Words (BoW) models are orderless representations that represent images as frequencies of visual words coming from a codebook. They are efficient, reasonably discriminative and robust to a number of nuisance parameters such as scale, translations, illumination changes, and other deformations. They often work by quantizing SIFT or other features to a symbol space which is necessary in large scale problems because (a) computing distances between a huge number of descriptors is ineffi-

cient, and (b) storing all the descriptors is not feasible. BoW models give a fixed size representation for all images, regardless of their size, and number of visual words. This is particularly suited for most of the learning algorithms that assume that the input space has a fixed size.

5.2.1 Visual Vocabularies

Visual vocabularies allow efficient indexing for local image features. There are several methods of codebook construction in literature. Typically, the vocabulary is constructed by clustering the local image features using the K-Means algorithm [39, 6]. K-Means partitions the space well, minimizing the variance between the data points and the cluster centers. However, findings in [18, 3, 37] suggest that the most frequently occuring words are not necessarily the most discriminative. Motivated by this hypothesis, a number of methods have attempted to build discriminative vocabularies by (a) alternative clustering algorithms [18, 25], (b) using class label information [27, 52, 57], and (c) even creating class-specific categories as in [30, 21, 42].

Other approaches to vocabulary construction involve manual labeling of image patches with semantic labels, as in [44, 50]. These semantic vocabulary approaches are more intuitive because they try to express the image in terms of what the constituent visual words in it mean.

Vocabulary trees [28], that employ hierarchical K-Means, allow recursive division of the feature space, enabling us to efficiently reduce the computational cost of assigning visual words to features, from linear to logarithmic in the size of the vocabulary. This allows us use much larger codebooks. Also, empirically, these visual words are more specific and are particularly suited to matching specific instances of objects.

It is worth considering what it is that a visual word captures. This depends on a variety of factors: (a) what data was used to construct the vocabulary, (b) what local features were computed, (c) the size of the codebook, (d) the vector-quantization strategy used, and (e) how the interest points in the original images were chosen. Regardless of these factors, it is reasonable to assume that keypoints assigned to the same visual words have a similar appearance in the low-level feature space. Also, in most practical solutions (where we use unsupervised methods for codebook construction), there is usually no guarantee of correlation between the visual words and the object parts/artefacts.

There are many choices available when creating a visual vocabulary. Each of these choices affects the semantics of our representation and the accuracy of our solution. However, there is still no clear consensus on the *correct* method of learning a visual vocabulary. The analogies between textual and visual data are only approximate: textual words are discrete, symbolic language defined semantic constructs, while natural images are continuous, complex entities. “Real sentences have a one-dimensional structure, while images are 2D projections of the 3D world.” [14]. While the search for the *universal* vocabulary continues, in this chapter, we describe an approach that helps us experimentally determine which vocabularies are better than others in a discriminative setting.

5.2.2 Soft Assignments

An important component of the BoW pipeline is the assignment of keypoints to visual words in the codebook. Traditionally, BoW models assign a keypoint to the nearest visual word. Soft assigments allow a keypoint to be associated with multiple visual words with varying degrees of membership. This enables modeling the ambiguity of assigning a keypoint to a visual word. Some of the recent methods [45] reveal that soft assignments lead to better representation of an image in terms of the visual word distribution, which in turn enhances the classification accuracy. Table 5.1 demonstrates the superiority of soft assignments over hard assignments on two popular image classification datasets, Scene 15 and Pascal 2007.

	Scene 15	VOC Pascal 2007 [6]
hard BoW	76.20	46.54
soft BoW	84.13	54.60

Table 5.1 BoW - Hard vs Soft Assignments on Scene15, Pascal VOC 2007

5.2.3 Fisher Vectors

Since their introduction to the image classification domain in 2007 [32], Fisher Kernels [15], have been employed by many state of the art approaches [6]. While BoW approaches use a $0 - \infty$ metric to approximate distances between keypoints, thus losing a lot of discriminative information, Fisher vectors assume the data distribution to be a Gaussian Mixture Model (GMM) and capture the discrepancy in the fit, thereby capturing the distances between the keypoints and the representative visual words. Since Fisher Vectors employ GMMs which are inherently soft assignmened, Fisher Vectors are also soft representations. Both these characteristics contribute to the discriminative power of these representations. However, this power comes at a huge computational and storage price. A fisher vector for K Gaussian components has a final representation size of $K(2D + 1)$, where D is the dimension of the local features, and requires computations of the same order. Recent approaches have attempted to compress Fisher vectors [17, 33, 38] to cope up with the high memory requirements. These methods have primarily intended to reduce storage requirements by compressing the huge vectors and uncompressing them on the fly during classification. In section 5.5, we describe strategies to reduce the representation size of Fisher vectors withour sacrificing discriminative power.

5.3 Which Visual Words are important?

As stated in section 5.1, the size of the codebook has a bearing on how well the visual words represent the keypoints. However, the most frequent words are not necessarily the most discriminative. In this section, we describe a simple metric that represents the discriminative power, and hence the importance of a visual word with respect to a task. Using this knowledge, we can improve the discriminative power

of a vocabulary by discarding unimportant visual words. In other words, we can reduce the codebook to a subset of important and discriminative visual words. By doing so, we hope to achieve a more compact and yet more discriminative representation. Such a framework would yield clues that help us understand which visual features are important for recognizing which categories. This can be seen as non-linear dimensionality reduction by feature selection, and our hope is to enhance classification performance by discarding the irrelevant visual words. Another advantage of rejecting visual words is that it yields a smaller vocabulary, and hence a smaller representation. This is particularly advantageous in case of very huge representations such as Fisher Vectors.

5.3.1 Class-wise Word Scores

We now describe a strategy to assign scores to visual words drawn from a vocabulary. Each visual word is assigned one score per class, and this score represents how important the word is for recognizing the class.

Visual words $v_k, k = 1, \ldots, K$ are words drawn from a vocabulary $\mathbf{V}$ of size K. $c_i, i = 1, \ldots, C$ represent the C classes/categories. Our hypothesis here is: a visual word is important for a class if (a) it occurs frequently in samples of that class, and rarely in samples of the other classes, **or** (b) it occurs rarely in samples of that class, and frequently in samples of the other class. We define the relative population γ_{ik} of visual word v_k in class c_i as follows:

$$\gamma_{ik} = \frac{p_{ik}}{\left(\sum_{l=1, l \neq k}^{K} p_{il} \right)} \tag{5.1}$$

where p_{ik} is the probability of the visual word v_k occuring in class c_i. This probability p_{ik} is dictated by the number of times v_k occurs in samples belonging to class c_i. p_{ik} is given by:

$$p_{ik} = \frac{n_{ik}}{\sum_{j=1}^{C} \sum_{l=1}^{K} n_{jl}} \tag{5.2}$$

Note that γ_{ik} captures only the relative frequency of a visual word in a class with respect to the other class, and is not the discriminative score we seek. Based on our hypothesis, we need a function over γ_{ik} that fires when the relative population of a word in a class is low or high and suppresses the intermediate values. To achieve this, for a particular class we subtract the median of all the γ_{ik} from the individual values and then take the absolute. The class-wise word score α_{ik} is thus defined as:

$$\alpha_{ik} = |\gamma_{ik} - median_k(\gamma_{ik})| \tag{5.3}$$

Computing α_{ik} for all (c_i, v_k) pairs thus requires us to populate the counts n_{ik} for all such pairs. In the next section, we describe how we exploit the class-wise word scores for improving our vocabulary $\mathbf{V}$.

5.3.2 Improving vocabularies by rejecting unimportant words.

The simple ratio α_{ik} described in Section 5.3.1 captures the importance of a word for recognizing a class. A word with low score is noisy and does not contribute much to discrimination. We can thus discard it. There may be various strategies of deciding which words to reject. One could have a hard threshold on α_{ik} or alternately use a threshold on the number of words that one needs to build a representation. In section 5.5, we describe an approach of doing this where we put a threshold on the proportion of visual words picked from the codebook. We denote this proportion by η. An $\eta = 0.5$ means that we choose half of the high scoring visual words from the codebook.

5.3.3 Discussion

Our approach relies on a simple metric for discarding irrelevant words from a vocabulary in a discrimination task. Thus, given a huge vocabulary, we have a systematic method of selecting the desired number of most discriminative visual words. This can (a) boost the classification performance if the η is chosen wisely, and (b) reduce the representation size; the size of the final representation is directly proportional to η.

The class-wise word scores α_{ik} represent the importance of various words for recognizing the different classes. These can be studied to (a) study what types of visual words help in discriminating two classes, and (b) identify pairs of confusing classes. Intuitively, classes which have the same important visual words are more likely to be confused with each other. We conducted experiments on Pascal VOC 2007 to study the pairs of classes which share the same important visual words. Figure 5.2 displays the heat map representation of our findings. For this set of experiments, K was fixed to 100 and $\eta = 0.75$. Here we first compute the set of the 75 most important words per class and then for each pair of classes count the number of visual words that co-occur in their sets of important words. This gives us the degree of confusion between two classes.

In figure 5.2, the degree of confusion between two classes is proportional to the degree of redness of the cell corresponding to the pair of classes in the matrix. Our experiments suggest that the category birds is highly likely to confuse with dogs, cows and sheep. This can be explained by observing the textural similarity of the feathers of a bird to the fur of the other animals. The category cow confuses with horse and sheep. Overall, the confusion among animal categories is more than that among the other object categories (cluster1). Figure 5.2 reveals that the class diningtable is most likely to be confused with bottle, chair, person and pottedplant (cluster2). This is because these other objects are often seen around the diningtable. Buses are likely to be confused with trains, tv-monitors are likely to be confused with chair and sofa, and so on. Thus, for the challenging Pascal dataset, both the surrounding context and the artefacts on the object contribute to the confusion.

5.4 Which Visual Vocabularies are important?

As described in section 5.1, since the codebook building phase is usually an offline step and typically done once, it may be worth computing a set of vocabularies, as opposed to a single vocabulary. These different vocabularies could come from (a) different types local features, (d) different initializations of K-Means, (c) different codebook building methods from the same feature set, (d) different sizes of codebook computed on the same features using the same method, and so on. Now that we have a metric of assigning importance to visual words, the next question we can pose is, out of the huge pool of vocabularies, *which codebooks are important?*

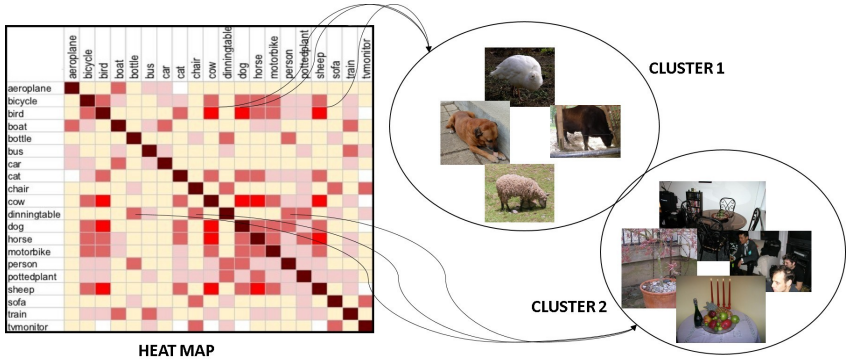


Figure 5.2 HeatMap representation of confusion among various Pascal VOC 2007 Classes. The degree of confusion between two classes is proportional to the redness of the cell corresponding to the pair. Cluster1 shows the confusing animal classes: bird, cow, dog, sheep. Cluster2 shows the confusing classes: dining table, person, bottle, chair.

5.4.1 Class-wise codebook score

Let $V_j, j = 1, \ldots, J$ be the set of visual codebooks, obtained from various different approaches described in the previous section. Our task is now to assign an importance to each codebook V_j per class which reflects how discriminative the codebook is for recognizing the class. Intuitively, a vocabulary is only as good as the visual words in it. We define the class-wise codebook score ϕ_{ij} in terms of the class-wise word scores of visual words in the vocabulary.

$$\phi_{ij} =< \alpha_{ik} >, k = 1, ..., K \tag{5.4}$$

where $< \cdot >$ denotes the average over K words.

The simple metric defined in Equation 5.4 captures the importance of a codebook. We can attempt to improve the classification performance of our method in one of two ways, (a) exploit the knowledge of importance of the codebooks (ϕ_{ij}) to reject poor vocabularies, or (b) mix and match good words from different vocabularies. Note that (b) does not require us to assess how good the vocabularies are.

Thus, the overall solution to building good, discriminative representations involves, (a) first selecting a subset of vocabularies, and (b) selecting the best visual words from the selected vocabularies. In case we choose to not explicity reject vocabularies, we could simply sort all the visual words drawn from all the codebooks with respect to their importance, and choose the most discriminative ones, rejecting the insignificant ones. This may implicitly force some vocabularies to be rejected, because none of their visual words were discriminative enough. Various factors such as domain knowledge, computational and memory constraints may dictate whether to reject certain vocabularies completely, or allow their visual words to be selected.

5.4.2 Discussion

Our metrics α and ϕ enable us to answer the two fundamental questions (a) which visual words are important, and (b) which codebooks are important. Armed with this knowledge, we can take one of two approaches, (a) vocabulary selection, or (b) vocabulary fusion. Vocabulary selection refers to the case where we force certain codebooks with low importance to be rejected while building class-specific representations. This is desirable because of computational, memory constraints. Vocabulary fusion refers to the design choice where we do not reject vocabularies explicitly, and choose the most discriminative words from the set of all visual words coming from all the codebooks. Note that we need not compute ϕ_{ij} explicitly in case of vocabulary fusion.

Traditional schemes for evaluating the power of codebook representations have relied on measuring the quantization error[1]. However, empirical findings have demonstrated that accurate representations are not necessarily discriminative representations. Thus, we do not have a term corresponding to the quantization error in our metric ϕ.

5.5 Experimental Results

This section explores and analyzes our method, and validating our major hypotheses. In this section we apply our knowledge of the discriminative power of visual words to improve our representations for the task of image classification. Additionally, we provide empirical evidence that supports our hypotheses. We work with different types of image representations, and compare our methods with competing approaches.

Datasets: In this chapter, we work with two popular image classification datasets, namely PASCAL VOC 2007 and the Scene 15 dataset.

The Scene 15 dataset consists of 4485 images split over 15 different scene categories. As in [39], we randomly choose 100 images per category for training and the rest for testing. We repeated the experiments 5 times and report the mean Average Precision of the retrieval task.

[1] We define the quantization error of a codebook to be the average distance between the data samples and the corresponding cluster centers

PASCAL VOC 2007 data has a total of 5011 training images and 4952 testing images in 20 classes. This is one of the hardest datasets used for visual classification as the objects in these images vary significantly in scale and orientation and some of the images have multiple objects present in them.

Experimental Setup: All our SIFT features on Pascal VOC 2007 were computed on multiple scales of 4, 6, 8, 10, and a step size 3, as in [6]. For Scene 15, we computed SIFT on a single scale of 12 and a step size of 6. We used the vl_feat library [46] for SIFT computation. Our LBP features were computed at image patches spanning 12×12 pixels, collected at a spatial stride of 6 pixels in both directions. Our Fisher vector representations are based on SIFT, LBP features. For efficiency, we compress the local features (SIFT, LBP) to a subspace of size 64 using Principle Component Analysis (PCA), as in [32]. We use the *yael* library for our GMM computations. For all the datasets, we trained a one-vs-rest *linear* svm [7] for each class. In our experiments involving soft BoW, we additionally use the χ^2 kernel map on top of our representations for better classification [47]. On both the datasets, we report the mean Average Precision of the retrieval task.

5.5.1 Codeword Selection

In this set of experiments, we apply our strategy to enhance an existing vocabulary by rejecting the noisy words in it. We use a threshold on the number of visual words, as opposed to a threshold on the discriminative power α of the visual words. As described in section 5.3.2, we denote the proportion of visual words picked from the codebook by η. Thus, $\eta = 0.5$ means that we chose half of the high scoring visual words from the codebook. From this point on, unless otherwise stated, it is to be assumed that we choose the visual words with the best scores (α_{ik}).

This set of experiments is intended to demonstrate our hypothesis that certain words in a vocabulary are not useful and can be rejected from the vocabulary, thus reducing the representation size. This has a twofold advantage: (a) computational and storage benefits, and (b) more accurate predictions. We try to achieve our objective here by studying the variation of the accuracy of our method with η.

η(K=100)	Scene15	Pascal
1.00	73.76	29.53
0.90	73.77	30.10
0.80	73.76	29.33
0.70	**73.79**	29.15
0.60	73.68	**30.37**

Table 5.2 Variation of mAP on the Scene 15 and Pascal VOC 2007 Datasets with variations in the number of visual words chosen (variations in η). We use dense SIFT features for this set of experiments, and the size of the baseline codebook was fixed to 100.

Table 5.2 describes how the classification performance on the Scene 15 dataset varies with variations in the number of visual words chosen/rejected. Kindly note, we trained a one-vs-rest classifier per class,

and thus a different set of visual words were rejected for each class. The features used here were dense SIFT, and the baseline vocabulary of size 100 was computed using K-Means. Our findings in table 5.2 reveal that the classification performance is best when we choose 60% of the total words in the vocabulary.

5.5.2 Multi-resolution codebooks

Codeword fusion, or *mixing and matching* words drawn from different vocabularies, as described in section 5.4.1, refers to the act of pooling visual words from all codebooks together and simply picking the ones with more discriminative power. In this set of experiments, we demonstrate the application of codeword fusion to build a codebook which has words of different coarseness (multi-resolution). We built SIFT K-Means codebooks of different sizes $25, 50, 75, 100, 125, 150$ on the Scene 15 dataset, and then pooled all the visual words together to get a *universal* vocabulary of size 525. Finally, we picked the 100 most discriminative visual words of these and used them for classification. We report the classification performance of the soft BoW representations in table 5.3.

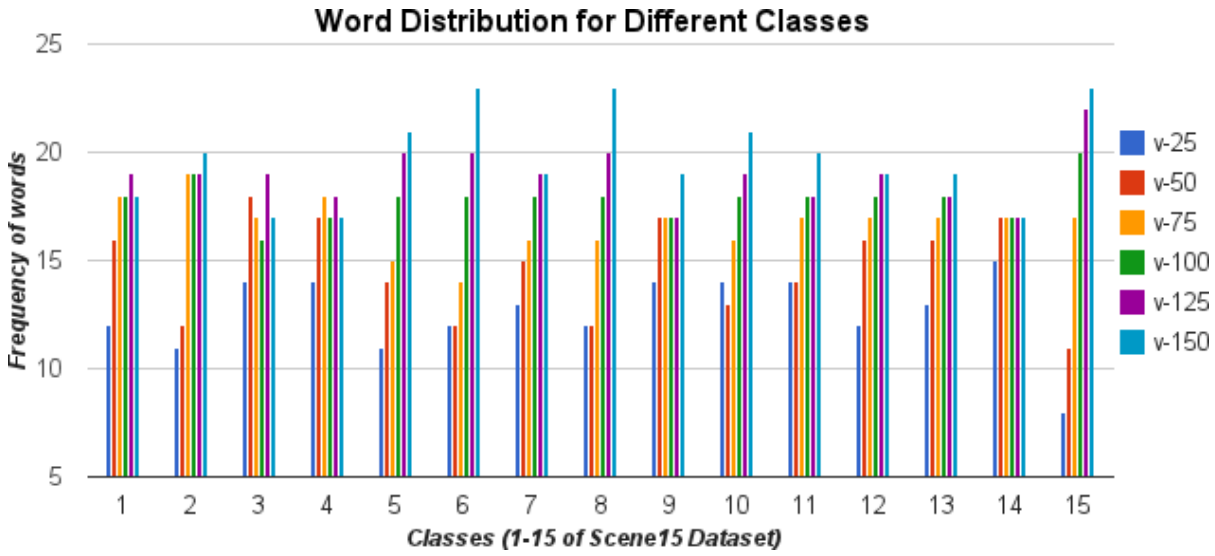


Figure 5.3 Bar charts showing the number of visual words chosen from each vocabulary for each class.

Our findings in table 5.3 indicate that choosing words drawn from a union of codebooks at multiple resolutions of coarseness helps us achieve superior results over the individual ones. Our fusion vocabulary of size 100 outperforms the single-resolution vocabulary of size 125. Table 5.3 shows a bar chart of the number words chosen from each vocabulary for each class. Most classes prefer a larger vocabulary.

Vocab Size	mAP
25	58.27
50	63.53
75	67.70
100	68.62
125	70.55
150	71.48
Fusion (100)	71.08

Table 5.3 Results of combining vocabularies of different coarseness on Scene15 Dataset

5.5.3 Feature Fusion

In this section, we describe an application of the codeword fusion strategy to achieve feature fusion. Feature fusion refers to the use of multiple feature descriptors for building our representation. We use two features in this set of experiments; in principle, however, any number of feature descriptors could be used.

In these set of experiments, we use two vocabularies, one computed on SIFT features, and the other on LBP. Codeword fusion gives us the set of most discriminative visual words (the ones with the highest α scores) drawn from the union of the two vocabularies. We then compute two image representations for each, one using the SIFT words, the other using the LBP words and concatenate the two representations to the final image representation. Kindly note that our representations here are soft-assigned BoW histograms.

The two codebooks based on the two feature descriptors described above had the same size of 200 visual words. This yielded a *universal* pool of 200 visual words after the mixing. From this *universal* pool of 200 visual words, we selected the 100 most discriminative words; hence our final vocabulary size was 100. We finally compute the soft BOW representations for the classification task. Table 5.4, shows the results of this feature fusion scheme on Scene15 and Pascal 2007 datasets. The table indicates that the accuracies obtained using the feature fusion significantly outperforms the accuracies obtained by either of the features alone.

Dataset	SIFT	LBP	Fusion
Scene15	69.82	67.85	**78.90**
Pascal 2007	29.21	22.59	**31.20**

Table 5.4 Results of Feature Fusion on Scene15 and PASCAL VOC 2007

5.5.4 Fisher Vectors

In these set of experiments, we intend to demonstrate two things: (a) the size of Fisher vector representations can be significantly reduced by using codeword selection, and (b) we can fuse different Fisher vector representations based on different low level features.

Table 5.5 shows the Fisher vectors codeword selection results on the Scene 15 and Pascal VOC 2007 datasets. We also show the corresponding representation sizes in each row. Our experiments indicate that the best accuracies are achieved at an $\eta = 0.6, 0.8$ for Scene 15 and Pascal respectively. The reduction in the representation sizes is also significant.

η $(K = 100)$	Scene 15	Pascal	Representation Size
1.00	88.46	54.70	12900
0.90	88.66	54.69	11610
0.80	88.54	54.74	10320
0.70	88.36	54.27	9030
0.60	88.75	53.64	7740

Table 5.5 Variation of mAP on the Scene 15 Dataset with variations in the number of visual words chosen (variations in η). We use dense SIFT features for this set of experiments, and the size of the baseline codebook was fixed to 100.

Our second set of experiments in this section demonstrates feature fusion with Fisher vectors. We build GMM codebooks of size 100 each on SIFT and LBP feature descriptors. As in section 5.5.3, we pool the Gaussion components together to get a *universal* codebook of 200 GMM components. Out of these we choose the 100 most discriminative GMM components to build our final image representation.

Table 5.6 lists the feature fusion results with Fisher vector representations. The gain in performance is very slight (0.05%) on the Pascal Dataset, while it is more significant ($> 3\%$) on the Scene 15 dataset. Thus, feature fusion by codeword fusion generally outperforms representations based on either features alone.

Dataset	SIFT	LBP	Fusion
Scene15	88.18	82.75	**91.83**
Pascal 2007	54.70	39.80	**54.75**

Table 5.6 Results of Feature Fusion using Fisher Vector Image Representations on Scene15 and PASCAL VOC 2007

5.5.5 Comparison with the State of the Art results

In this section, we compare our results to the state of the art results on Scene15 and Pascal2007. Our best performing method uses a feature fusion of SIFT, and LBP words with Fisher representations, described in section 5.2.3.

Authors in [6], have used Fisher Vector image representations to achieve state of the art results on the VOC Pascal 2007 dataset. On Scene 15, [55] uses multiple features to achieve the state of the art results. Table 5.7 compares our results with the state of the art results on Scene15 and Pascal 2007 datasets. While on the Scene 15 dataset, we choose 100 GMM components out of 100 SIFT based, 100 LBP based GMM components, on Pascal 2007, we choose 200 GMM components out of 200 SIFT based, 200 LBP based GMM components.

Dataset	Previous Best	Ours
Scene15	88.18 [55]	**91.83**
Pascal 2007	**61.69** [6]	61.65

Table 5.7 Comparison with State of the Art results on Scene15 and PASCAL VOC 2007

Note that results obtained in [6], for Pascal2007 were obtained using vocabulary (GMM) of size 256 with 8 spatial (4 + 3 + 1) regions on top of it, while our results were obtained without using spatial regions. Also, while authors in [6] reduce their SIFT features to an $80-$dimensional PCA space, we reduce our descriptors to a $64-$dimensional space. Our representations are thus smaller than those used in [6] by a factor of 13. Despite our much compact representations, we are able to achieve results comparable to theirs

5.5.6 Discussion

The ubiquitous BoW models are preferred because of their intuitiveness, simplicity, effectiveness, and efficiency (ISEE). The $0 - \infty$ metric they employ for approximating distances between keypoints makes them efficient, but comes at the cost of discriminative power. We realized that good representations are not necessarily discriminative, and we proposed a solution, that attempts to preserve all the desirable characteristics of BoW representations (ISEE), while enhancing the discriminative performance(ISEED).

Most recent state of the art methods that use BoW representations use complex models, such as kernels, distance functions, and complex classifiers for imrpoving the accuracy of predictions. Almost everything in the book has been attempted [6], for enhancing the accuracy of predictions. Consequently, BoW models today are more complex than they ever were. However, there are many practical solutions that still prefer the traditional (ISEE) BoW models.

Fisher Kernels, which have emerged as the state of art approach on almost all image classification datasets, represent the new class of complex approaches. They improve over the $0 - \infty$ BoW metric, by (a) soft assignments to GMM components, and (b) capturing all the distances between the keypoints and cluster centers. Thus, they have a lot of discriminative information which explains their tremendous success. Our experimental findings indicate that the gain in performance induced by our strategies is quite prominent in case of BoW soft assignments, and only slight in case of Fisher Kernels. To explain this, we should understand the source of power of Fisher Vectors. Fisher Vectors are powerful,

because they encode the discrepancy in the fit well, not because the visual words are more discriminative. However, our strategies are useful for Fisher vector representations still, because they help reduce the representation size significantly.

Table 5.7 compares our results with the state of the art approaches. The best approach on Pascal [6], uses a representation size that is 13 times bigger than our representation size. At this point, we stress that the objective of this chapter is the transformation from ISEE to ISEED, and not to design a state of the art approach. Another observation is, the performance gains are bigger in Scene 15 over Pascal. We attribute this to the fact that Pascal has a lot of inter-class confusion, more among the animal categories, than in the other object categories (Figure 5.2).

5.6 Summary

In this chapter, we presented simple, intuitive solutions to two fundamental questions, (a) *which visual words are important*, and (b) *which codebooks are important.* Our method relies on computing metrics that capture the relative populations of visual words in classes, thereby assessing which words are more likely to occur in which classes, which yields clues to aid class-discrimination. We describe strategies to (a) improve visual vocabularies by rejecting noisy words, (b) mix and match visual words drawn from different codebooks, and (c) identifying and rejecting codebooks with little discriminative information. We conduct experiments to analyse our method, describe a few applications, thereby producing empirical evidence that supports our hypothesis. This work opens up new avenues for exploration. The metrics we introduced are general, fairly simple to understand, and implement, and we think that further speculation, investigation, and design of comprehensive metrics that take into account more domain knowledge will lead us to better separation between classes in a discriminative task.

Chapter 6

Conclusions and Future Work

In this thesis we attempt to make the image representations better, in terms of information loss due to various quantization steps involved in the traditional bag of words. Reduction of information loss is achieved at the cost of computational and storage complexity. We address this problem too and proposed a novel solution for this as well. We compare our methods with the traditional approaches experimentally and produce evidence to demonstrate the superiority of our methods over the traditional approaches. The major contribution of this thesis are:

BOW: A soft clustering based exposition: In chapter3, we addressed the problem of information loss due to hard assignments in traditional bag of words approach and showed that this can be reduced by using soft assignments. We compare the soft assignment approaches (absolute and relative) with the traditional hard assignments in bag of words and demonstrated experimentally that both soft assignments are much better as compared to hard assignments by comparing the classification accuracy and mAP on various benchmark datasets. Also absolute or the possiblistic soft assignments are better than the relative or the probabilistic soft assignments.

Sparse Discriminative Fisher Vectors: In chapter 4, we first showed that Fisher vector image representations outperforms the other global representations on most benchmark datasets. However there is always a tradeoff between performance and resources. The representative power of Fisher vectors comes at a storage and computational cost. While most recent works in this direction have attempted to compress Fisher representations to save disk space, we have introduced a novel approach to make Fisher vector sparse which will reduce their explicit representation size. Further, we have introduced a novel strategy for making Fisher vectors discriminative by introducing class information in the representations. We demonstrated the superiority of our methods on four popular visual classification datasets and our method consistently out performed the baseline Fisher representations. This performance of our model is at par with the state of the art approaches on these datasets.

Discriminative Visual Words: In chapter 5, we presented simple, intuitive solutions to two fundamental problems, for a given class which codebooks are important and for a given codebook, which visual words are important. We did so by capturing the relative frequency of visual words for each code-

book for a given class and picking up the ones which are more likely to occur in that class by rejecting the noisy words. We produced empirical evidence on different datasets that support our hypothesis.

During the course of this research we worked on different things, tried various approaches, and finally devised few strategies of our own. However, there is always a scope of improvement in this research field. As there is still a lot to achieve before we can replicate the working of human brain perfectly, so the research will continue to achieve that goal of making machines work like humans.

6.1 Future Work

Our research in this area of enhancing various steps of bag of visual words model opened various directions which one can explore to further enhance BOW performance in various computer vision tasks. Our findings have shown that all the steps of BOW model can be enhanced separately.

Visual Feature Extraction though being the most stable step in BOW pipeline. Performance of a model depends on the quality of features extracted in this initial step. We have different types of features serving different use-cases. Researchers have shown that combination of features can give us an edge as compared to using them alone. But finding this combination is done empirically. Automating this step can be one direction from here. We can also explore the possibility of some new feature other than SIFT which as of now dominates the area of computer vision.

Performance of bag of words depends directly on the size of visual vocabulary which directly influences the complexity of the model. So basically to get higher performance we need to train more complex model which will depend on the resources we have. So another direction could be how to get same/better performance even with small vocabularies. Also we can explore some other ways to find an optimal vocabulary or a combination of vocabulary in a better way. We have given a starting point in that direction in chapther 5, but still we can explore a lot more in that field.

Quality of assignment/coding step is determined by the reconstruction error which we get when we try to reconstruct the original feature from its representative in the assignment step. Hard assignments have the highest reconstruction error and thus they perform the least. So we can explore some new methods which will reduce the reconstruction error but on the other hand won't increase requirements of our resources which are limited. This is the case with Fisher vector image representations. Though they reduce the reconstruction error of the visual features, but they do so with increased memory requirements and computational complexity. Our findings in chapter 4 demonstrate that this can be achieved. Though our technique is very simple and intuitive, one can be more fancy with that and take it as a starting point to move forward.

As a part of this thesis we have demonstrated that refining each step separately will improve the performance of BOW model, so basically one can try enhancing any step of this pipeline to enhance it further. Since these steps are independent of each other so enhancing all of them and then combining them together will enhance it multiple times.

Related Publications

1. **Bag of Visual Words: A Soft Clustering Based Exposition (Oral)**
   Vinay Garg, Sreekanth Vempati and C. V. Jawahar
   In Proceedings of the Third National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics, India, 2011

2. **Sparse Discriminative Fisher Vectors in Visual Classification (Oral)**
   Vinay Garg, Siddhartha Chandra and C. V. Jawahar
   Indian Conference on Vision, Graphics and Image Processing, 2012

Bibliography

[1] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

[2] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[3] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *CVPR*, 2008.

[4] A. Bosch, A. Zisserman, and X. Munoz. Scene classification using a hybrid generative/discriminative approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(4), 2008.

[5] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In *CVPR*, 2010.

[6] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, 2011.

[7] C. chung Chang and C.-J. Lin. Libsvm: a library for support vector machines, 2001.

[8] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV*, 2004.

[9] L. V. der Maaten. Learning discriminative fisher kernels. In *ICML*, 2011.

[10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[11] 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 *WGMBV*, 2004.

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

[13] S. Gao, I. W.-H. Tsang, and L.-T. Chia. Kernel sparse representation for image classification and face recognition. In *Proceedings of the 11th European conference on Computer vision: Part IV*, ECCV, 2010.

[14] K. Grauman and B. Leibe. Synthesis lecture on visual object recognition. In *Book: Visual Object Recognition*.

[15] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *NIPS*, 1998.

[16] H. Jegou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010.

[17] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local image descriptors into compact codes. *PAMI*, 2011.

[18] F. Jurie and B. Triggs. Creating efficient codebooks for visual recognition. In *ICCV*, 2005.

[19] J. Krapac, J. J. Verbeek, and F. Jurie. Modeling spatial layout with fisher vectors for image categorization. In *ICCV*, 2011.

[20] R. Krishnapuram and J. M. Keller. The possibilistic c-means algorithm: insights and recommendations. *Fuzzy Systems, IEEE Transactions on*, 4(3):385–393, 1996.

[21] D. Larlus and F. Jurie. Latent mixture vocabularies for object categorization. In *BMVC*, 2006.

[22] S. Lazebnik and M. Raginsky. Supervised learning of quantizer codebooks by information loss minimization. In *PAMI*, 2007.

[23] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *Int. J. Comput. Vision*, 43(1):29–44, June 2001.

[24] E. P. X. Li-Jia Li, Hao Su and L. Fei-Fei. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *NIPS*, 2010.

[25] J. Liu and M. Shah. Scene Modeling Using Co-Clustering. In *ICCV*, 2007.

[26] D. G. Lowe. Distinctive image features from scale-invariant keypoints. In *IJCV*, 2003.

[27] F. Moosmann, E. Nowak, and F. Jurie. Randomized clustering forests for image classification. *PAMI*, 2008.

[28] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

[29] M. Pandey and S. Lazebnik. Scene recognition and weakly supervised object localization with deformable part-based models. *CVPR*, 2011.

[30] F. Perronnin. Universal and adapted vocabularies for generic visual categorization. In *PAMI*, 2008.

[31] F. Perronnin, C. Dance, G. Csurka, and M. Bressan. Adapted vocabularies for generic visual categorization. In *In ECCV*, 2006.

[32] F. Perronnin and C. R. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007.

[33] F. Perronnin, Y. Liu, J. Sa andnchez, and H. Poirier. Large-scale image retrieval with compressed fisher vectors. In *CVPR*, 2010.

[34] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, 2010.

[35] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*, 2008.

[36] A. Quattoni and A. B. Torralba. Recognizing indoor scenes. In *CVPR*, 2009.

[37] P. Quelhas, F. Monay, J. M. Odobez, G. D. Perez, and T. Tuytelaars. A Thousand Words in a Scene. *PAMI*, 2007.

[38] J. Sanchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *CVPR*, 2011.

[39] C. Schmid. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[40] G. Sharma, F. Jurie, and C. Schmid. Discriminative Spatial Saliency for Image Classification. In *CVPR*, 2012.

[41] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.

[42] T. Tuytelaars and C. Schmid. Vector quantizing feature space with a regular lattice. In *ICCV*, 2007.

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

[44] J. C. van Gemert, J.-M. Geusebroek, C. J. Veenman, C. G. M. Snoek, and A. W. M. Smeulders. Robust scene categorization by learning image statistics in context. In *SLAM - CVPR*, 2006.

[45] J. C. van Gemert, C. J. Veenman, A. W. M. Smeulders, and J. M. Geusebroek. Visual word ambiguity. In *PAMI*, 2010.

[46] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008.

[47] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *CVPR*, 2010.

[48] A. Vedaldi and A. Zisserman. Sparse kernel approximations for efficient classification and detection. In *CVPR*, 2012.

[49] J. Vogel and B. Schiele. Natural scene retrieval based on a semantic modeling step. In *In CIVR*. Springer Verlag, 2004.

[50] J. Vogel and B. Schiele. Semantic modeling of natural scenes for content-based image retrieval. *IJCV*, 2007.

[51] J. Wang, J. Yang, K. Yu, F. Lv, T. S. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010.

[52] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *ICCV*, 2005.

[53] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. Huang, and S. Yan. Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 2010.

[54] J. Wu and J. M. Rehg. Centrist: A visual descriptor for scene categorization. *PAMI*, 2011.

[55] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. *CVPR*, 2010.

[56] J. Yang, K. Yu, Y. Gong, and T. S. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, pages 1794–1801. IEEE, 2009.

[57] L. Yang, R. Jin, C. Pantofaru, and R. Sukthankar. Discriminative cluster refinement: Improving object category recognition given limited training data. In *CVPR*, 2007.

[58] X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. In *ECCV*, 2010.