Bag of Words and Bag of Parts models for Scene Classification in Images

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

Master of Science (by Research)
in
Computer Science Engineering

by

Mayank Juneja
200702022
mayank.juneja@research.iiit.ac.in

Center For Visual Information Technology
International Institute of Information Technology
Hyderabad - 500 032, INDIA
April 2013

-

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Bag of Words and Bag of Parts models for Scene Classification in Images" by Mayank Juneja, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Prof. C.V. Jawahar

_____ _____

Date Adviser: Prof. A.P. Zisserman

To my Family

Acknowledgments

I would like to first thank my advisers Dr. Andrea Vedaldi, Prof. C.V. Jawahar and Prof Andrew Zisserman for all the guidance and support. I got to learn some unique things from each one of them. I can summarize these as *"art of implementing ideas into efficient and working codes"* from Dr. Andrea, *"importance of every small details"* from Prof. Jawahar, and *"striving for excellence"* from Prof. Zisserman.

It was a great fun and learning experience to be a part of CVIT. I would like to thank all the friends at CVIT for the useful discussions - Yashaswi, Abhinav, Vinay, Siddhartha, Srijan, Ankit, Chandrashekhar, Jay, Shrikant, Siddharth Kherada, Harshit Agarwal, Mihir, Harshit Sureka, Parikshit, Rohit Nigam, Aditya, Rohit Gautam, Shashank Mujjumdar, Anand, Nagendar, Nataraj, Vijay, Omkar. I would also like to thank friends at IIIT - Ashish, Gautam, Manan, Jaspal who made the college life a memorable one.

A special thanks to Prof. P. J. Narayanan, Prof. Jayanthi Sivaswamy and Prof. Anoop Namboodiri for creating a wonderful research atmosphere at CVIT.

I am grateful to Mr. R. S. Satyanarayana for all the administrative support. I would also like to thank Phani, Rajan and Nandini for helping with the annotations tasks.

Most importantly, I would like to thank my family for supporting me always, especially my mother and father who motivated me to stay focused and carry on with my research work, and my younger brother Udit who always questioned my work and made me think deeply about my work in order to answer him.

Abstract

Scene Classification has been an active area of research in Computer Vision. The goal of scene classification is to classify an unseen image into one of the scene categories, *e.g*. beach, cityscape, auditorium, etc. Indoor scene classification in particular is a challenging problem because of the large variations in the viewpoint and high clutter in the scenes. The examples of indoor scene categories are corridor, airport, kitchen, etc. The standard classification models generally do not work well for indoor scene categories. The main difficulty is that while some indoor scenes (*e.g*. corridors) can be well characterized by global spatial properties, others (*e.g*. bookstores) are better characterized by the objects they contain. The problem requires a model that can use a combination of both the local and global information in the images.

Motivated by the recent success of the Bag of Words model, we apply the model specifically for the problem of Indoor Scene Classification. Our well-designed Bag of Words pipeline achieves the state-of-the-art results on the MIT 67 indoor scene dataset, beating all the previous results. Our Bag of Words model uses the best options for every step of the pipeline. We also look at a new method for partitioning of images into spatial cells, which can be used as an extension to the standard Spatial Pyramid Technique (SPM). The new partitioning is designed for scene classification tasks, where a non-uniform partitioning based on the different regions is more useful than the uniform partitioning.

We also propose a new image representation which takes into account the discriminative parts from the scenes, and represents an image using these parts. The new representation, called Bag of Parts can discover parts automatically and with very little supervision. We show that the Bag of Parts representation is able to capture the discriminative parts/objects from the scenes, and achieves good classification results on the MIT 67 indoor scene dataset. Apart from getting good classification results, these blocks correspond to semantically meaningful parts/objects. This mid-level representation is more understandable compared to the other low-level representations (*e.g*. SIFT) and can be used for various other Computer Vision tasks too.

Finally, we show that the Bag of Parts representation is complementary to the Bag of Words representation and combining the two gives an additional boost to the classification performance. The combined representation establishes a new state-of-the-art benchmark on the MIT 67 indoor scene dataset. Our results outperform the previous state-of-the-art results [58] by 14%, from 49.40% to 63.10%.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The amount of visual data has increased exponentially over the past few years. Major factors for the rise being the advent of social networks (Facebook [16]), online photo/video sharing services (Flickr [23], Youtube [74], Instagram [28]), increase in number of smart-phones, etc. With this tremendous increase in the visual data, there is a need to extract and associate semantic information with the data. The information can be of various types, for example assigning meaningful labels to images, locating objects, recognizing human activities, detecting faces, etc. One of these tasks is recognising the scenes from an image, i.e. given an image, a computer should be able to tell what scene is depicted in the image. It could be an outdoor scene (forest, cityscape), or an indoor scene (airport, kitchen). From a human point of view, the problem looks easy, because of the complicated neural mechanisms of perception taking place in the human brain. However for a computer, an image is just a set of pixels without any high level semantic knowledge about what is present in the image, which scene the image depicts. To solve this problem, Computer Vision and Machine Learning algorithms come into picture.

1.1 Problem Statement and Contributions

The objective of this thesis is to solve the problem of indoor scene classification. Given a predefined set of scene categories, we would like to classify any unseen image into one of the scene categories. For example, given the images in Figure 1.1, our algorithm should be able to say that the images are from *Dining Room*, *Bookstore* and *Corridor* respectively. To evaluate our methods presented in this work, we use the MIT 67 indoor scene dataset of Quattoni and Torralba [49]. We make two major contributions in our work. First, we investigate the popular Bag of Words pipeline [59] for the scene classification task. We design it specifically for scene-classification by choosing the best options at each step of the pipeline. Our pipeline is able to achieve state-of-the-art results for Indoor Scene Classification. But, we feel that the bag of words is not able to capture the distinctive objects and structures. These objects are semantically meaningful and help in building a better scene recognition pipeline. For example, an auditorium image consists of chairs, bookstore consists of books, grocery store consists of food items. Motivated by this observation, we propose a new model which discovers the distinctive parts in the dif-

ferent scenes and use them to represent the images. We call it as the Bag of Parts model. This is second contribution of our work. We show that the Bag of Parts representation indeed captures semantically meaningful and distinctive elements from different scenes, and achieves good classification results. We also show that this representation is complementary to the Bag of Words representation, and combining the two representations gives an additional boost in the classification performance. The combined representation establishes a new state-of-the-art benchmark on the MIT 67 indoor scene dataset.

(a) Dining Room (b) Bookstore (c) Corridor

Figure 1.1: Example images showing three different scene types: (a) Dining Room, (b) Bookstore, (c) Corridor.

1.2 Challenges

We discuss some of the challenges faced while solving the Scene Recognition problem. Some of these challenges are common to other computer vision tasks also.

- **Viewpoint changes**

  Images of the same scene taken from different viewpoints can change the appearance of the scene. These kind of changes can be induced because of the different position of the camera while taking pictures. Figure 1.2 shows images of auditorium taken from different viewpoints.

- **Illumination**

  Illumination variation is one of the common problems which can cause large variations in the intensity values of the pixels, which makes the recognition task difficult. The major cause for illumination variation is different lighting conditions while taking the pictures.

Figure 1.2: **Viewpoint Variation.** Images of auditorium from different viewpoints. The example shows that the viewpoint can cause large variation in the appearance of the scene.

- **Inter-class similarity and Intra-class variation**

  The images belonging to different scene categories can look quite similar to each other (Inter-class similarity). On the other hand, images belonging to the same category can have a lot of variations (Intra-class variation). Both of these can cause confusions while predicting the scene for an image, thus making the problem harder to solve. Figure 1.3 shows an example of Inter-class similarity between three categories: concert-hall, movie theater and auditorium.

(a) Concert Hall (b) Movie Theater (c) Auditorium

Figure 1.3: **Inter-class Similarity.** Images from Concert Hall, Movie Theater and Auditorium. The example shows the similarity in the images from different scene types.

- **Semantic Overlap between classes**

  In some cases, the images are captured in such a way that it is very hard for even for a human to predict the scene. They can be classified as more than one scene category. Figure 1.4 shows some example images from different classes with semantic overlap. The example shows the difficulty in predicting the correct scene type.

(a) (b) (c)

Figure 1.4: **Semantic Overlap.** Images from different classes with semantic overlap. The example shows the difficulty in predicting the correct scene type. (a) can be classified as cloister or corridor, (b) can be classified as airport or waiting room, (c) can be classified as subway or train station.

1.3 Applications

Scene recognition has lots of applications. Some of them are listed below:

- **Robotics**

  Scene recognition is an essential task in robotics. For autonomous mobile robot navigation, it is important that the robot knows about its location by recognising the scene in front of it, and then make a decision accordingly.

- **Semantic Search**

  As the computers become more intelligent, there is a need for more semantic search capabilities. For example, a Google Image search query could be "Show me images of the airports", a Facebook search query could be "Give me images of my friends who were in a movie theater last weekend". Such queries can not be solved just by Computer Vision, but it is a very crucial component in the whole pipeline.

- **Photography**

  A professional photographer adjusts various settings (focus, exposure, etc.) in a camera before taking a picture of a scene. A scene recognition system can be used in cameras where the camera can make all the adjustments automatically based on the type of the scene recognised. Some camera manufacturers (*e.g.* Nikon) have already included this kind of system in their cameras.

- **Indexing/Labeling**

  As the amount of visual data increases, the recognition algorithms can be used to tag/label the images with semantic labels and then use the labels to do indexing. The labeling and indexing

can be used to archive the data efficiently, mine useful information from the data, and much more. As the visual data is exploding, efficient indexing has become necessary.

1.4 Thesis Outline

In Chapter 2, we review the existing methods for Scene Classification and explain the dataset used for evaluating the algorithms. We also explain the technical details of the basics of Computer Vision and Machine Learning, *e.g*. Feature representations, Support Vector Machines. We also explain the Bag of Words pipeline for image classification and show our results on the TRECVID 2010 Semantic Indexing Challenge.

In Chapter 3, we present the Bag of Words pipeline designed for Scene Classification. We present the results with the improved pipeline on MIT Scene 67 dataset. We also discuss an idea about a more meaningful partitioning in images, suited for scene classification.

In Chapter 4, we introduce a new and more semantically meaningful representation for images, called Bag of Parts representation. We present the results obtained by the Bag of Parts representation and the combined Bag of Words and Bag of Parts representations on MIT Scene 67 dataset.

Chapter 5 summarizes the contributions of our work.

Chapter 2

Background

In this chapter, we do a survey of the existing methods for scene classification in the literature. We present background knowledge related to image classification. This will be helpful in understanding the technical details presented in the next chapters. We also explain the standard Bag of Words pipeline, and present our results on the TRECVID 2010 Semantic Indexing Challenge.

2.1 Literature Survey

Scene classification is a popular task that has been approached from a variety of different angles in the literature. Several works [37, 44, 53, 54, 60, 62, 63] have focused on the problem of Indoor v/s Outdoor image classification. Apart from solving just the problem of Indoor v/s Outdoor classification, there have been works on classifying images into different scene categories. Oliva and Torralba [39] introduced a compact and robust global descriptor (GIST), which tries to represent the dominant spatial structure of a scene (in terms of perceptual dimensions: naturalness, openness, roughness, expansion, ruggedness). These dimensions can be reliably estimated using spectral and coarsely localized information.

Indoor scene recognition has also received a lot of attention, because the models designed for Indoor v/s Outdoor classification generally do not work well for indoor scene categories. The main difficulty is that while some indoor scenes (*e.g*. corridors) can be well characterized by global spatial properties, others (*e.g*. bookstores) are better characterized by the objects they contain. The problem requires a model that can use both the local and global information in the images. Quattoni and Torralba [49] study the problem of modeling scenes layout by a number of prototypes, which capture characteristic arrangements of scene components. Each prototype is defined by multiple ROIs which are allowed to move on a small window, and their displacements are independent of each other. The ROIs are obtained by human annotator, and a classification model is learnt on top of the ROIs. Wu and Rehg [72] propose a novel holistic image descriptor, CENTRIST (CENsus TRansform hISTogram) for scene classification. CENTRIST encodes the structural properties in images and contains rough geometrical information in the scene, while suppressing the textural details. Liu and Shah [35] model the scenes in terms of semantic concepts. The clusters of semantic concepts (intermediate concepts) are discovered using

Maximization of Mutual Information (MMI) co-clustering approach. Zhu *et al*. [75] explore the problem of jointly modeling the interaction of objects and scene topics in an upstream model, where topics are semantically meaningful and observed. This is done as a joint max-margin and max-likelihood estimation method. The popular method for Object detection, Deformable Part Models (DPMs) [19]

Who	**When**	**How**	**Performance**
Pandey and Lazebnik [41]	ICCV 2011	The method uses the standard DPM's in combination with global image features.	43.10
Wu and Rehg [72]	PAMI 2011	A new visual descriptor CENsus TRansform hISTogram (CENTRIST) is introduced, for recognizing topological places or scene categories. CENTRIST mainly encodes the structural properties within an image and suppresses detailed textural information.	36.90
Kwitt *et al*. [30]	ECCV 2012	The method uses a set of predefined themes, a classifier trained for the detection of each theme, and each image fed to all theme classiers. The image is finally represented by the vector of resulting classication labels.	44.00
Wang *et al*. [70]	CVPR 2012	The model incorporates covariance patterns from natural images into feature construction. Covariance is modeled as sparse linear combination of regular patterns, and combined with learning of linear features.	33.70
Parizi *et al*. [42]	CVPR 2012	A reconfigurable version of a spatial BoW model is proposed that associates different BoW descriptors that associates to different image segments. A scene is represented as a collection of region models ("parts") arranged in a reconfigurable pattern.	37.93
Singh *et al*. [58]	ECCV 2012	A mid-level visual representation is proposed in terms of discriminative patches.	49.40
Sadeghi and Tappen *et al*. [51]	ECCV 2012	Discriminative characteristics of the scenes are captured by a set of Latent Pyramidal Regions (LPR). Each LPR is represented in a spatial pyramid and uses non-linear locality constraint coding for learning both shape and texture patterns of the scene.	44.84

Table 2.1: List of recent papers (past couple of years) which have focused on scene classification problem.

has also been tried to solve the problem of scene categorisation. Standard DPM's are used by Pandey and Lazebnik [41] to capture recurring visual elements and salient objects in combination with global image features. The problem of part initialisation and learning is not addressed, and the quality of

the parts that can be obtained in this manner remains unclear. Li *et al*. [34] try to solve the problem by using explicit object detectors. They use a set of pre-trained object detectors and apply the set of detectors to get an object filter bank representation of images. It requires hundreds and thousands of object detectors, training of which could be expensive. Sadeghi and Tappen [51] address this problem by capturing discriminative characteristics of the scenes by a set of Latent Pyramidal Regions (LPR). Each LPR is represented in a spatial pyramid and uses non-linear locality constraint coding for learning both shape and texture patterns of the scene. Parizi *et al*. [42] propose a reconfigurable version of a spatial BoW model that associates different BoW descriptors to different image segments, corresponding to different types of "stuff". A scene is represented as a collection of region models ("parts") arranged in a reconfigurable pattern. The assignment of region model to image regions is specified by a latent variable. The models can be learned using both generative and discriminative methods. Singh *et al*. [58] explores learning parts in both an unsupervised and weakly supervised manner, where the weakly supervised case uses only the class label of the image. The part-learning procedure described in their work is expensive.

To summarize, the scene classification methods can be categorized into three major groups:

(i) Methods which use low-level image descriptors (*e.g*. [42, 72]).

(ii) Methods which use mid-level representation like patches, object banks (*e.g*. [34, 41, 58]).

(iii) Methods which use complicated mathematical models (*e.g*. [75]).

In the past couple of years, there have been plenty of papers which have focused on scene classification problem. We list down the recent methods in Table 2.1, along-with their performance on the MIT Scene 67 Indoor dataset. As we discuss in the next chapters, our method beats all the existing results on this dataset.

2.2 Dataset and Evaluation Measures

2.2.1 Dataset

Image datasets are an essential component of computer vision research. As humans are trained from their knowledge and experience, similarly datasets are required to train the computers using different algorithms. These datasets also allow to compare and benchmark performance of the different algorithms. The datasets can be classified into different categories, based on the problem they are designed to tackle. Some of the popular publicly available datasets are **Caltech 101, Caltech 256, CIFAR-100, ImageNet** (Classification), **PASCAL VOC** (Classification, Detection, Segmentation), **INRIA Persons** (Detection), **UCF Sports Action Dataset** (Action Recognition). The problem of Scene Classification can be solved as a sub-problem of Image Classification. But the classification datasets like Caltech-256, PASCAL VOC are more object-centric, most of the classes contain a primary object (*e.g*. Car, Bike, Cat, Person, Bottle, TV, etc.).

Figure 2.1: **MIT 67 Indoor data set** Summary of the 67 indoor scene categories, organized into 5 big scene groups. (Figure source : [49])

The image classification datasets are not designed for the problem of scene recognition, which is a challenging open problem in high level vision. Quattoni and Torralba [49] introduced a new dataset, MIT 67 indoor scene dataset designed specifically for indoor scenes. It is the largest available dataset for indoor scenes, before this most of the work on scene recognition focused on a reduced set of indoor and outdoor categories. The MIT data comprises 67 indoor scene categories loosely divided into **stores** (*e.g*. bakery, toy store), **home** (*e.g*. bedroom, kitchen), **public spaces** (*e.g*. library, subway), **leisure** (*e.g*. restaurant, concert hall) and **work** (*e.g*. hospital, TV studio).

The images in the dataset were collected from different sources: online image search tools (Google and Altavista), online photo sharing sites (Flickr) and the LabelMe dataset. Figure 2.1 shows the 67 scene categories used in this study. The database contains 15620 images. A subset of the dataset that has approximately the same number of training and testing samples per class is used for the experiments. The partition that we use is:

Train Set: contains 67*80 images, total of 5360 images

Test Set : contains 67*20 images, total of 1340 images.

All images have a minimum resolution of 200 pixels in the smallest axis.

Apart from the MIT 67 indoor scene dataset, there are other datasets which have been used for evaluating scene classification methods. A couple of them are Scene-15 [18, 31] and the SUN dataset [73]. Scene-15 dataset is not a challenging dataset because of small size and limited number of categories. Recent methods [24] have shown classification accuracies of upto 90% on this dataset. On the other hand, SUN dataset has 899 categories. The database consists of both outdoor and indoor scenes and there is a lot of variation in the amount of training data available for each class, making it difficult to evaluate the methods. The recent scene classification methods have used the MIT 67 indoor scene dataset to evaluate the performance. In this thesis, we use MIT 67 indoor scene dataset for evaluating our methods.

2.2.2 Evaluation Measures

In this section, we explain the evaluation measures which are used to measure the performance of the methods described in this thesis. The performance of the classification methods is evaluated on the Test Images. We use two popular evaluation measures:

(i) Mean Average Precision (MAP)

(ii) Classification Accuracy

2.2.2.1 Mean Average Precision (MAP)

Mean Average Precision is an evaluation measure commonly used for evaluating ranked lists in Information Retrieval. Suppose we are building a system to classify images of scene "auditorium". Let the system be evaluated on a Test Set T. We define four accuracy counts:

- **True Positives (t_p)** : Number of "auditorium" images classified as "auditorium".
- **False Positives (f_p)** : Number of "non-auditorium" images classified as "auditorium".
- **True Negatives (t_n)** : Number of "non-auditorium" images classified as "non-auditorium".
- **False Negatives (f_n)** : Number of "auditorium" images classified as "non-auditorium".

Now, Precision and Recall are defined as follows:

- **Precision** : Fraction of the the true positives in the retrieved results.

$$P = \frac{t_p}{t_p + f_p} \tag{2.1}$$

- **Recall** : Fraction of all the auditorium images.

$$R = \frac{t_p}{t_p + f_n} \tag{2.2}$$

Precision and Recall are used to characterize the performance of a classifier. A good classifier is good at ranking actual "auditorium" images near the top of the list, its precision stays high as recall increases. This can be observed by plotting Precision as a function of Recall (often called as PR curves). The performance of the classifier is measured by a single number, area under the PR curve, also called as Average Precision (AP). For a multi-class classification problem, we measure the performance by the mean of Average Precision (AP) values of the individual classifiers. Figure 2.2 shows an example of a PR curve for a classifier. The Average Precision (AP), measured as the Area under the Curve (AUC) is 85.20%.

2.2.2.2 Classification Accuracy

The performance of a classifier is also evaluated in terms of the correctness of labeling an image of the given class. Accuracy of a binary classifier is defined as the proportion of the total number of predictions that were correct. Following the notations from the previous section, the accuracy can be defined as:

$$\text{Accuracy } = \frac{t_p + t_n}{t_p + f_p + t_n + f_n} \tag{2.3}$$

For a multi-class classification problem, the class of an image is predicted by the classifier which assigns the maximum score to that image. A confusion matrix (Table 2.2) is constructed, where each column of the matrix represents the images in a predicted class, and each row represents the instances in an actual class. The final accuracy of the system is calculated as the mean of the diagonal of the confusion matrix.

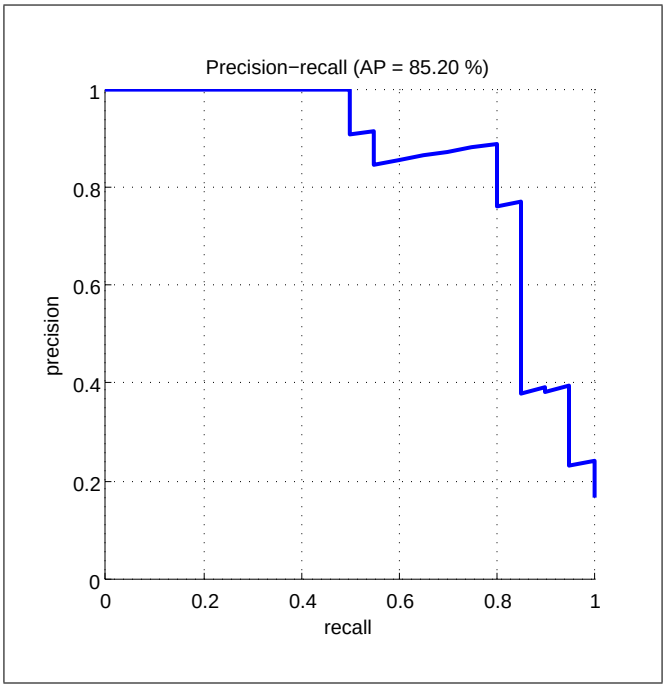


Figure 2.2: **Precision Recall (PR) Curve.** Example of a Precision Recall curve of a classifier with Average Precision (AP)=85.20%.

2.3 Tools and Techniques

2.3.1 Image Descriptors

Image descriptors provide description of the visual content in images. Computer Vision algorithms require an appropriate representation of the images. The different image descriptors try to capture different characteristics in images such as shape, edges, color, texture, spatial structure. These descriptors can be classified into two categories: (a) Global descriptors which are computed on the entire image, and (b) Local descriptors which are computed locally in small regions or points of interest in an image. In this section, we describe two of the most commonly used image descriptors in the literature, SIFT and HOG. We also use these two as the image descriptors in our methods described in the thesis.

2.3.1.1 SIFT

Scale-Invariant Feature Transform (SIFT) proposed by Lowe [36] is one of the popular local feature detector and descriptor. SIFT descriptor is invariant to Affine transformation, Lighting changes, Noise. The original SIFT implementation includes both an interest point detector and feature descriptors at the interest points. The descriptor associates to the regions a signature which identifies their appear-

		Predicted Class		
		Class A	Class B	Class C
Actual Class	Class A	0.80	0.15	0.05
	Class B	0.10	0.75	0.15
	Class C	0.05	0.05	0.90

Table 2.2: **Confusion Matrix.** Example of a confusion matrix for a 3-class classification problem.

ance compactly and robustly. Figure 2.3 shows an example of SIFT descriptor computation at some keypoints, and how they can be used to match points in different images.

Computing SIFT descriptor at a point starts with sampling the image gradient magnitudes and orientations in a region around the point. The samples are then accumulated into orientation histograms (bin size = 8), summarizing the contents over 4×4 subregions. These orientation histograms capture the local shape. The final descriptor is formed from a vector containing the values of 4×4 array of orientation histogram around the point. This leads to a SIFT feature vector with $4 \times 4 \times 8 = 128$ elements. To obtain illumination invariance, the feature vector is normalized by the square root of the sum of squared components.

2.3.1.2 HOG

Histogram of Gradients (HOG) proposed by Dalal & Triggs [13] is a feature descriptor originally used for Pedestrian Detection [13] and now popularly used for object detection [21]. The idea behind the HOG descriptors is that the local object appearance and shape can be characterized within an image by the distribution of local intensity gradients or edge directions.

The HOG descriptor is usually computed over a window in an image. The image window is first divided into small spatial regions, also called as "cells" (usually the size of a cell is 8×8 pixels). Then, the gradient directions or edge orientations are computed over the pixels of the cell. The gradients of each cell are accumulated into a 1-D histogram. The combined histogram entries form the HOG representation of the image window. For better invariance to illumination, shadowing, etc., it is also useful to contrast-normalize the local responses before using them. This is done by accumulating a measure of local histogram "energy" over somewhat larger spatial regions ("blocks") and using the results to normalize all of the cells in the block. Figure 2.4 summarizes the pipeline for computing the HOG descriptors.

2.3.2 k-means Clustering

k-means clustering is an unsupervised clustering algorithm, commonly used for constructing vocabularies for Bag of Words model. Given a set of $\mathbf{n}$ data points and number of clusters $\mathbf{K}$, k-means

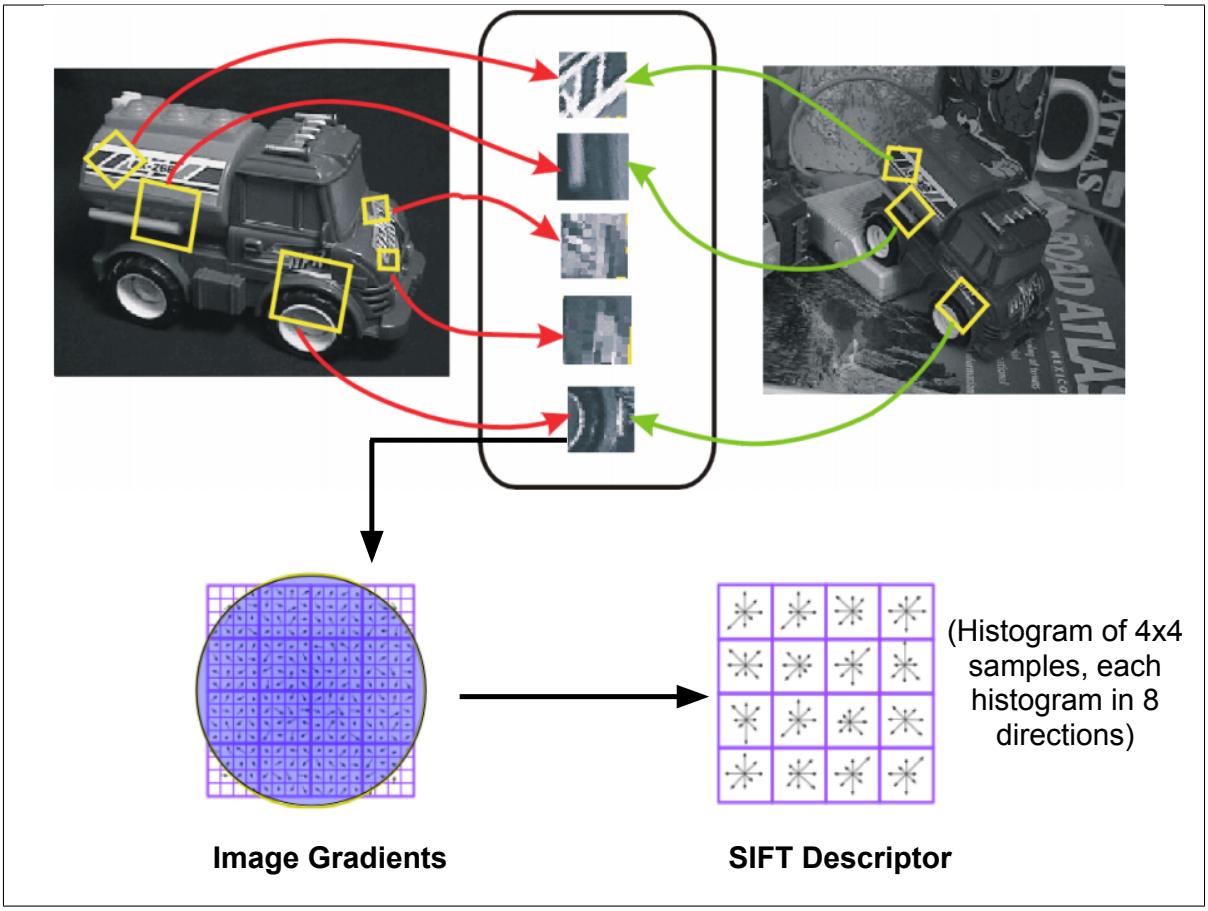


Figure 2.3: **SIFT Descriptors.** Example showing computation of SIFT descriptor in images. (Figure source : [32].)

partitions the data points into $\mathbf{K}$ clusters, such that each data point belongs to the cluster with the nearest mean. Figure 2.5 shows an example with some data points and the clusters formed with 3 clusters.

Let the $\mathbf{n}$ data points be $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$, where $\mathbf{x}_i \in \mathbb{R}^D$, and $\mathbf{K}$ be the number of clusters ($\mathbf{K} <= \mathbf{n}$). $\mathbf{S} = \mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_K$ be the set of clusters each consisting of some data points, μ_i be the mean of points in the set $\mathbf{S}_i$, $\mu_i = \frac{\sum_{\mathbf{x}_j \in \mathbf{S}_i} \mathbf{x}_j}{\sum_{\mathbf{x}_j \in \mathbf{S}_i} 1}$

k-means tries to minimise the following objective function:

$$\mathbf{S} = \arg\min_{\mathbf{S}} \sum_{i=1}^{\mathbf{K}} \sum_{\mathbf{x}_j \in \mathbf{S}_i} \|\mathbf{x}_j - \mu_i\|^2 \tag{2.4}$$

The k-means algorithm uses an iterative refinement technique to solve the optimization problem. The iterative procedure is also referred to as Lloyd's algorithm. The algorithm starts with randomly

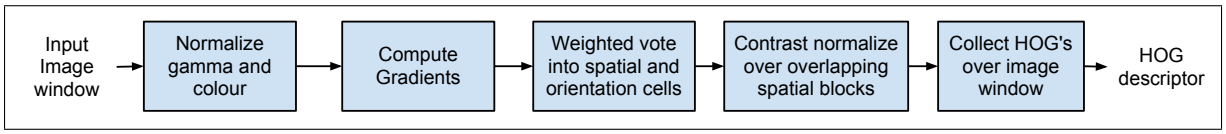
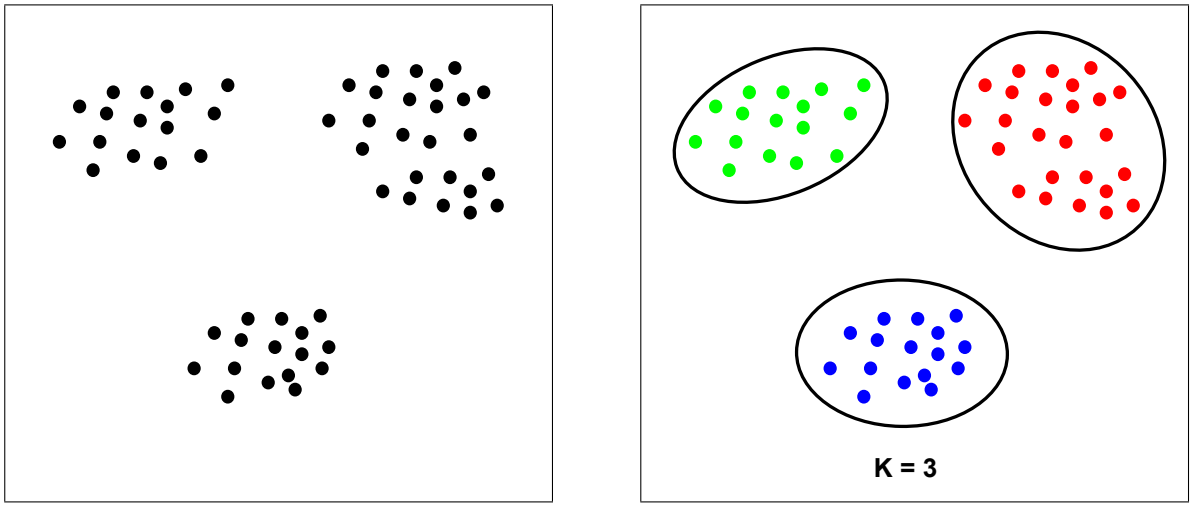


Figure 2.4: Overview of HOG descriptor extraction pipeline. (Figure source: [13]).



(a) Input data points

(b) Clusters

Figure 2.5: k**-means Clustering.** Example data points, and the clusters computed by k-means clustering.

initializing the means $\mu_1, \mu_2, \ldots, \mu_K$. The algorithm proceeds by alternating between the following two steps:

- **Assignment Step** During the assignment step, each data point is assigned to the cluster whose mean is closest to that data point.

$$\mathbf{S}_i = \{\mathbf{x}_p : \|\mathbf{x}_p - \mu_i\| \leq \|\mathbf{x}_p - \mu_j\| \quad \forall 1 \leq j \leq \mathbf{K}\} \tag{2.5}$$

- **Update Step** During the update step, the mean of each cluster is recomputed after the new assignments from the previous step.

$$\mu_i = \frac{\sum\limits_{\mathbf{x}_j \in \mathbf{S}_i} \mathbf{x}_j}{\sum\limits_{\mathbf{x}_j \in \mathbf{S}_i} 1} \tag{2.6}$$

The algorithm converges when the assignments of the data points do not change. The "assignment" step is also referred to as **expectation** step and the "update" step as **maximization** step, making the algorithm a variant of the Expectation-Maximization algorithm.

There is no guarantee that the algorithm will converge to the global optimum, and the result depends on the initialisation of the cluster means. One common practice is to randomly chose $\mathbf{K}$ points from the data points as the initial cluster means, and run it multiple times with different initialisations. There are other variants of initialisations in the literature, for example k-means++[5], which avoids the poor clusterings found by the standard k-means algorithm..

The only parameter involved in k-means clustering is $\mathbf{K}$, the number of clusters. The value usually depends on the nature of data, and should be chosen appropriately by experiments.

2.3.3 Classification using Support Vector Machines (SVM)

Classification is one of the most common task in machine learning. A supervised learning system that performs classification is known as a learner or, more commonly, a classifier. Formally, given a set of input items, $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ and a set of labels/classes, $\mathbf{Y} = \{y_1, y_2, \ldots, y_n\}$ and training data $\mathbf{T} = \{(\mathbf{x}_i, y_i) | y_i \text{ is the label/class for } \mathbf{x}_i\}$, a classifier is a mapping from $\mathbf{X}$ to $\mathbf{Y}$, $f(\mathbf{T}, \mathbf{x}) = y$. There are multiple algorithms for solving the classification problem, *e.g.* Bayes classifier, k-nearest neighbor, Neural Networks, Random Forests, Support Vector Machines, etc. In this thesis, we use Support Vector Machines (SVM) for classification, which we explain in the following paragraphs.

Support Vector Machine [8, 11] is a popular and powerful classification learning tool. It is a supervised learning method, i.e. it learns from a given set of labelled training data and predicts the label for an unseen test sample. We will explain SVMs for two-class case, which is also called as a "binary" classifier. The basic idea behind a linear classifier is to separate the given D-dimensional data points with a $(D-1)$ dimensional hyperplane. For a given set of points, there may exist multiple hyperplanes which can separate the data (Figure 2.6(a)). The best classifier of all these hyperplanes is the one which provides the maximum separation of the data points (Figure 2.6(b)). It essentially means that the best hyperplane should maximise the distance between the nearest points on both sides of the hyperplane (nearest points to the hyperplane from each class). This distance is defined as the "margin", and the SVM selects the hyperplane with the maximum margin. The hyperplane obtained is called **maximum-margin hyperplane** and the linear classifier is called **maximum-margin classifier**.

Given a set of $\mathbf{n}$ labelled training samples,

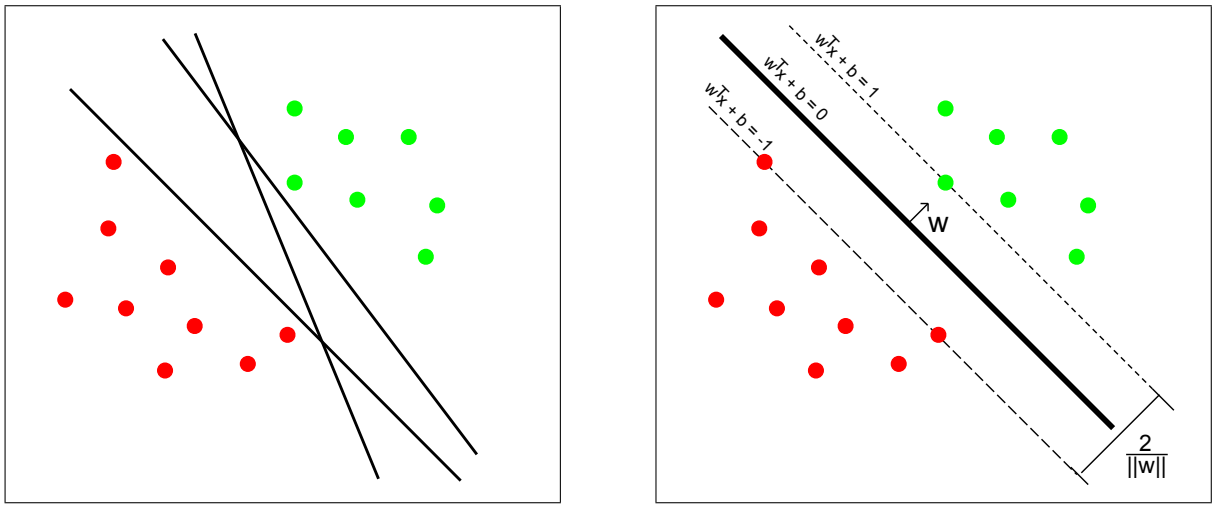
$$\mathbf{S} = \{\{\mathbf{x}_i; y_i\} | \quad \mathbf{x}_i \in \Re^D, y_i \in \{-1, 1\}\}_{i=1}^n$$

$\mathbf{x}_i$ is the D-dimensional data point, y_i represents the class to which the point $\mathbf{x}_i$ belongs.

A separating hyperplane with $\mathbf{w}$ as the normal vector, can be written as

$$\mathbf{w^T x} + b = 0$$

Here, b is called the bias term, $\frac{b}{\|\mathbf{w}\|}$ gives the perpendicular distance from the origin to the hyperplane. Our goal is to find $\mathbf{w}$ and b, such that the "margin" is maximised. We can select two parallel hyperplanes which separate the data and are as far as possible (Figure 2.6). These hyperplanes can be written as follows:



(a) Multiple separating hyperplanes

(b) Max-Margin Hyperplane

Figure 2.6: Example showing multiple separating hyperplanes and the max-margin hyperplane output by SVM.

$$\mathbf{w^T x} + b = 1$$
$$\mathbf{w^T x} + b = -1$$

Now, the distance between the two parallel hyperplanes is $\frac{2}{\|\mathbf{w}\|}$. Since the distance needs to be maximised, it translates to minimizing $\|\mathbf{w}\|$. Since we do not want any data points falling in between the two parallel hyperplanes, the following constraints are added:

$$\mathbf{w^T x}_i + b \geq 1 \quad \forall\, \mathbf{x}_i \text{ s.t. } y_i = 1$$
$$\mathbf{w^T x}_i + b \leq -1 \quad \forall\, \mathbf{x}_i \text{ s.t. } y_i = -1$$

The two constraints can be combined and rewritten as:

$$y_i(\mathbf{w^T x}_i + b) \geq 1 \quad \forall\, \mathbf{x}_i$$

We can substitute $\|\mathbf{w}\|$ with $\frac{1}{2}\|\mathbf{w}\|^2$, without changing the solution, this makes the optimization problem easier to solve. The optimization problem can now be written in primal form as:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \tag{2.7}$$

$$\text{subject to} \quad y_i(\mathbf{w^T x}_i + b) \geq 1 \quad \forall \mathbf{x}_i$$

The optimization problem of SVM described above can be solved by different algorithms. In our work, we use PEGASOS: Primal Estimated sub-GrAdient SOlver for SVM algorithm [55], which is

described here. It is an iterative, gradient descent based algorithm which solves the SVM problem in primal form. Given a training set $\mathbf{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$, where $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \{+1, -1\}$, the SVM problem (Equation 2.7) can be rewritten as the following minimization problem:

$$\min_{\mathbf{w}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{m}\sum_{(\mathbf{x},y)\in S} l(\mathbf{w}; (\mathbf{x}, y)), \tag{2.8}$$

where

$$l(\mathbf{w}; (\mathbf{x}, y)) = \max\{0, 1 - y\langle \mathbf{w}, \mathbf{x}\rangle\}. \tag{2.9}$$

The bias term, b from the original problem is omitted here. In PEGASOS, bias can be added as an additional element in the original data points $\mathbf{x}_i$'s as $[\mathbf{x}_i\ ;\ b]$.

The input parameters to the Pegasos algorithm are:

(i) T: the number of iterations to be performed

(ii) k: the number of samples to use for calculating sub-gradients

The complete algorithm is given below:

Algorithm 1: Pegasos Algorithm.

Input: S, λ, T, k
Initialize: Choose $\mathbf{w}_1$ s.t. $\|\mathbf{w}_1\| \leq 1/\sqrt{\lambda}$
for $t = 1, 2, \ldots, T$ **do**
  Choose $A_t \subseteq S$, where $|A_t| = k$
  Set $A_t^+ = \{(\mathbf{x}, y) \in A_t : y\langle \mathbf{w}_t, x\rangle < 1\}$
  Set $\eta_t = \frac{1}{\lambda t}$
  Set $\mathbf{w}_{t+\frac{1}{2}} = (1 - \eta_t\lambda)\mathbf{w}_t + \frac{\eta_t}{k}\sum_{(\mathbf{x},y)\in A_t^+} y\,\mathbf{x}$
  Set $\mathbf{w}_{t+1} = \min\left\{1, \frac{1/\sqrt{\lambda}}{\|\mathbf{w}_{t+\frac{1}{2}}\|}\right\}$
end
Output: w_{T+1}

2.4 Bag of Words Method

The bag of words model in computer vision is inspired from the bag of words model in the text domain [52] where a document is represented as an unordered collection of words. The order of words has no significance, it is the frequencies of the words that matters, for example the phrase "*classification problem in images*" has the same representation as the phrase "*in images classification problem*". The bag of words model in computer vision got popular with Video Google [59], and is often referred to as bag of visual words model. The idea has also been used in the texture recognition [33] where different

textures can be characterized by the repetition of basic elements, called as textons [29]. The typical Bag of Words (BoW) pipeline (Figure 2.7) is composed of the following steps:

1. Extracting local image features (*e.g*. SIFT, HOG),
2. Generating vocabulary of visual words (*e.g*. by k-means clustering),
3. Encoding and Generating spatial histograms, and
4. Learning a classifier based on the image descriptors (*e.g*. SVM).

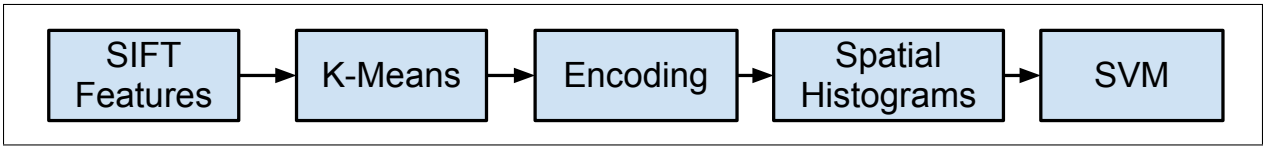


Figure 2.7: Different steps in the Bag of Words pipeline.

2.4.1 Extracting local features

The first step in the bag of words pipeline is computing the local features, which capture the local characteristics of an image. Any feature descriptor can be selected, the popular ones being SIFT (Section 2.3.1.1) and HOG (Section 2.3.1.2). In our work, we use SIFT descriptors. SIFT descriptors are extracted at a dense grid of uniformly-spaced points. At each point on the dense grid, the SIFT descriptors are computed over multiple circular support patches [6, 7]. The multiple descriptors are computed at each point to allow for scale variation between images. The SIFT descriptors below a certain contrast threshold are mapped to null vectors. Figure 2.8 shows the computation of the SIFT descriptors in an image at a dense grid, and at multiple scales at each point.

2.4.2 Codebook generation

After computing the local descriptors, the next step in the pipeline is to cluster the local descriptors into a codebook of visual words. The alternatives to the term "codebook" are "vocabulary" or "dictionary" of visual words, which is analogous to the dictionary of words in text domain. The idea behind a codebook is that an image can be represented in terms of these visual words. k-means clustering (Section 2.3.2) is a popular method to construct a vocabulary of visual words. A set of random descriptors from a subset of the Training set images is used to construct the visual vocabulary. The number of visual words is a parameter that depends on the dataset, and is generally determined experimentally.

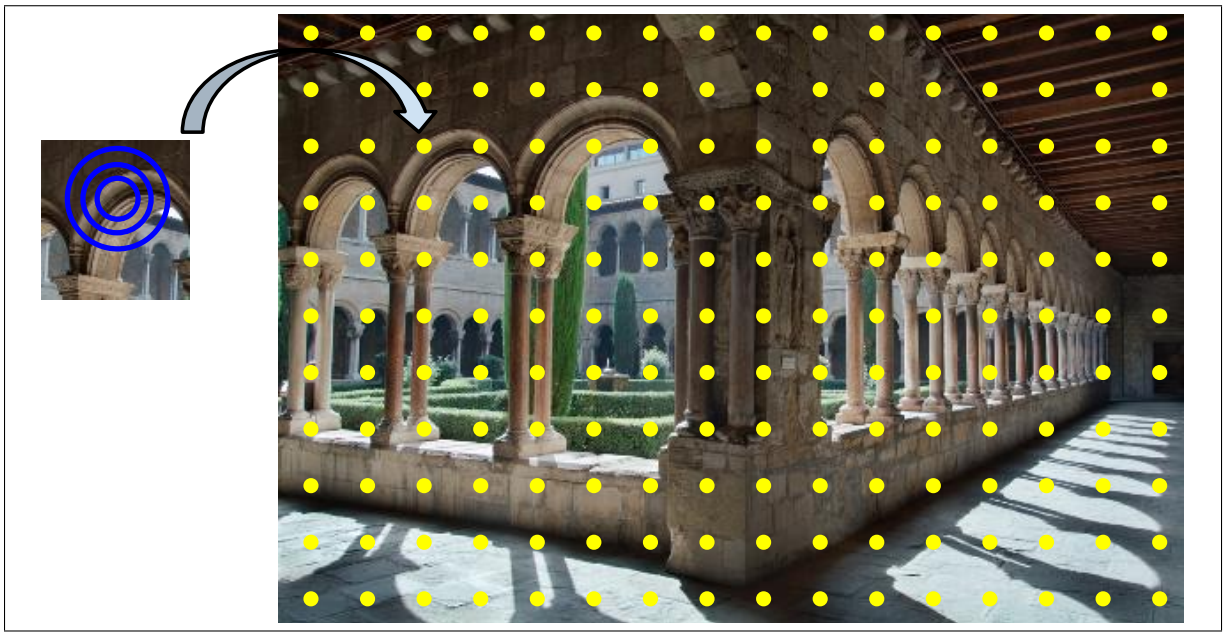


Figure 2.8: Example of SIFT descriptors extracted at a dense grid of points, and at multiple scales at each point.

2.4.3 Spatial Histograms

After codebook generation, an image is then represented as a bag of visual-words. Each local descriptor $\mathbf{x}_i$ is encoded by the nearest visual word in the Euclidean space (Equation 2.10).

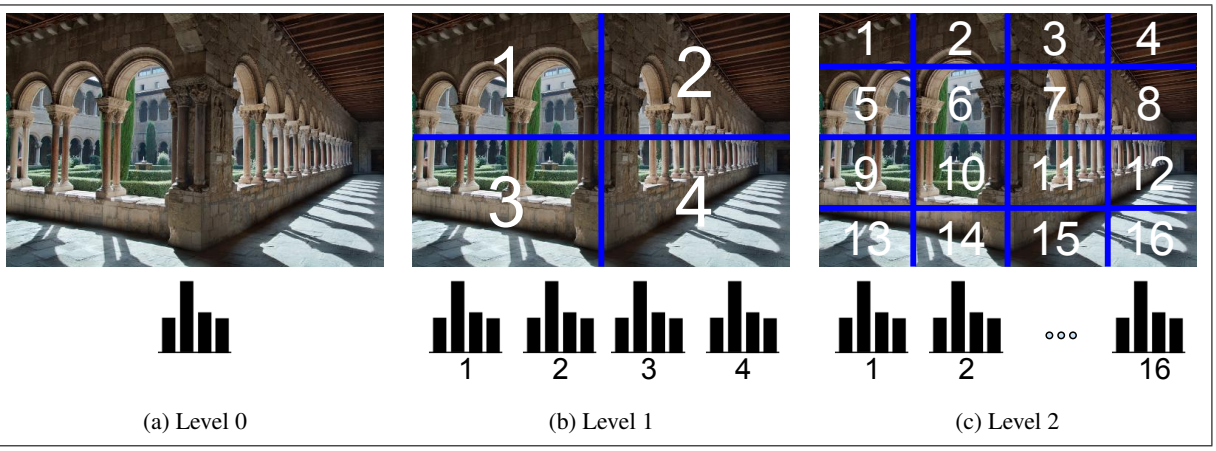
$$c_i = \arg\min_k \|\mathbf{x}_i - \mu_k\|^2 \tag{2.10}$$

After getting the encodings for all the local descriptors in an image, it is described by a vector (or histogram) that stores the distribution of all the visual words. The size of the histogram is equal to the vocabulary size, where each bin corresponds to a visual word.

Bag of Words representation does not capture any spatial information. Spatial information is introduced by the use of spatial pyramid [31]. An image is tiled into regions at multiple resolutions. An encoding is then computed for each spatial region at each resolution level (Figure 2.9). The final descriptor of an image is a concatenation of the encodings of different spatial regions into a single vector. Note that the histograms of each region are individually normalized before concatenation. The distance between the two vectors reflects the extent to which the images contain similar appearance and the extent to which the appearances correspond in their spatial layout.

Figure 2.9 shows an example of Spatial Pyramids, the spatial regions are obtained by dividing the image in 1×1, 2×2, and 4×4 grids, for a total of 21 regions. The histogram of each of the regions

is K-dimensional, and the dimension of the complete, concatenated feature vector for an image is $(1 + 2 \times 2 + 4 \times 4)K = 21K$.



(a) Level 0 (b) Level 1 (c) Level 2

Figure 2.9: **Spatial Pyramid.** An image tiled into regions at multiple resolutions and histograms computed for each spatial cell.

2.4.4 Learning

The next step in the pipeline is learning. After getting the image descriptors, the aim is to learn models (classifiers) for the different classes. SVM (Section 2.3.3) is a commonly used classifier in BoW pipeline. For each class, a separate SVM is learnt which can predict whether an unseen image belongs to that particular class or not. For training an SVM for a particular class, the descriptors of the images belonging to that class act as positive data points for the SVM, and descriptors of rest of the images act as negative data points. The set of these positive and negative images is also referred to as the Training Set, while the set of unseen images whose classes are to be predicted, is referred to as the Test Set.

2.5 Bag of Words for TRECVID 2010

Our team participated in the "light" version of the semantic indexing task (SIN), TRECVID 2010 [40]. The goal of the challenge was to automatically identify the occurrence of various semantic features/concepts such as "Cityscape", "Airplane", "Nighttime" etc., which occur frequently in video information. The challenge had 2 versions - "full" with 130 concepts, and "light" with 10 concepts. The DEVELOPMENT data set (provided for training) and the TEST data set, each consisted of 200 hours of video data drawn from the IACC.1 collection using videos with durations between 10 seconds and 3.5 minutes.

We extract our own keyframes for every shot of both the TRECVID 2010 DEVEL and TEST data sets. The DEVEL set is subdivided into two halves denoted TRAIN and VAL and used for training and validation, respectively. This subdivision respects movie boundaries to guarantee the statistical independence of the keyframes in the two subsets. Ground truth labels for the DEVEL keyframes were obtained by an internal team of annotators (refer Section 2.5.1).

2.5.1 Annotations

Annotations were carried out (only for the DEVEL subset) at the frame level for each of the ten classes. For some of the selected object-like classes, Region of Interest (ROI) annotations were also carried out. After obtaining the first set of ground truth labels, multiple rounds of refinement were carried out to remove the errors in the annotation.

The refinement of annotations of the VAL set was carried out by using a weak classifier as follows :

1. Train a classifier on the TRAIN set.
2. Re-rank all the images in the VAL set based on the classifier output.
3. Refine the annotations of the top 1000 ranked frames and the bottom 1000 ranked frames. The high ranked mistakes in annotation are corrected in this way.

Similarly the refinement of annotations of the TRAIN set was done by using a classifier trained on the VAL set. Table 2.3 gives the statistics of the keyframes in DEVEL set for each category, along with the number of positives and negatives in each category.

2.5.2 Method and Implementation

We use Pyramid Histogram of Visual Words [7] to represent an image. The Pyramid Histogram of Visual Words (PHOW) descriptors consist of visual words which are computed on a dense grid. Here visual words are vector quantized SIFT descriptors [36] which capture the local spatial distribution of gradients.

Local appearance is captured by the visual words distribution. SIFT descriptors are computed at points on a regular grid with spacing M pixels. We have used gray level representations for each image. At each grid point, the descriptors are computed over circular support patches with radii r. Thus, each point is represented by four SIFT descriptors. These dense features are vector quantized into visual words using k-means clustering. Here, we have used a vocabulary of 1000 words. Each image is now represented by a histogram of these visual words occurrences. We have used M = 5, K = 1000 and radii r = 10; 15; 20; 25. To deal with the empty patches, we zero all the SIFT descriptors with L2 norm below a threshold (200). In order to capture the spatial layout representation, which is inspired by the pyramid representation of Lazebnik et. al. [31] , an image is tiled into regions at multiple resolutions. A histogram of visual words is then computed for each image sub-region at each resolution level. To

Category	TRAIN		VAL	
	# positives	# negatives	# positives	# negatives
Cityscape	1438	100596	958	86666
Singing	1384	101312	1372	86567
Nighttime	417	102561	329	87921
Boat Ship	261	102803	246	87995
Airplane Flying	150	103041	89	88185
Hand	110	99077	119	85807
Demonstration or Protest	109	102779	136	87833
Telephone	41	103127	67	88266
Bus	34	103245	25	88390
Classroom	31	103246	7	88381

Table 2.3: Statistics of keyframes in DEVEL set (TRAIN + VAL) for each category of TRECVID-2010 Semantic Indexing Challenge.

summarize, the representation of an appearance descriptor for an image is a concatenation of the histograms of different levels into a single vector which are referred to as Pyramid Histogram of Visual Words (PHOW). We have used two levels for the pyramid representation. The distance between the two PHOW descriptors reflects the extent to which the images contain similar appearance and the extent to which the appearances correspond in their spatial layout.

The image-level classifier is a non-linear SVM on top of a bag of dense visual words. To train a large-scale SVM efficiently, we use PEGASOS SVM [56] (as implemented in the VLFeat library [66]). While PEGASOS is a linear SVM solver, we use the explicit feature map for χ^2 kernel [68] to extend it efficiently to use a χ^2 (non-linear) kernel. The whole setup is fast and efficient compared to traditional SVM techniques that do not use the feature map idea. For example, on our framework training an SVM using 100K frames requires only 2 minutes and classifying 100K frames requires only 1 minute on an Intel Xeon CPU clocked at 1.86 GHz.

Category	AP	inf AP
Training Set **Testing Set**	**TRAIN** **VAL**	**TRAIN+VAL** **TEST**
Cityscape	0.49	0.11
Nighttime	0.34	0.12
Demonstration Or Protest	0.27	0.07
Airplane Flying	0.21	0.13
Boat Ship	0.21	0.15
Singing	0.17	0.05

Table 2.4: **Performance of the SVM image classifier.** The table reports the average precision of the method described in Section 2.5.2 when trained on TRAIN and evaluated on VAL, and TRECVID inferred AP when trained on TRAIN+VAL. To compute average precision on TRAIN+VAL the complete and cleaned annotations were used. In several cases the difference in AP and xinfAP is very large, suggesting either that there is a significant statistical difference between the DEVEL and TEST data subsets, or that the accuracy of the xinfAP estimate is poor (xinfAP may still be adequate to rank different methods).

Figure 2.10: Top 15 shots from the TEST set (shown by keyframes) using an SVM for Airplane Flying category

Figure 2.11: Top 15 shots from the TEST set (shown by keyframes) using an SVM for Cityscape category

Figure 2.12: Top 15 shots from the TEST set (shown by keyframes) using an SVM for Classroom category

Figure 2.13: Top 15 shots from the TEST set (shown by keyframes) using an SVM for Demonstration or Protest category

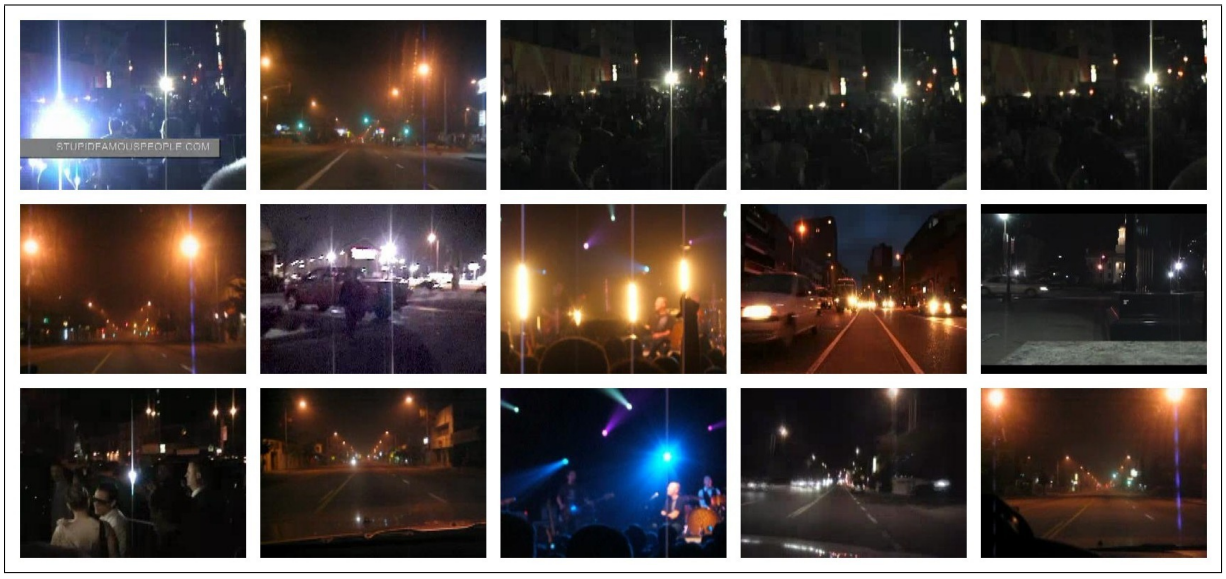


Figure 2.14: Top 15 shots from the TEST set (shown by keyframes) using an SVM for Nighttime category

2.5.3 Results

The results on the TEST set are reported as xinfAP (inferred AP) [40], which is the measure used by TRECVID to evaluate a method. Based on the evaluations by TRECVID, our method was ranked the best method for *Nighttime*, and second best for *Hand, Airplane flying*. Figure 2.10 shows the qualitative results for Airplane Flying category. Quantitative results on the VAL and TEST set are reported in Table 2.4. For evaluating the results on the VAL set, we show the standard mean AP values.

Scene Classification Results

Our method worked well for most of the scene categories, (e.g. *Nighttime, Cityscape, Demonstration or Protest, Classroom*. The qualitative results for scene categories are shown in Figure 2.11 for Cityscape, Figure 2.12 for Classroom, Figure 2.13 for Demonstration or Protest, Figure 2.14 for Nighttime. These results are obtained on the shots from the TEST set.

Chapter 3

Bag of Words for Scene Classification

In this chapter, we explore the Bag of Words model for the problem of scene classification. We examine the different steps of the Bag of Words pipeline, and use the best options at each step. Our well designed pipeline is not only fast and efficient, but also establishes the state-of-the-art results on MIT Indoor Scene 67 dataset, beating all the previously published results. We also go beyond the standard Spatial Pyramid technique, and demonstrate an idea of non-uniform partitioning in images which may be more suitable for scene classification tasks.

3.1 Introduction

We explained the standard Bag of Words model in Section 2.4. A lot of methods have been proposed in the literature for image classification tasks. The aim is to develop better and improved models to solve the classification problem. But in the process, the power of Bag of Words model is often overlooked. Recently, Chatfield *et al*. [10] demonstrated that a well designed Bag of Words pipeline can achieve state-of-the-art results on the challenging datasets like Caltech 101 [17] and PASCAL VOC 2007 [15]. Motivated by this, we explore the Bag of Words model for the problem of Scene Classification in images. At each step of the Bag of Words pipeline, we pick the best choices available. Section 3.2 explains the essential components of our well designed Bag of Words pipeline, Section 3.3 explains the method (image representation and learning), Section 3.4 shows the state-of- the-art results on MIT Indoor Scene 67 dataset obtained with our Bag of Words pipeline.

We also look at the possibility of improvements in the standard Spatial Pyramid method. By design, Bag of Words does not capture the spatial information, some weak spatial information can be introduced by using Spatial Pyramid [31]. Instead of having a uniform division of an image into cells, we present some ideas to have a non-uniform division of image into cells, which may be more suitable for the scene classification task in hand. Section 3.5 discusses some of these ideas and shows some results obtained using our proposed ideas.

3.2 Bag of Words for Scene Classification

Section 2.4 explains the standard Bag of Words pipeline. The major steps in the pipeline are: (a) Extracting local image features, (b) Generating vocabulary of visual words, (c) Computing feature encodings and generating spatial histograms, and (d) Learning a classifier based on the image descriptors.

We investigate the best options for each step of the BoW pipeline, as explained in [10]

(a) Local Image Features

(b) Encoding methods:

(i) Hard assignment (*i.e*. Vector Quantisation)

(ii) Kernel-codebook encoding(KCB)

(iii) Locality-constrained Linear Coding (LLC)

(iv) Improved Fisher Vector (IFV)

3.2.1 Local Image Features

The standard Bag of Words pipeline 2.4 uses SIFT descriptors [36] as the local image descriptors. It is a well known fact that using $\chi 2$ or Hellinger distance to compare histograms often yields superior performance compared to using Euclidean distance measure. Since SIFT is essentially a histogram, a natural question is whether using alternative histogram distance measures benefit SIFT. Arandjelovic and Zisserman [4] show that using a square root (Hellinger) kernel instead of the standard Euclidean distance to measure the similarity between SIFT descriptors leads to a dramatic performance boost in image retrieval tasks. In practice, the conversion from SIFT to RootSIFT is done as a simple algebraic operation in two steps:

(i) L1 normalize the SIFT vector (originally it has unit L2 norm)

(ii) square root each element of the SIFT vector.

This conversion can be done online, without any additional storage space. In our pipeline, we replace SIFT by RootSIFT at each step.

3.2.2 Encoding Methods

In the standard Bag of Words pipeline 2.4, after the codebook construction, the next step is to encode the local features using the codebook. These codes are then pooled together to compute the descriptor for an image. The standard pipeline uses a hard quantisation (vector quantisation) to encode the local image features, where each local feature is encoded by the nearest visual word in the Euclidean space. Alternative encodings have been proposed which replace the hard quantisation of the features and retain more information about the original features. These alternate encodings try to encode the local features

in different ways, *e.g*. by expressing features as combinations of visual words, or by recording the difference between the features and the visual words.

We investigate a number of encoding methods as described in [10]: (i) Hard assignment (*i.e*. Vector Quantisation) [12, 59]; (ii) Kernel-codebook encoding [47, 65]; (iii) Locality-constrained Linear Coding (LLC) [69]; and (iv) Improved Fisher Vector (IFV) [45] . For all the encoding methods except IFV, let us assume:

- x_i be the local descriptors; $x_i \in R^D$, and D is the dimension of the local descriptors.
- the codebook be represented by the visual words $\mu_1, \mu_2, \ldots \mu_K$; $\mu_i \in R^D$, and K is the size of the codebook. The codebook is generated using K-means (Section 2.4.2).

Vector Quantisation (Hard assignment)

Vector Quantisation (**VQ**) [12, 59] is the simplest encoding method in which the code for a local descriptor x_i is given by the nearest visual word in the Euclidean space (Equation 3.1). Since each local descriptor is strictly encoded in terms of a single visual word, this encoding is also called as Hard Assignment.

$$c_i = \arg\min_k \|x_i - \mu_k\|^2 \tag{3.1}$$

Kernel Codebook Encoding

Kernel Codebook Encoding **(KCB)** [47, 65] is an encoding method where a local descriptor is assigned to multiple visual words, weighted by some distance measure in the descriptor space. It tries to solve two drawbacks of Hard Assignment encoding:

(i) two local descriptors which are very close to each other may be assigned to different visual words,

(ii) all the descriptors assigned to a single visual word are given the same weights, irrespective of the distance between the descriptor and the visual word.

The encoding of a local descriptor x_i is defined in terms of weights corresponding to the visual words (Equation 3.2)

$$c_{ik} = \frac{K(x_i, \mu_k)}{\sum_{j=1}^{K} K(x_i, \mu_j)} \tag{3.2}$$

$$\text{where} K(x, \mu) = exp(-\tfrac{\gamma}{2}\|x - \mu\|^2)$$

Locality-constrained linear (LLC) encoding

LLC encoding [69] utilizes the locality constraints to project each local descriptor into its local-coordinate system. Let the codebook be denoted by $B = [\mu_1, \mu_2, \ldots, \mu_K]$. LLC encoding tries to minimise the following objective function:

$$\min_{C} \sum_{i=1}^{N} \|x_i - Bc_i\|^2 + \lambda \|d_i \odot c_i\|^2 \tag{3.3}$$

$$s.t. \quad 1^T c_i = 1, \forall i$$

where $C = [c_1, c_2, \ldots, c_N]$ is the set of codes corresponding to local descriptors $x_1, x_2, \ldots, x_N$; $c_i \in R^D$

$$d_i = \exp(\frac{\text{dist}(x_i, B)}{\sigma}) \tag{3.4}$$

where $\text{dist}(x_i, B) = [\text{dist}(x_i, \mu_1), \text{dist}(x_i, \mu_2), \ldots, \text{dist}(x_i, \mu_K)]^T$, and $\text{dist}(x_i, \mu_j)$ is the Euclidean distance between local descriptor x_i and visual word μ_j. σ is used for adjusting the weight decay speed for the locality adaptor.

The analytical solution to the above optimisation is given as:

$$c_i = (C_i + \lambda \text{diag(d)}) \backslash 1 \tag{3.5}$$

where C_i is called the data covariance matrix, given by $C_i = (B - 1x_i^T)(B - 1x_i^T)^T$.

The codes c_i (Equation 3.5) obtained are finally L1-normalized.

Improved Fisher Vector (IFV)

Fisher encoding [45] captures the average first and second order differences between the local descriptors and the centers of a GMM, which can be thought of as a soft visual vocabulary. The construction of the encoding starts by learning a Gaussian Mixture Model (GMM) from the entire data set, and then associating the N local descriptors (from an image) to the K Gaussian components using soft assignment. Let, q_{ki}, $k = 1, \ldots, K$, $i = 1, \ldots, N$ be the soft assignments of the N local descriptors to the K Gaussian components, and for each $k = 1, \ldots, K$, define the vectors

$$\mathbf{u}_k = \frac{1}{N\sqrt{\pi_k}} \sum_{i=1}^{N} q_{ik} \Sigma_k^{-\frac{1}{2}} (\mathbf{x}_t - \mu_k),$$

$$\mathbf{v}_k = \frac{1}{N\sqrt{2\pi_k}} \sum_{i=1}^{N} q_{ik} \left[(\mathbf{x}_t - \mu_k) \Sigma_k^{-1} (\mathbf{x}_t - \mu_k) - 1 \right]. \tag{3.6}$$

Then the Fisher encoding of the set of local descriptors is given by the concatenation of $\mathbf{u}_k$ and $\mathbf{v}_k$ for all K components, giving an encoding of size $2DK$

$$\mathbf{f}_{\text{Fisher}} = [\mathbf{u}_1^\top, \mathbf{v}_1^\top, \ldots \mathbf{u}_K^\top, \mathbf{v}_K^\top]^\top. \tag{3.7}$$

Pooling

After getting the encodings for the local descriptors, the encodings are pooled to generate the spatial histograms (Section 2.4.3). The pooling can be done in following two ways:

- **Sum Pooling :**

  In sum pooling, the value of each bin is equal to the sum of the encodings of the local descriptors corresponding to the visual word.

$$h_j = \sum_{i=1}^{N} c_{ij} \qquad j = 1, \ldots, K \tag{3.8}$$

- **Max Pooling :**

  In max pooling, the value of each bin is equal to the maximum across all the encodings of the local descriptors corresponding to the visual word.

$$h_j = \max_{i=1,\ldots,N} c_{ij} \qquad j = 1, \ldots, K \tag{3.9}$$

The histogram obtained, $H = [h_1, h_2, \ldots, h_K]$ is the global descriptor of the image.

3.3 Image Representations and Learning

3.3.1 Dense visual words.

Dense RootSIFT descriptors [4, 36] are extracted from the image with a spatial stride of three to five pixels and at six scales, defined by rescaling with factors $2^{-\frac{i}{2}}, i = 0, 1, \ldots, 5$. Low-contrast SIFT descriptors (identified as the ones for which the average intensity gradient magnitude is below a threshold) are mapped to null vectors. The RootSIFT descriptors are then mapped to visual words. For the IFV encoding, the visual word dictionary is obtained by training a Gaussian Mixture Model (diagonal covariance) with 256 centers; for the other encodings, the dictionary is learned by using k-means and setting k to 2,000.

3.3.2 Spatial encoding.

Weak geometric information is retained in the descriptors by using spatial histogramming [31]. For the IFV encoding, the image is divided into 1×1, and 2×2 grids, obtaining a total of 5 spatial pooling regions; for the other encodings, the image is divided into 1×1, and 2×2, and 4×4 grids, obtaining a total of 21 spatial pooling regions. For the IFV encoding, we use max pooling for encoding the local descriptors; for the other encodings we use sum pooling. The descriptors for each region are individually normalised, and then stacked together to give the final image descriptor. For the IFV encoding, the image descriptor is 204,800-dimensional, and for other encodings, the image descriptor is 42,000-dimensional.

3.3.3 Learning and Classification.

Learning uses the PEGASOS SVM [55] algorithm, a linear SVM solver. In order to use non-linear additive kernels instead of the linear one, the χ^2 explicit feature map of [68] is used. Using the feature map increases the dimension of the input feature vector by 3 times. For the IFV encoding, we use square-root (Hellinger) kernel. The parameter C of the SVM (regularisation-loss trade-off) is determined by 4-fold cross validation. For multi-class image classification problems, 1-vs-rest classifiers are learned. In this case, it was found beneficial to calibrate the different 1-vs-rest scores by fitting a sigmoid [48] to them based on a validation set.

Class	**VQ**	**KCB**	**LLC**	**IFV**	**Class**	**VQ**	**KCB**	**LLC**	**IFV**
airport_inside	35.00	40.00	45.00	55.00	inside_subway	61.90	61.90	71.43	80.95
artstudio	10.00	15.00	20.00	30.00	jewelleryshop	40.91	18.18	31.82	36.36
auditorium	61.11	55.56	61.11	72.22	kindergarden	70.00	70.00	75.00	70.00
bakery	26.32	26.32	31.58	31.58	kitchen	52.38	52.38	47.62	52.38
bar	33.33	33.33	33.33	44.44	laboratorywet	31.82	27.27	27.27	50.00
bathroom	55.56	66.67	55.56	66.67	laundromat	59.09	54.55	59.09	81.82
bedroom	33.33	33.33	28.57	33.33	library	45.00	50.00	45.00	45.00
bookstore	35.00	35.00	50.00	50.00	livingroom	40.00	35.00	50.00	45.00
bowling	90.00	95.00	90.00	90.00	lobby	40.00	40.00	45.00	55.00
buffet	65.00	60.00	70.00	70.00	locker_room	47.62	42.86	47.62	57.14
casino	73.68	73.68	68.42	68.42	mall	40.00	40.00	45.00	45.00
children_room	38.89	38.89	38.89	33.33	meeting_room	59.09	54.55	50.00	68.18
church_inside	73.68	73.68	68.42	78.95	movietheater	50.00	55.00	50.00	55.00
classroom	55.56	55.56	72.22	66.67	museum	34.78	34.78	39.13	43.48
cloister	95.00	90.00	95.00	90.00	nursery	70.00	70.00	70.00	70.00
closet	72.22	66.67	72.22	77.78	office	9.52	14.29	9.52	19.05
clothingstore	61.11	61.11	55.56	72.22	operating_room	42.11	42.11	42.11	36.84
computerroom	83.33	77.78	83.33	83.33	pantry	60.00	60.00	60.00	75.00
concert_hall	60.00	50.00	60.00	75.00	poolinside	50.00	35.00	45.00	60.00
corridor	57.14	57.14	57.14	61.90	prisoncell	55.00	55.00	60.00	70.00
deli	5.26	15.79	26.32	10.53	restaurant	40.00	40.00	30.00	60.00
dentaloffice	61.90	61.90	52.38	57.14	restaurant_kitchen	60.87	52.17	43.48	60.87
dining_room	38.89	33.33	38.89	50.00	shoeshop	42.11	36.84	31.58	47.37
elevator	76.19	71.43	85.71	85.71	stairscase	60.00	60.00	55.00	75.00
fastfood_restaurant	52.94	52.94	76.47	82.35	studiomusic	78.95	73.68	78.95	84.21
florist	73.68	73.68	73.68	73.68	subway	38.10	42.86	52.38	61.90
gameroom	45.00	35.00	35.00	55.00	toystore	22.73	18.18	31.82	45.45
garage	66.67	50.00	66.67	66.67	trainstation	60.00	65.00	60.00	75.00
greenhouse	80.00	80.00	80.00	85.00	tv_studio	72.22	72.22	61.11	72.22
grocerystore	47.62	42.86	42.86	57.14	videostore	45.45	40.91	40.91	40.91
gym	44.44	33.33	33.33	72.22	waitingroom	19.05	23.81	28.57	33.33
hairsalon	42.86	57.14	52.38	66.67	warehouse	47.62	47.62	52.38	52.38
hospitalroom	65.00	70.00	65.00	75.00	winecellar	57.14	47.62	57.14	76.19
inside_bus	73.91	73.91	73.91	82.61	**Average**	**52.14**	**50.59**	**53.03**	**60.77**

Table 3.1: Per-class classification accuracies for Bag of Words (BoW). All results in %.

Encoding	Step size: 3	Step size: 4	Step size: 5
VQ [42,000]	52.14 (51.08)	50.38 (50.50)	49.76 (50.42)
KCB [42,000]	50.59 (49.65)	49.21 (49.19)	50.41 (49.92)
LLC [42,000]	51.52 (51.87)	53.03 (51.73)	51.70 (52.09)
IFV [204,800]	60.33 (60.90)	60.67 (61.39)	**60.77** (61.05)

Table 3.2: **BoW scene classification results.** The table compares various BoW classifiers on MIT Scene 67, reporting classification accuracy and mean average precision (in parentheses) for each case. The best results significantly outperform the current state-of-the-art accuracy. The dimension of the image representation is given in square brackets.

Method	Accuracy (%)	Mean AP (%)
ROI + Gist [49]	26.05	-
MM-scene [75]	28.00	-
CENTRIST [72]	36.90	-
Object Bank [34]	37.60	-
DPM [41]	30.40	-
RBoW [42]	37.93	-
LPR [51]	44.84	-
Patches [58]	38.10	-
DPM+Gist-color+SP [41]	43.10	-
Patches+GIST+SP+DPM [58]	49.40	-
VQ [42,000]	52.14	51.08
KCB [42,000]	50.59	49.65
LLC [42,000]	53.03	51.73
IFV [204,800]	60.77	61.05

Table 3.3: Average classification performance of different methods (previous publications and this paper). The dimension of the image representation is given in square brackets.

3.4 Experiments and results

The four variants of BoW were compared (Section 3.2.2, Table 3.2) by varying the encoding and the sampling density of the RootSIFT features. The best performance is 60.77% by using the IFV encoding with sampling step of five pixels.

Very recently, [58] proposed a part-learning method demonstrating state-of-the-art performance in the MIT Scene 67 dataset. Their best performing achieves an accuracy of 49.4% combining, in additions to the learned parts, BoW, GIST, and DPM representations. We were therefore mildly surprised to find that, by following the best practices indicated by [10], a solid bag of word implementation is actually able to *outperform* (by 11%) the combined method of [58] as well as all other previous methods on the MIT Scene 67 by using a *single feature channel* based on RootSIFT features (Table 3.3). Table 3.1

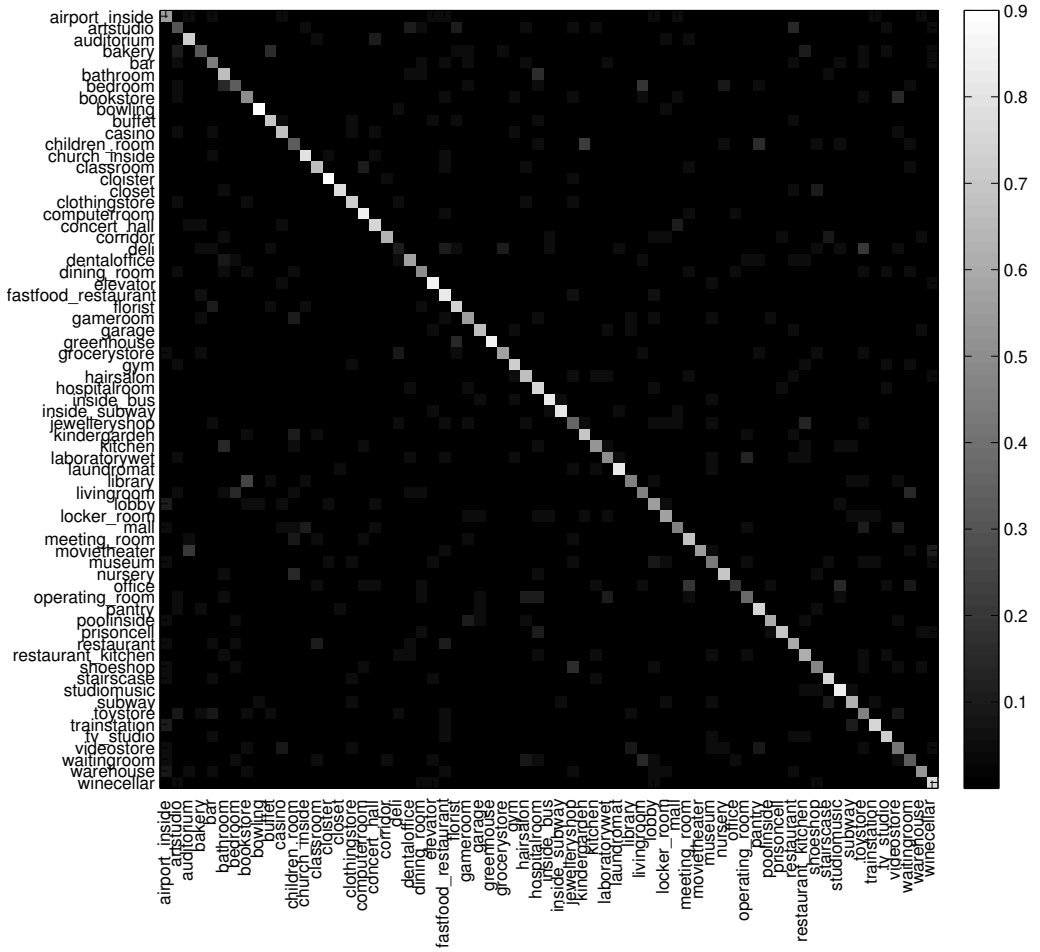


Figure 3.1: **Confusion matrix obtained with IFV.** The diagonal represent the correct classifications, and the brighter non-diagonal spots denote the confusion between two classes. The average of the diagonal of the confusion matrix gives the classification accuracy.

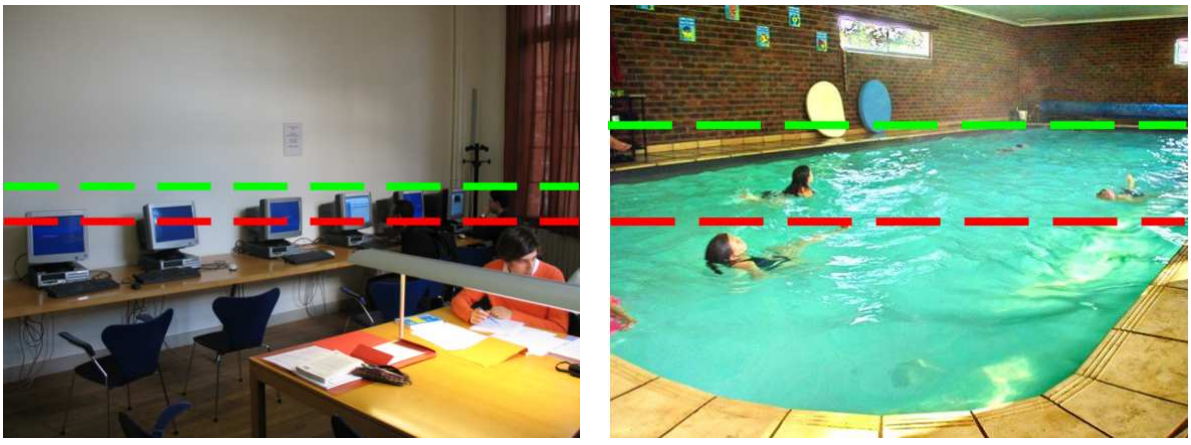
reports the per-class classification accuracy for the Bag of Words model. We also show the confusion matrix obtained with IFV encoding (Figure 3.1). The brighter spots in the confusion matrix shows the confusion between different classes, mainly because of the inter-class similarity and semantic overlap between classes (Section 1.2).

3.5 Beyond Spatial Pyramids

By design, Bag of Words does not capture the spatial information. Weak spatial information can be introduced by using Spatial Pyramid [31], as described in Section 2.4.3. However, the division of the image into cells is done uniformly across all the images. For example, for a 2×2 grid of cells, the

image is divided from center in both horizontal and vertical directions. But for the image classification problem, the uniform division may not be the most useful one. Consider the scene classification task, a better strategy would be to divide the image in such a way that each cell is able to capture some homogeneous regions of the scene. Figure 3.2 shows this with some examples. The red dotted line shows the uniform partitioning of the image into a 2×1 grid (passing through the center of the image), while the green dotted line shows a better partitioning of the image, where each cell has more homogeneous content of the scene. For the computer-room image, the green line is able to separate the computers from the background wall. For the swimming pool image, the complete swimming pool is in the lower half (green line), while in case of uniform partition, the red line cuts the swimming pool into 2 parts, and it appears in both the lower and upper half.

There has been some recent work in this direction of finding discriminative, non-uniform partitions of the images. Sharma and Jurie [57] proposed a method of learning discriminative spatial partitioning from a space of grids, where each grid is obtained by a series of recursive axis aligned splits of cells. Hakan *et al*. [26] formulate the estimation of the partition as a latent variable problem. Both these methods use axis-aligned grids for partitioning the image. A better method would be one in which the partitions can have more degrees of freedom, like a piecewise linear line. In this section, we present some ideas for estimating better partitions in images, which can be used with Bag of Words model. We start with a small experiment (Section 3.5.1) to check whether the non-uniform partitioning helps in scene classification or not. We then formulate the problem of estimating the partition as Maximum Likelihood Estimation (Section 3.5.2) and estimate it using an iterative procedure.



(a) Computer Room (b) Swimming Pool

Figure 3.2: Examples showing the uniform partitioning v/s non-uniform partitioning in images.

3.5.1 Manual partition

This section describes a simple classification experiment which is performed to verify whether the non-uniform partitioning gives any boost in the classification performance, as compared to the regular uniform partitioning. For this experiment, we consider 3 classes from the MIT 67 indoor scene dataset [49]: auditorium, concert hall and movie theater. The 3 classes are quite confusing, and the images from the 3 classes look very similar to each other. We manually partitioned the images into 2×1 grid, ensuring that the regions in the 2 cells are homogeneous. The dividing line need not be axis parallel, it can be an inclined line as well. Figure 3.3 shows some partitions in images from the auditorium class.

There are 100 images for each class, out of which 80 are used for training and 20 for testing. The classification pipeline used is same as described in Section 2.4. The only change comes while computing the spatial histograms. Instead of using regular grids, we use the new partitions for constructing the spatial histograms. We use the histograms of the base image as well, as in the standard Spatial Pyramid technique. In this experiment for 2×2 partition, we use the non-uniform partitioning in the horizontal direction, the image is partitioned vertically from the center. Table 3.4 shows the classification results measured in terms of average classification accuracy, and compares the results with uniform partitioning. The results do support the claim that a non-uniform partitioning performs better than a uniform partitioning.

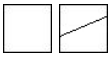
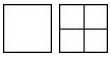
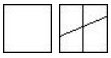
Method	Representation	Accuracy (%)
Base Image		69.63
Base Image + 2×1 Uniform Partition		67.78
Base Image + 2×1 Manual Partition		71.67
Base Image + 2×2 Uniform Partition		69.81
Base Image + 2×2 Manual Partition		73.52

Table 3.4: Classification results for 3-class experiment.

3.5.2 Maximum Likelihood Estimation of the Partition

In this section, we propose a Maximum Likelihood Estimate (MLE) [46] based solution to estimate the partitions in an image. In this section, we consider 2×1 partition in an image. As described in Section 3.5.1, a partition which divides the image into homogeneous regions should be a good partition. One way to estimate this partition would be to estimate the distribution of visual words for the upper and lower regions of images belonging to one class. For example, lower region of all swimming pool

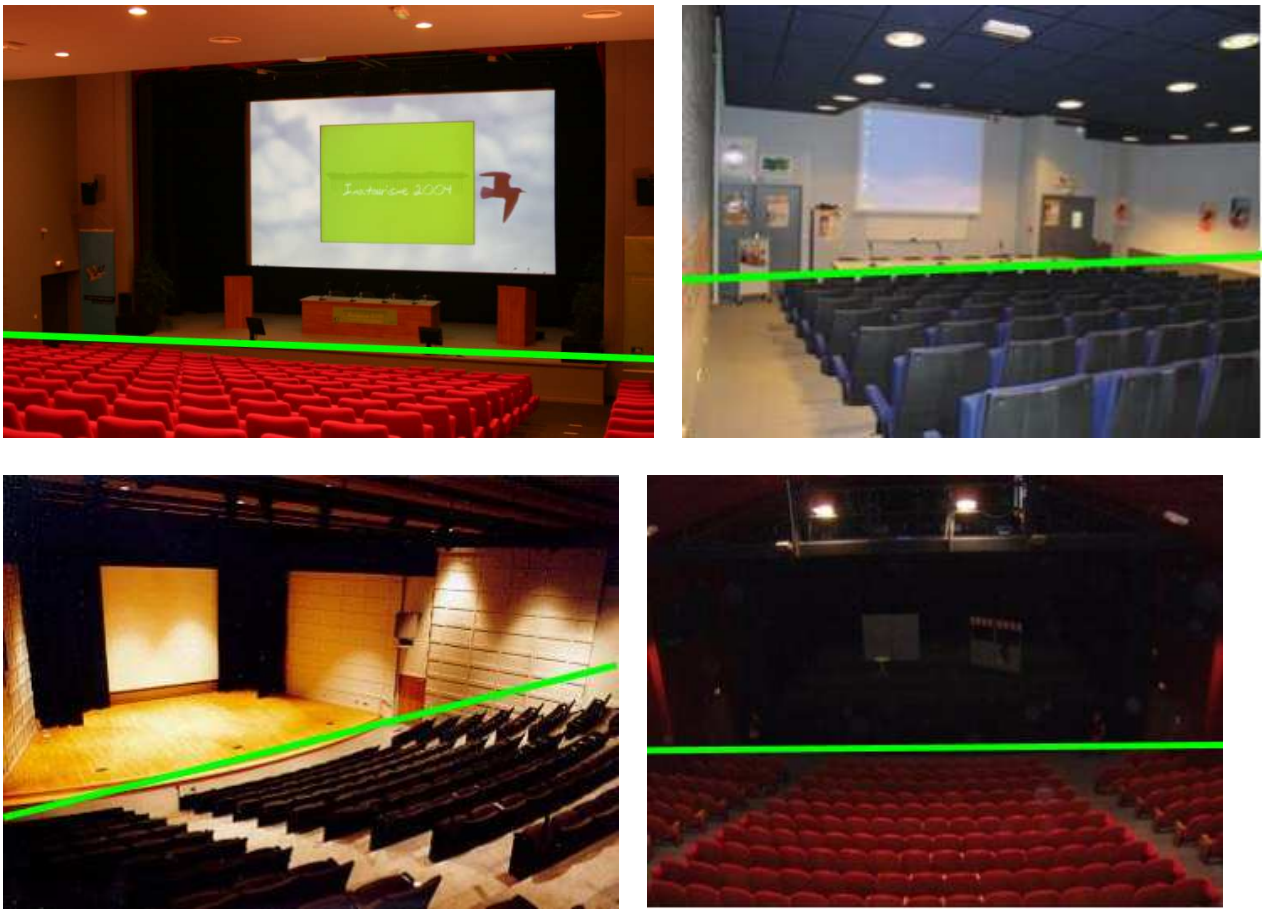


Figure 3.3: **Manual Partitioning.** Examples of manual partitioning in images from the auditorium class.

images (Figure 3.2) should be homogeneous, consisting of the swimming pool (water), and the upper region should consist of the homogeneous background walls.

Let A and B be the upper and lower parts of the image, defined by the partition H. The partition H need not be a straight line, it can be estimated as piecewise linear. We formulate the MLE solution as follows:

- Let the data be the feature occurrences $x_1, ..., x_n$ in the whole image. In our case, the features are the points at which SIFT features are computed.
- Let θ_A, θ_B be the parameters of a probabilistic model of the data. Here, θ_A, θ_B are the histograms ($l1$ normalised) of visual words in the upper and lower parts of the image respectively.

- Now, the probability (density) of a feature x_i is defined as

$$p(x_i|\ \theta_A, \theta_B) = \begin{cases} p(x_i|\theta_A) & \text{if } x_i \text{ in } A \\ p(x_i|\theta_B) & \text{if } x_i \text{ in } B \end{cases} \tag{3.10}$$

  where $p(x_i|\theta) = \theta(W^{x_i})$ (W^{x_i} is the visual word at x_i point and $\theta(v)$ is the count of visual word v in θ histogram)

- The probability of all the features jointly is the product (as we assume them to be iid)

$$p(x_1, \ldots, x_n|\theta_A, \theta_B) = \prod_i p(x_i|\theta_A, \theta_B) \tag{3.11}$$

  Converting into the log-likelihood for simplicity, we get

$$\log p(x_1, \ldots, x_n|\theta_A, \theta_B) = \sum_{x_i \in A} \log p(x_i|\theta_A) + \sum_{x_i \in B} \log p(x_i|\theta_B) \tag{3.12}$$

The solution is given by the regions A, B which maximises the score (Equation 3.12). In practice, we use an iterative procedure to find the solution. The pseudocode is given below (Algorithm 2).

Algorithm 2: Pseudocode for estimating partition.

Input: Image I
Output: A, B
Initialisation: $s = 0$, A, B: partition by center line in I
$x_1, \ldots, x_n$ = points at which SIFT feature are computed in I
$W = [v_1, \ldots, v_K]$ (vocabulary constructed from the SIFT features, using k-means)
Visual word v at point x_i is given by W^{x_i}
while *s is not converged* **do**

$$\theta_A[v_j] = \sum_{i=1}^{n} (W^{x_i} == v_j) \quad \text{if } x_i \text{ in } A$$

$$\theta_B[v_j] = \sum_{i=1}^{n} (W^{x_i} == v_j) \quad \text{if } x_i \text{ in } B$$

$$A, B = \arg\max_{A', B'} \Big(\sum_{x_i \in A'} \log p(x_i|\theta_A) + \sum_{x_i \in B'} \log p(x_i|\theta_B) \Big)$$

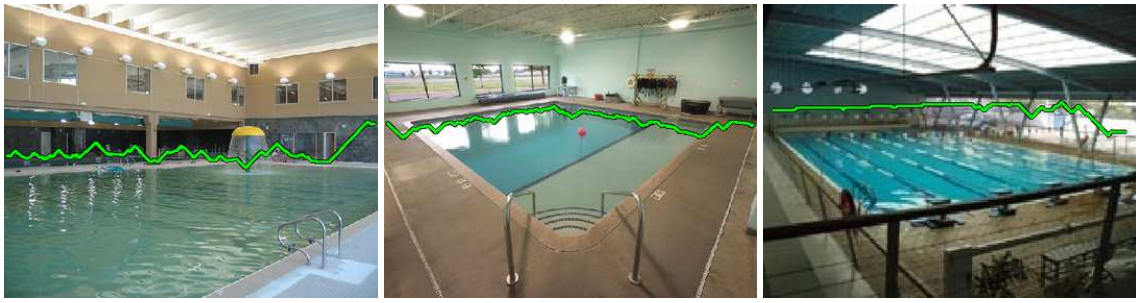
end

We use the above method of estimating non-uniform partitions and test it on images from MIT Scene 67 data set. Figure 3.4 shows some examples of the partitions obtained on some of the images.

(a) Auditorium

(b) Meeting Room

(c) Corridor



(d) Pool Inside

(e) Computer Room

Figure 3.4: Examples of partitions obtained on images from MIT 67 scene dataset.

3.6 Summary

We have presented a well designed Bag of Words pipeline for scene classification. We started with the standard BoW pipeline, and motivated by [10], explored the different options available for each step of the pipeline. As a result of our experiments, we are able to build a method which is not only fast and efficient, but also establishes the state-of-the-art results on MIT Indoor Scene 67 dataset, beating all the previously published results. We have also presented a novel idea of non-uniform partitioning in images which may be more suitable for classification tasks. This technique can be used as an alternative to the existing Spatial Pyramid technique.

Chapter 4

Bag of Parts for Scene Classification

The automatic discovery of distinctive parts for an object or scene class is challenging since it requires simultaneously to learn the part appearance and also to identify the part occurrences in images. We propose a simple, efficient, and effective method to do so. We address this problem by learning parts incrementally, starting from a single part occurrence with an Exemplar SVM. In this manner, additional part instances are discovered and aligned reliably before being considered as training examples. We also propose entropy-rank curves as a means of evaluating the distinctiveness of parts shareable between categories and use them to select useful parts out of a set of candidates.

We apply the new representation to the task of scene categorisation on the MIT Scene 67 benchmark. We show that our method can learn parts which are significantly more informative and for a fraction of the cost, compared to previous part-learning methods such as Singh *et al*. [58]. We also show that a well constructed bag of words or Fisher Vector model can substantially outperform the previous state-of-the-art classification performance on this data.

4.1 Introduction

The notion of *part* has been of central importance in object recognition since the introduction of pictorial structures [22], constellation models [71], object fragments [2, 61], right up to recent state-of-the-art methods such as Deformable Part Models (DPMs) [19]. Yet, the *automatic discovery* of good parts is still a difficult problem. In DPM, for example, part occurrences are initially assumed to be in a fixed location relative to the ground truth object bounding boxes, and then are refined as latent variables during learning [19]. This procedure can be satisfactory in datasets such as PASCAL VOC [14] where bounding boxes usually induce a good alignment of the corresponding objects. However, when the alignment is not satisfactory, as for the case of highly-deformable objects such as cats and dogs [43], this approach does not work well and the performance of the resulting detector is severely hampered.

In this work, a simple, efficient, and effective method for discovering parts automatically and with very little supervision is proposed. Its power is demonstrated in the context of scene recognition where, unlike in object recognition, object bounding boxes are not available, making part alignment very chal-

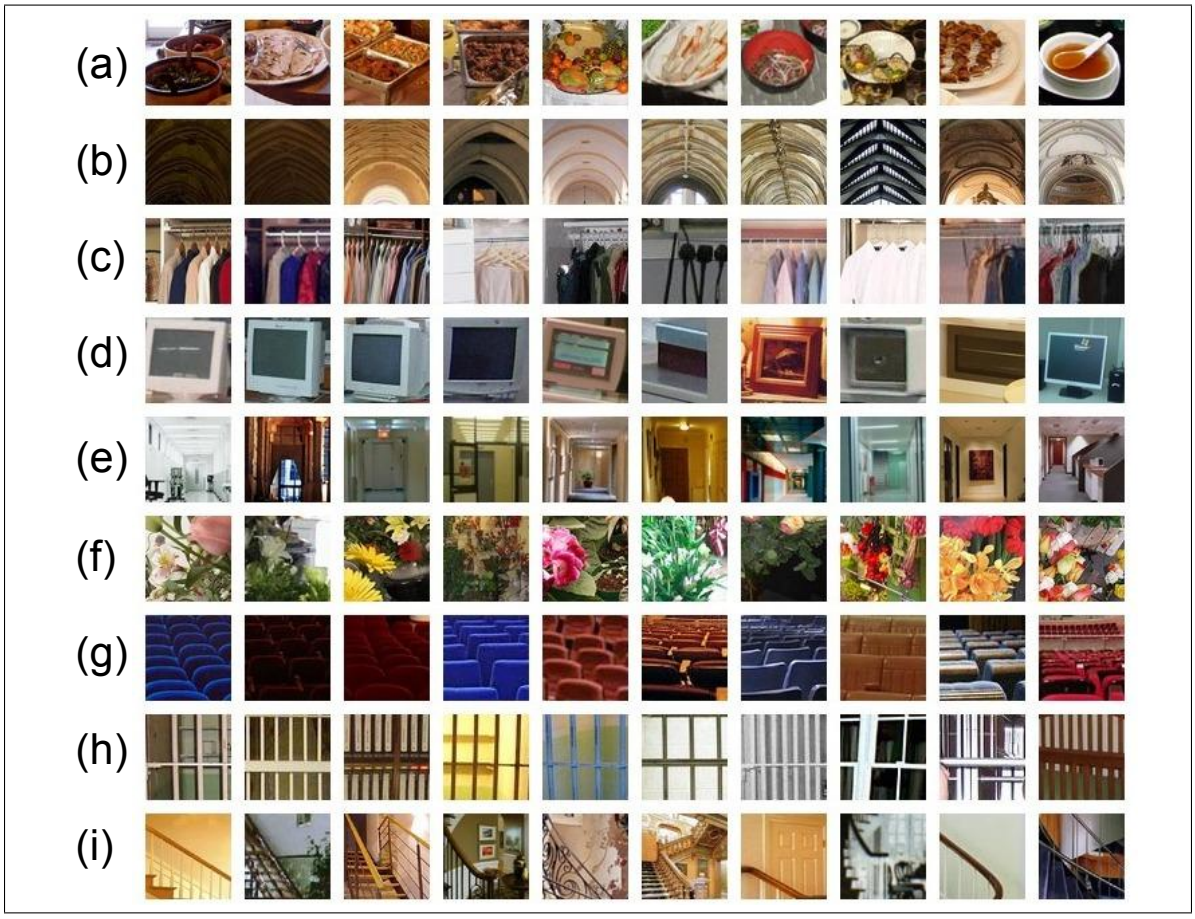


Figure 4.1: Example of occurrences of distinctive parts learned by our method from weakly supervised image data. These part occurrences are detected on the test data. (a) buffet, (b) cloister, (c) closet, (d) computerroom, (e) corridor, (f) florist, (g) movietheater, (h) prisoncell, (i) stairscase.

lenging. In particular, the method is tested on the MIT Scene 67 dataset, the standard benchmark for scene classification. Figure 4.1 shows examples of the learned parts detected on the test set.

To achieve these results two key issues must be addressed. The first is to find and align part instances in the training data while a model of the part is not yet available. This difficulty is bypassed by learning the model from a *single exemplar* of a candidate part. This approach is motivated by [38], that showed that a single example is often sufficient to train a reasonable, if a little restrictive, detector. This initial model is then refined by alternating mining for additional part instances and retraining. While this procedure requires training a sequence of detectors, the LDA technique of [27] is used to avoid mining for hard negative examples, eliminating the main bottleneck in detector learning [21, 67], and enabling a very efficient part learning algorithm.

The second issue is to select *distinctive parts* among the ones that are generated by the part mining process. The latter produces in fact a candidate part for each of a large set of initial seeds. Among

these, the most informative ones are identified based on the novel notion of *entropy-rank*. This criterion selects parts that are informative for a small proportion of classes. Differently to other measures such as average precision, the resulting parts can then be *shared* by more than one object category. This is particularly important because parts should be regarded as mid-level primitives that do not necessarily have to respond to a *single* object class.

The result of our procedure is the automatic discovery of distinctive part detectors. We call them "blocks that shout" due to their informative nature and due to the fact that, in practice, they are implemented as HOG [13] block filters (Figure 4.2).

4.1.1 Related work

Parts have often been sought as intermediate representations that can complement or substitute lower level alternatives like SIFT [36]. In models such as DPMs, parts are devoid of a specific semantic content and are used to represent deformations of a two dimensional template. Often, however, parts do have a semantic connotation. For example, in Poselets [9] object parts correspond to recognizable clusters in appearance and configuration, in Li *et al*. [34] scene parts correspond to object categories, and in Raptis *et al*. [50] action parts capture spatio-temporal components of human activities.

The learning of parts is usually integrated into the learning of a complete object or scene model [3, 19]. Only a few papers deal primarily with the problem of learning parts. A possibility, used for example by Poselets [9], is to use some spatial annotation of the training images for weakly supervised learning. Singh *et al*. [58] explores learning parts in both an unsupervised and weakly supervised manner, where the weakly supervised case (as here) only uses the class label of the image. Their weakly supervised procedure is applied to the MIT Scene 67 dataset, obtaining state-of-the-art scene classification performance. As will be seen though, our part-learning method is: (i) simpler, (ii) more efficient, and (iii) able to learn parts that are significantly better at scene classification.

4.2 Blocks that shout: learning distinctive parts

In characterising images of particular scene classes, *e.g*. computer room, library, book store, auditorium, theatre, etc., it is not hard to think of distinctive parts: chairs, lamps, doors, windows, screens, etc., readily come to mind. In practice, however, a distinctive part is useful only if it can be detected automatically, preferably by an efficient and simple algorithm. Moreover, distinctive parts may include other structures that have a weaker or more abstract semantic, such as the corners of a room or a corridor, particular shapes (rounded, square), and so on. Designing a good vocabulary of parts is therefore best left to *learning*.

Learning a distinctive part means identifying a localized detectable entity that is informative for the task at hand (in our example discriminating different scene types). This is very challenging because (i) one does not know if a part occurs in any given training image or not, and (ii) when the part

occurs, one does not know its location. While methods such as multiple instance learning have often been proposed to try to identify parts automatically, in practice they require careful initialisation to work well.

Our strategy for part learning combines three ideas: seeding, expansion, and selection. In *seeding*, (Section 4.2.1) a set of candidate part instances (seeds) is generated by sampling square windows (blocks) in the training data guided by segmentation cues. All such blocks are treated initially as potentially different parts. In *expansion*, (Section 4.2.2) each block is used as a seed to build a model for a part while gradually searching for more and more part occurrences in the training data. This paced expansion addresses the issue of detecting and localizing part exemplars. Finally, *selection* (Section 4.2.3) finds the most distinctive parts in the pool of candidate parts generated by seeding and expansion by looking at their predictive power in term of entropy-rank. The procedure is weakly supervised in that positives are only sought in the seeding and expansion stages within images of a single class.

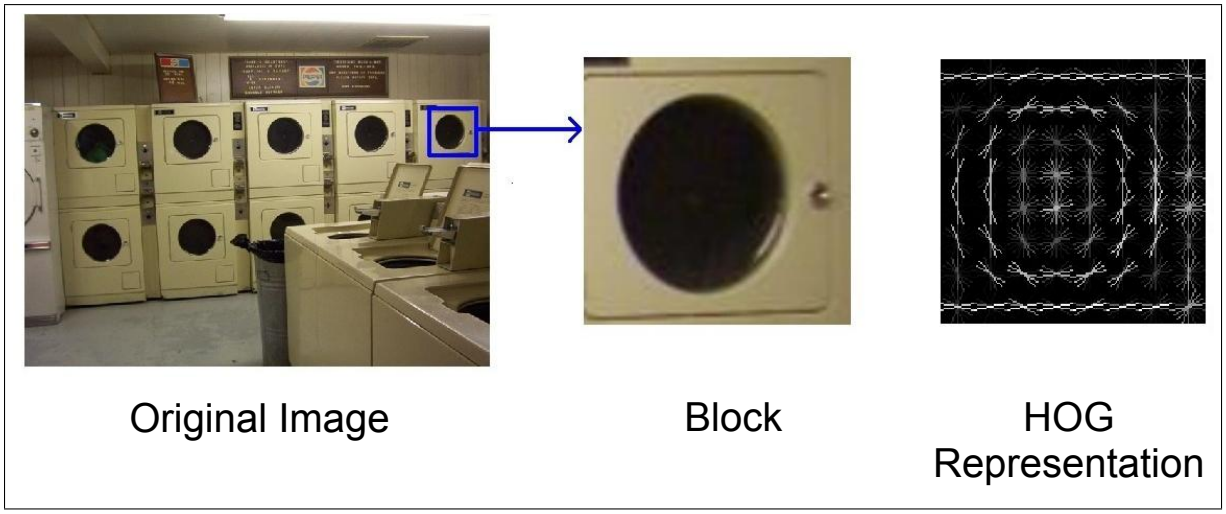
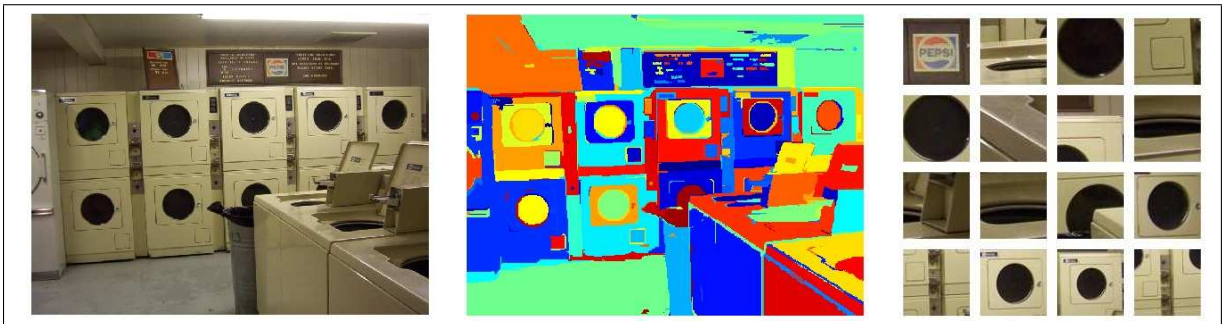


Figure 4.2: An example block learnt automatically for the laundromat class. This block is a characteristic part for this class.

Once these distinctive parts are obtained, they can be used for a variety of tasks. In our experiments they are encoded in a similar manner to BoW: an image descriptor is computed from the maximum response of each part pooled over spatial regions of the image (Section 3.2). Experiments show that this representation is in fact *complementary* to BoW (Section 4.4.2).

4.2.1 Seeding: proposing an initial set of parts

Initially, no part model is available and, without further information, any sub-window in any training image is equally likely to contain a distinctive part. In principle, one could simply try to learn a part model by starting from *all possible image sub-windows* and identify good parts *a-posteriori*, during the



(a) Original Image (b) Superpixels (c) Seed blocks

Figure 4.3: **Selecting seed blocks.** The super-pixels (b) suggest characteristic regions of the image, and blocks are formed for these. Blocks with low spatial gradient are discarded.

selection stage (Section 4.2.3). Unfortunately, most of these parts will in fact not be distinctive (*e.g.* a uniform wall) so this approach is highly inefficient.

We use instead low-level image cues to identify a subset of image locations that are more likely to be centered around distinctive parts. Extending the notion of *objectness* [1], we say that such promising locations have high *partness*. In order to predict partness, we use image over-segmentations, extending the idea of [64] from objects to parts.

In detail, each training image is first segmented into superpixels by using the method of [20]. This is repeated four times, by rescaling the image with scaling factors $2^{-\frac{i}{3}}, i = 0, 1, 2, 3$. Superpixels of intermediate sizes, defined as the ones whose area is in the range 500 to 1,500 pixels, are retained. These threshold are chosen assuming that the average area of an unscaled image is 0.5 Mpixels. Superpixels which contain very little image variation (as measured by the average norm of the intensity gradient) are also discarded.

Part models are constructed on top of HOG features [13]. At each of the four scales, HOG decomposes the image into cells of 8×8 pixels. Each part is described by a block of 8×8 HOG cells, and hence occupies an area of 64×64 pixels. A part seed is initialized for each superpixel by centering the 64×64 pixel block at the center of mass of the superpixel. Compared to sampling blocks uniformly on a grid, this procedure was found empirically to yield a much higher proportion of seeds that successfully generate good discriminative parts.

Figure 4.3 shows an example of the superpixels computed from a training image, and the seed blocks obtained using this procedure.

4.2.2 Expansion: learning part detectors

Learning a part detector requires a set of part exemplars, and these need to be identified in the training data. A possible approach is to sample at random a set of part occurrences, but this is extremely unlikely

to hit multiple occurrences of the same part. In practice, part initialisation can be obtained by means of some heuristic, such as clustering patches, or taking parts at a fixed location assuming that images are at least partially aligned. However, the detector of a part is, by definition, the most general and reliable tool for the identification of that part occurrences.

There is a special case in which a part detector can be learned without worrying about exemplar alignment: a training set consisting exactly of one part instance. It may seem unlikely that a good model could be learned from a single part example, but Exemplar SVMs [38] suggest that this may be in fact the case. While the model learned from a single occurrence cannot be expected to generalise much, it is still sufficient to identify reliably at least a few other occurrences of the part, inducing a gradual expansion process in which more and more part occurrences are discovered and more variability is learned.

In practice, at each round of learning the current part model is used to rank blocks from images of the selected class and the highest scoring blocks are considered as further part occurrences. This procedure is repeated a set number of times (ten in the experiments), adding a small number of new part exemplars (ten) to the training set each time. All the part models obtained in this manner, including the intermediate ones, are retained and filtered by distinctiveness and uniqueness in Section 4.2.3. Figure 4.4 shows an example seed part on the left, and the additional part occurrences that are are added to the training set during successive iterations of expansion.

Note that this expansion process uses a *discriminative* model of the part. This is particularly important because in image descriptors such as HOG most of the feature components correspond to irrelevant or instance specific details. Discriminative learning can extract the distinctive information (*e.g.* shape), while generative modelling (*e.g.* k-means clustering) has difficulty in doing so and constructing "semantic" clusters.

LDA acceleration. The downside of this mining process is that the part detector must be learned multiple times. Using a standard procedure that involves hard negative mining for each trained detector [21, 67] would then be very costly. We use instead the LDA technique of [27] which can be seen as learning *once* a soft but universal model of *negative* patches (a similar method is described in [25]). In practice, the parameter vector w of a part classifier is learned simply as $w = \Sigma^{-1}(\bar{x} - \mu_0)$ where $\bar{x}$ is the mean of the HOG features of the positive part samples, μ_0 is the mean of the HOG blocks in the dataset, and Σ the corresponding covariance matrix. HOG blocks are searched at all locations at the same four scales of Section 4.2.1.

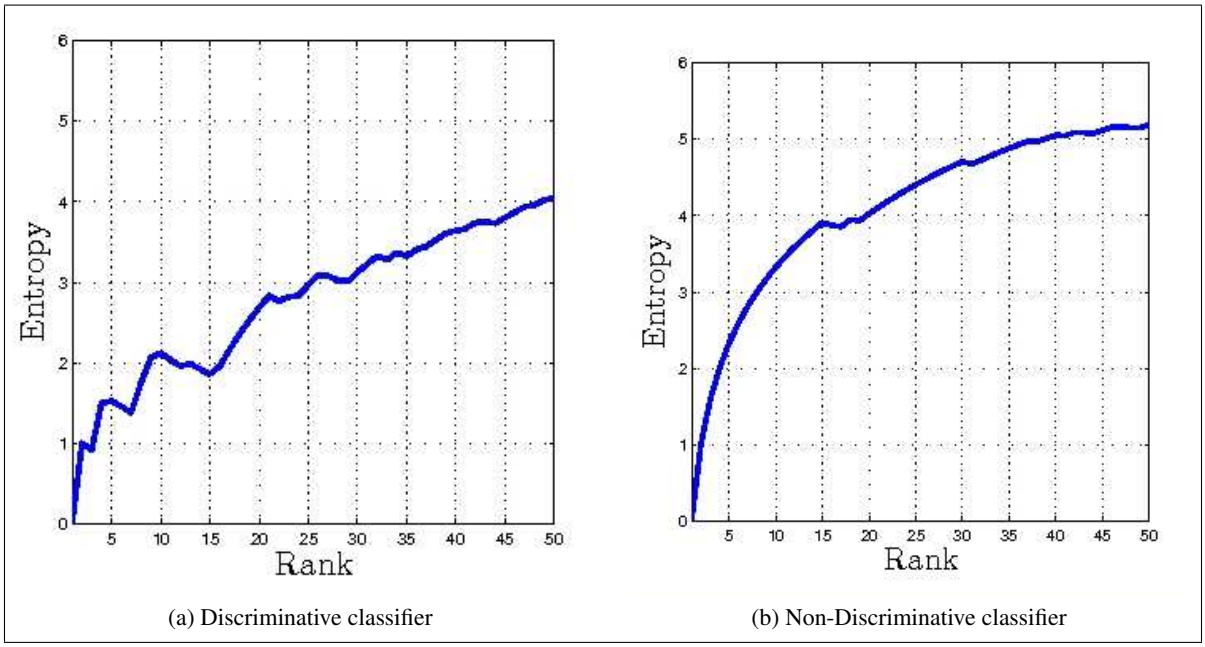
4.2.3 Selection: identifying distinctive parts

Our notion of a discriminative block is that it should occur in many of the images of the class from which it is learnt, but not in many images from other classes. However, it is not reasonable to assume that parts (represented by blocks) are so discriminative that they only occur in the class from which they are learnt. For example, the door of a washing machine will occur in the laundromat class, but can also occur in the kitchen or garage class. Similarly, a gothic arch can appear in both the church and cloister

Figure 4.4: **Mining of part instances.** The seed (initial) block is on the left. On the right, the additional example blocks added to the positive training set for retraining the part detector are shown in the order that they are added. Note that mining uses blocks selected from a certain scene category, but no other supervision is used.

class. However, one would not expect these parts to appear in many other of the indoor classes. In contrast, a featureless wall could occur in almost any of the classes.

In selecting the block classifiers we design a novel measure to capture this notion. The block classifiers were learnt on training images for a particular class, and they are tested as detectors on validation images of all classes. Blocks learned from a class are not required to be detected only from images of that class; instead, the milder constraint that the distribution of classes in which the block is detected should have low entropy is imposed. In this manner, distinctive but shareable mid-level parts can be selected. For the laundromat example above, we would expect the washing machine door to be detected in only a handful of the classes, so the entropy would be low. In contrast the block for a wall would be detected across many classes, so its distribution would be nearer uniform across classes, and hence the entropy higher.



(a) Discriminative classifier

(b) Non-Discriminative classifier

Figure 4.5: **Entropy-Rank (ER Curves).** Entropy values reach a uniform value as more and more images are ranked, with images from different classes coming in. Classifier (a) has low entropy at top ranks, which shows that it is picking the blocks from a few classes. On the other hand, Classifier (b) has more uniform entropy, which shows that it is picking the blocks from many classes, making the classifier less discriminative. For each class, classifiers with low AUC are selected.

To operationalise this requirement, each block is evaluated in a sliding-window manner on each validation image. Then, five block occurrences are extracted from each image by max-pooling in five image regions, corresponding to the spatial subdivisions used in the encoding of Section 4.3. Each block occurrence (z_i, y_i) detected in this manner receives a detection score z and a class label y equal to the label of the image. The blocks are sorted on their score z, and the top r ranking blocks selected. Then the entropy $H(Y|r)$ is computed:

$$H(Y|r) = -\sum_{y=1}^{N} p(y|r) \log_2 p(y|r), \tag{4.1}$$

where N is the number of image classes and $p(y|r)$ is the fraction of the top r blocks (z_i, y_i) that have label $y_i = y$. We introduce *Entropy-Rank Curves (ER curves)* to measure the entropy of a block classifier at different ranks. An ER curve is similar to a Precision-Recall Curve (PR curve), with rank on the x-axis and entropy values on the y-axis.

Figure 4.5 shows the ER curves of a discriminative and a non-discriminative block detector, respectively. Note, entropy for all part classifiers converges to a constant value (which depends on the class prior) as the rank increases. Analogously to Average Precision, we then take the Area Under Curve

(AUC) for the ER graph as an overall measure of performance of a classifier. The top scoring detectors based on this measure are then retained.

The final step is to remove redundant part detectors. In fact, there is no guarantee that the part mining procedure will not return the same or similar parts multiple times. The redundancy between a pair of detectors w' and w'' is measured by their cosine similarity $\langle w'/\|w'\|, w''/\|w''\| \rangle$ (recall that they are linear classifiers). For each class, n detectors are selected sequentially by increasing ER scores, skipping detectors that have cosine similarity larger than 0.5 with any of the detectors already selected.

4.3 Image representations and learning

4.3.1 Bag of Parts Representation

The part detectors developed in Section 4.2 are used to construct "bag of parts" image-level descriptors. In order to compute an image-level descriptor from the parts learned in Section 4.2, all the corresponding classifiers are evaluated densely at every image location at multiple scales. Part scores are then summarised in an image feature vector by using max-pooling, by retaining the maximum response score of a part in a region. The pooling is done at spatial-pyramid fashion [31] (1×1, 2×2 grids), and encodings of each spatial region are stacked together to form the final image representation of the image (the "bag of parts"). Note that the method of selecting the parts (Section 4.2.3) and here the encoding into an image feature vector both use max-pooling over the spatial-pyramid.

4.3.2 Bag of Words and Bag of Parts Combined representation

In the final experiment, we explore the complementarity of the Bag of Words and Bag of Parts representations. Bag of Words and Bag of Parts histograms are computed as described in Section 3.3 and Section 4.3.1 respectively. The two representations are combined by stacking the corresponding vectors after proper normalisation.

4.3.3 Learning

Learning uses the PEGASOS SVM [55] algorithm, a linear SVM solver. In order to use non-linear additive kernels instead of the linear one, the χ^2 explicit feature map of [68] is used (the bag of parts histograms are l^1 normalised). Using the feature map increases the dimension of the input feature vector by 3 times. The parameter C of the SVM (regularisation-loss trade-off) is determined by 4-fold cross validation. For multi-class image classification problems, 1-vs-rest classifiers are learned. In this case, it was found beneficial to calibrate the different 1-vs-rest scores by fitting a sigmoid [48] to them based on a validation set.

4.4 Experiments and results

4.4.1 Bag of Parts

The part-learning algorithm is evaluated on the task of scene classification on the MIT 67 indoor scene dataset of Quattoni and Torralba [49]. Note that, differently from object recognition datasets such as PASCAL VOC [15], in scene classification no geometric cue such as object bounding boxes is given to help initialising parts.

Evaluation uses the protocol of [49], using the same training and test split as [49] where each category has about 80 training images and 20 test images. In addition the training set is subdivided into about 64 train and 16 validation images. Performance is reported in terms of average classification accuracy as in [49] (*i.e*. the average of the diagonal of the confusion matrix) and, additionally, in term of mean Average Precision (mAP).

Method	Accuracy (%)	Mean AP (%)
ROI + Gist [49]	26.05	-
MM-scene [75]	28.00	-
CENTRIST [72]	36.90	-
Object Bank [34]	37.60	-
DPM [41]	30.40	-
RBoW [42]	37.93	-
LPR [51]	44.84	-
Patches [58]	38.10	-
Bag of Parts [3,350](Ours) (1×1 grid)	40.31	37.31
Bag of Parts [16,750](Ours) ($1 \times 1 + 2 \times 2$ grid)	46.10	43.55

Table 4.1: Average classification performance of single methods (previous publications and this paper). The dimension of the image representation is given in square brackets.

Method	Number of parts selected per class				
	10	20	30	40	50
Bag of Parts	42.34	44.81	44.96	46.00	46.10
LLC + Bag of Parts	56.66	55.98	55.93	56.01	55.94
IFV + Bag of Parts	62.80	62.75	62.65	62.02	63.10

Table 4.2: **Variation with number of part classifiers.** The table reports the variation of classification accuracy with number of part classifiers selected per class.

Blocks are learned as described in Section 4.2. The 31-dimensional cell HOG variant of [21] is used in all the experiments. For the seeding described in Section 4.2.1, the segmentation algorithm [20] is run with parameters $k = 0.5$, $\sigma = 200$, and $\min = 20$. The average number of part candidates obtained for each class is 3,800. For each of these seed blocks, a classifier is learned by following the expansion procedure of Section 4.2.2. We sample about 620,000 hog blocks randomly from the training set, and compute the mean (μ_0) and covariance (Σ) of this set. Since Σ comes out to be low-rank and non-invertible, a regulariser ($\lambda = 0.01$) is added to the diagonal elements of the covariance matrix.

Once the parts have been learned as described in Section 4.2.2 and selected as in Section 4.2.3, the bag of parts representation is extracted from each training image as described in Section 4.3. Finally, 67 one-vs-rest SVMs are learned from the training images, and the resulting scene classifiers are evaluated on the test data. As one can expect, the classification accuracy increases as more parts are added to the representation (Table 4.2), but the peak is at around 50 parts per category. The probable reason is a lack of training material (after all the parts and classifiers are learned on the same data) that causes overfitting. To overcome this, we left-right flip the images in the positive training set, and add them as additional positives. We also show the confusion matrix obtained with Bag of Parts method (Figure 4.6).The brighter spots in the confusion matrix shows the confusion between different classes, mainly because of the inter-class similarity and semantic overlap between classes (Section 1.2).

Top Detections on Validation Set

Figure 4.7 shows examples from the validation set images, with high detection score for a part classifier learnt on different classes class. Note, only one detection per image is shown. Along with the detections, we also show:

- the seed block (from the training set image) used for learning the part classifier. The seed block is selected using the unsupervised method described in Section 4.2.1.
- the final learnt HOG template of the part classifier. The final classifier is obtained after 10 rounds of retraining.

These results show that the part classifiers capture semantically meaningful parts/objects which are characteristic features of the respective classes.

Overall, the proposed part learning method compares very favourably with the previous state-of-the-art on part detection [58]. First, the parts found by our algorithm are much more informative. In particular, our accuracy on MIT Scene 67 is 46.10% when 50 parts per category are used. By comparison, the accuracy of [58] is 38.10%, and they use 210 parts per category. So our parts improve the accuracy by 8% using a quarter of the detectors.

Second, our part learning method is significantly more efficient than the discriminative clustering approach of [58] for three reasons. (i) [58] initialise their clustering algorithm by standard (generative) K-means, which, as they note, performs badly on the part clustering task; our exemplar SVM approach avoids that problem. (ii) These clusters are formed on top of a random selection of initial patches;

we found that aligning seed patches to superpixels substantially increases the likelihood of capturing interesting image structures (compared to random sampling). (iii) They use *iterative hard-mining* to learn their SVM models. This approach was tested in our context and found to be 60 times slower than LDA training that avoids this step.

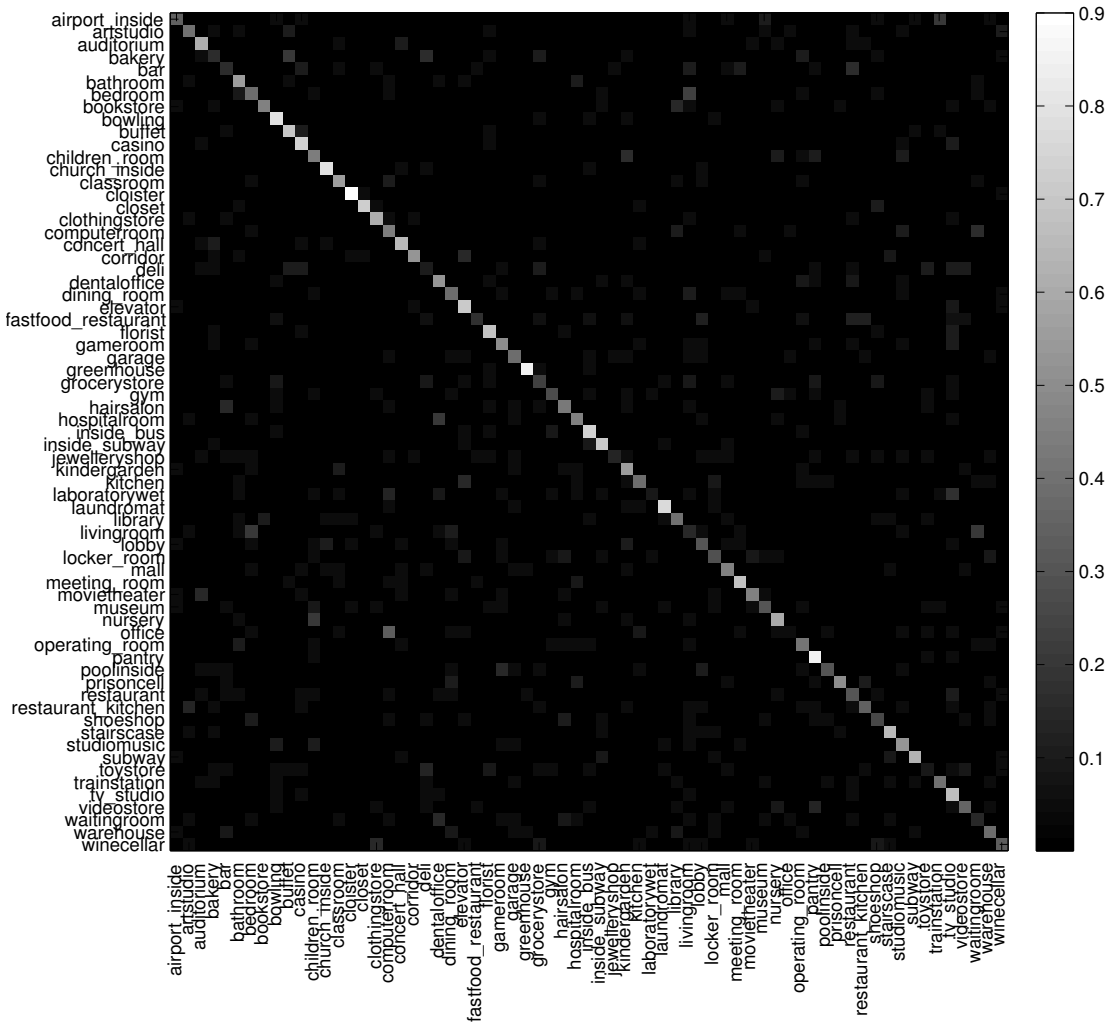


Figure 4.6: **Confusion matrix obtained with Bag of Parts.** The diagonal represent the correct classifications, and the brighter non-diagonal spots denote the confusion between two classes. The average of the diagonal of the confusion matrix gives the classification accuracy.

(a) closet

(b) computerroom

(c) florist

(d) corridor

Figure 4.7: Seed blocks and the learnt HOG templates, and detections on the validation set images.

4.4.2 Bag of Words and Bag of Parts

In the final experiment, the bag of parts and bag of words representations are combined as described in Section 4.3 and 67 one-vs-rest classifiers are learned as detailed above. We pick 50 parts per class and compute the bag of parts representation. Table 4.3 reports the overall performance of the combined descriptors and compares it favourably to BoW, and hence to all previously published results. Figure 4.8 shows qualitative results obtained by the combined bag of parts and bag of words method. Table 4.4 reports the per-class classification accuracy for the Bag of Parts model, and the combined Bag of Parts and Bag of Words model.

Method	Acc. (%)	Mean AP (%)
DPM+Gist-color+SP [41]	43.10	-
Patches+GIST+SP+DPM [58]	49.40	-
LLC + Bag of Parts (Ours)	56.66	55.13
IFV + Bag of Parts (Ours)	63.10	63.18

Table 4.3: Average classification performance of combination of methods (previous publications and this paper).

Class	BoP	BoP + LLC	BoP + IFV	Class	BoP	BoP + LLC	BoP + IFV
airport_inside	35.00	40.00	50.00	inside_subway	71.43	76.19	80.95
artstudio	40.00	35.00	45.00	jewelleryshop	13.64	40.91	36.36
auditorium	61.11	66.67	72.22	kindergarden	55.00	80.00	75.00
bakery	15.79	26.32	36.84	kitchen	38.10	52.38	61.90
bar	16.67	38.89	55.56	laboratorywet	9.09	45.45	54.55
bathroom	55.56	55.56	66.67	laundromat	77.27	72.73	81.82
bedroom	38.10	47.62	47.62	library	40.00	45.00	45.00
bookstore	45.00	50.00	65.00	livingroom	15.00	30.00	40.00
bowling	80.00	95.00	90.00	lobby	30.00	40.00	45.00
buffet	70.00	75.00	65.00	locker_room	28.57	52.38	61.90
casino	73.68	63.16	84.21	mall	45.00	45.00	45.00
children_room	44.44	38.89	38.89	meeting_room	68.18	59.09	63.64
church_inside	78.95	68.42	73.68	movietheater	45.00	55.00	60.00
classroom	55.56	61.11	66.67	museum	30.43	39.13	43.48
cloister	90.00	95.00	95.00	nursery	60.00	70.00	80.00
closet	72.22	72.22	72.22	office	4.76	4.76	23.81
clothingstore	61.11	61.11	55.56	operating_room	42.11	52.63	42.11
computerroom	44.44	77.78	83.33	pantry	85.00	60.00	70.00
concert_hall	65.00	65.00	95.00	poolinside	30.00	45.00	65.00
corridor	52.38	66.67	57.14	prisoncell	50.00	65.00	70.00
deli	10.53	36.84	21.05	restaurant	30.00	40.00	60.00
dentaloffice	52.38	61.90	71.43	restaurant_kitchen	34.78	56.52	69.57
dining_room	38.89	44.44	55.56	shoeshop	26.32	26.32	47.37
elevator	71.43	90.48	90.48	stairscase	65.00	70.00	75.00
fastfood_restaurant	17.65	64.71	64.71	studiomusic	52.63	84.21	89.47
florist	68.42	78.95	84.21	subway	61.90	57.14	61.90
gameroom	50.00	45.00	55.00	toystore	9.09	27.27	50.00
garage	38.89	55.56	61.11	trainstation	40.00	80.00	85.00
greenhouse	85.00	80.00	85.00	tv_studio	66.67	66.67	61.11
grocerystore	23.81	61.90	57.14	videostore	36.36	50.00	50.00
gym	27.78	50.00	66.67	waitingroom	14.29	28.57	33.33
hairsalon	42.86	52.38	66.67	warehouse	38.10	57.14	57.14
hospitalroom	45.00	65.00	85.00	winecellar	33.33	57.14	71.43
inside_bus	73.91	78.26	91.30	**Average**	**46.10**	**56.66**	**63.10**

Table 4.4: Per-class classification accuracies for Bag of Parts (BoP) and IFV + BoP (Combined). All results in %.

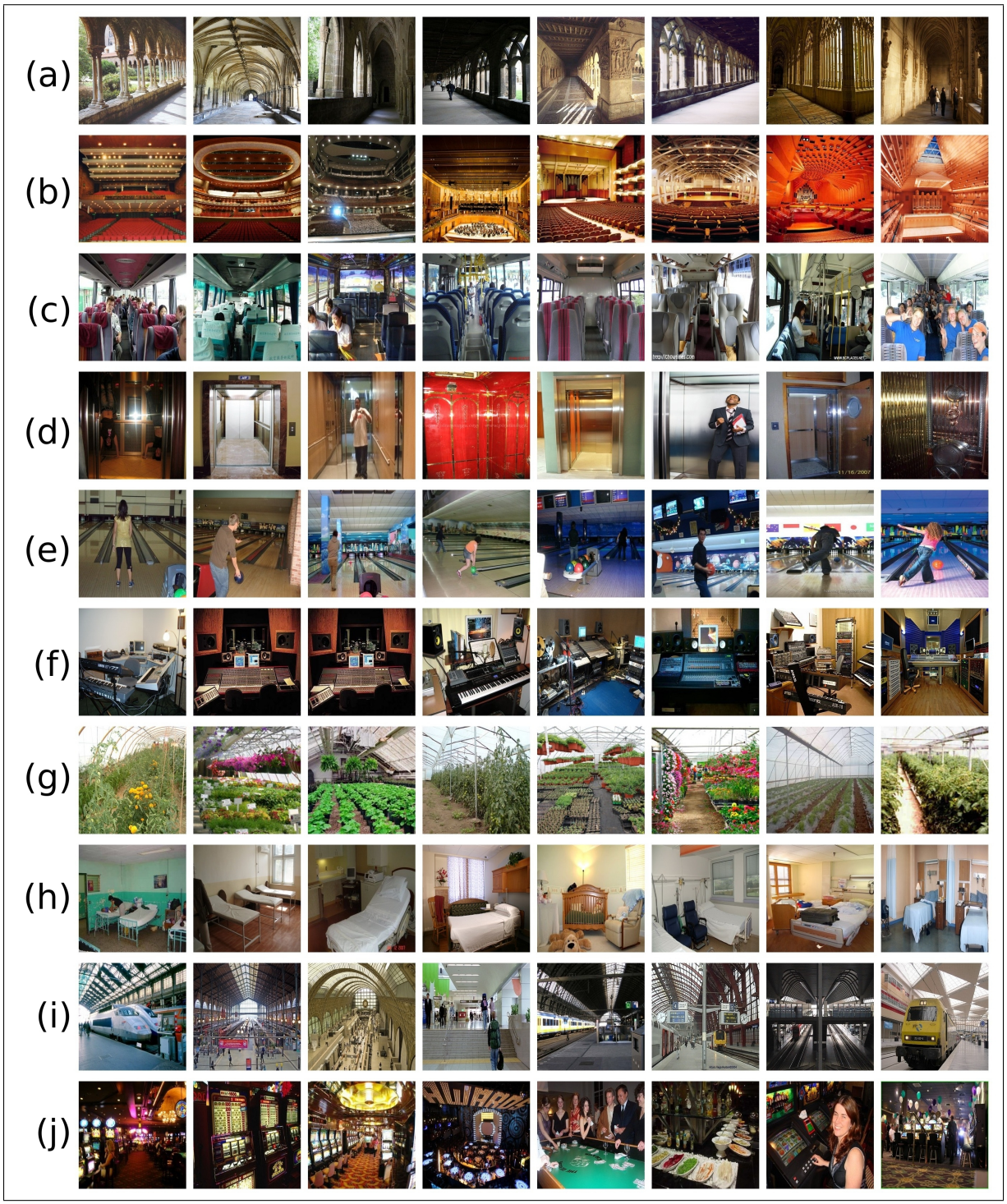


Figure 4.8: Categories with the highest classification rate (Bag of Words + Bag of Parts combined method). Each row shows the top eight results for the category. (a) cloister, (b) concert hall, (c) inside bus, (d) elevator, (e) bowling, (f) studiomusic, (g) greenhouse, (h) hospitalroom, (i) trainstation, and (j) casino.

4.5 Summary

We have presented a novel method to learn distinctive parts of objects or scenes automatically, from image-level category labels. The key problem of simultaneously learning a part model and detecting its occurrences in the training data was solved by paced learning by Exemplar SVMs, growing a model from just one occurrence of the part. The distinctiveness of parts was measured by the new concept of entropy-rank, capturing the idea that parts are at the same time predictive of certain object categories but shareable between different categories. The learned parts have been shown to perform very well on the task of scene classification, where they improved a very solid bag of words baseline or Fisher Vector baseline that in itself establishes the new state-of-the-art on the MIT Scene 67 benchmark.

The outcome of this work are blocks that correspond to semantically meaningful parts/objects. This mid-level representation is useful for other tasks, for example to initialize the region models of [42], and yields more understandable and diagnosable models than the original bag of visual words method.

Chapter 5

Conclusions and Future Work

In this thesis, we have explored the problem of Scene Classification and presented the solutions to solve the problem. We started with investigating the standard Bag of Words pipeline for image classification, and built a very fast and efficient pipeline for large scale semantic video retrieval. We have demonstrated this on the Semantic Indexing Task (SIN), TRECVID 2010. Out of all the submissions to this challenge, our results were ranked the best results for *Nighttime*, and second best for *Hand, Airplane flying*.

Motivated by this, we applied the Bag of Words pipeline for Indoor Scene Classification. We investigate the recent improvements in the feature descriptors (*e.g*. RootSIFT), encoding methods (*e.g*. LLC, Fisher Vector) and use them to build our Bag of Words method for Indoor Scene Classification. We show that our method achieves the state-of-the-art results on the MIT 67 indoor scene dataset, beating all the previous results. This is an interesting result because our method which uses only a single feature channel is able to outperform the methods which use multiple feature channels.

We also proposed a new method for partitioning of images, which is an extension to the standard Spatial Pyramid Technique (SPM). The new partitioning is designed for scene classification tasks, where a non-uniform partitioning based on the different regions is more useful than the uniform partitioning.

We also observed that indoor scenes are characterized by some distinctive elements present in the scene. These high level objects/elements are not captured by the Bag of Words representation. An image representation which takes into account the distinctive parts from the scenes, should help in classification. We proposed a new representation, called Bag of Parts which can discover parts automatically and with very little supervision. We have also introduced a novel measure of entropy-rank to measure the distinctiveness of parts. We have shown that such Bag of Parts representation is able to capture the discriminative parts from the scenes, and achieves good classification results. The outcome of this work are blocks that correspond to semantically meaningful parts/objects. This mid-level representation yields more understandable and diagnosable models than the original bag of visual words method.

Finally, we have also shown that the Bag of Parts representation is complementary to the Bag of Words representation, and combining the two gives a boost to the classification performance. The combined representation establishes a new state-of-the-art benchmark on the MIT 67 indoor scene dataset.

5.1 Future Work

We list down some of the possible extensions of the work presented in this thesis:

- **Fisher encoding of Bag of Parts.**

  In our current method, the Bag of Parts representation is encoded by taking the maximum detection score of each part over an image. It would be interesting to try Fisher Vector encoding to encode the Bag of Parts representation.

- **Faster learning of Parts.**

  Our part learning algorithm is fast, primarily because we avoid hard-negative mining and the LDA technique of [27]. However there are many seed blocks to start with, therefore the total time to learn all the parts is high. One possible option to reduce the overall time could be to reduce the number of seed blocks further.

- **Better methods to remove redundant parts.**

  Redundant part detectors are currently removed in a greedy method. The redundancy between a pair of detectors is measured by their cosine similarity. For each class, n detectors are selected sequentially by increasing ER scores, skipping detectors that have cosine similarity larger than 0.5 with any of the detectors already selected. This selection procedure can be improved by selecting the detectors as a global optimal solution.

Related Publications

- *Mayank Juneja*, Andrea Vedaldi, C.V. Jawahar, Andrew Zisserman
  "Blocks that Shout: Distinctive Parts for Scene Classification"
  in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition* (CVPR), Portland, USA 2013.

Bibliography

[1] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. 2012.

[2] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9, 1997.

[3] Y. Amit and A. Trouvé. Generative models for labeling multi-object configurations in images. In *Toward Category-Level Object Recognition*. Springer, 2006.

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

[5] D. Arthur and S. Vassilvitskii. `k-means++`: The advantages of careful seeding. In *Proc. ACM-SIAM Symp. on Discrete Algorithms*, 2007.

[6] A. Bosch, A. Zisserman, and X. Muñoz. Scene classification via pLSA. In *Proc. ECCV*, 2006.

[7] A. Bosch, A. Zisserman, and X. Muñoz. Image classification using random forests and ferns. In *Proc. ICCV*, 2007.

[8] B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[9] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3D human pose annotations. In *Proc. ICCV*, 2009.

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

[11] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[12] G. Csurka, C. Bray, C. Dance, and L. Fan. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

[13] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. CVPR*, 2005.

[14] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) challenge. *IJCV*, 88(2):303–338, June 2010.

[15] M. Everingham, A. Zisserman, C. Williams, and L. V. Gool. The PASCAL visual obiect classes challenge 2007 (VOC2007) results. Technical report, Pascal Challenge, 2007.

[16] Facebook. `http://www.facebook.com`.

[17] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPR Workshop*, 2004.

[18] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 524–531 vol. 2, 2005.

[19] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *PAMI*, 2009.
[20] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2), 2004.
[21] P. F. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Proc. CVPR*, 2008.
[22] M. A. Fischler and R. A. Elschlager. The representation and matching of pictorial structures. In *IEEE Trans. on Computers*, 1973.
[23] Flickr. `http://www.flickr.com`.
[24] V. Garg, S. Chandra, and C. V. Jawahar. Sparse discriminative fisher vectors in visual classification. In *Proceedings of the Eighth Indian Conference on Computer Vision, Graphics and Image Processing*, ICVGIP '12, pages 55:1–55:8, New York, NY, USA, 2012. ACM.
[25] M. Gharbi, T. Malisiewicz, S. Paris, and F. Durand. A gaussian approximation of feature space for fast image similarity. Technical Report 2012-032, MIT CSAIL, 2012.
[26] V. N. Hakan Bilen and L. V. Gool. Object and action classification with latent variables. In *Proceedings of the British Machine Vision Conference*, pages 17.1–17.11. BMVA Press, 2011. http://dx.doi.org/10.5244/C.25.17.
[27] B. Hariharan, J. Malik, and D. Ramanan. Discriminative decorrelation for clustering and classification. In *Proc. ECCV*, 2012.
[28] Instagram. `http://www.instagram.com`.
[29] B. Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97, 1981.
[30] R. Kwitt, N. Vasconcelos, and N. Rasiwasia. Scene recognition on the semantic manifold. In *Proceedings of the 12th European conference on Computer Vision - Volume Part IV*, ECCV'12, pages 359–372, Berlin, Heidelberg, 2012. Springer-Verlag.
[31] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bag of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. CVPR*, 2006.
[32] D. Lee. `http://people.cs.umass.edu/~elm/Teaching/ppt/SIFT.pdf`.
[33] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *IJCV*, 43(1):29–44, June 2001.
[34] L.-J. Li, H. Su, E. Xing, and L. Fei-Fei. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *Proc. NIPS*, 2010.
[35] J. Liu and M. Shah. Scene modeling using co-clustering. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–7, 2007.
[36] D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. ICCV*, 1999.
[37] J. Luo and A. Savakis. Indoor vs outdoor classification of consumer photographs using low-level and semantic features. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 2, pages 745–748. IEEE, 2001.
[38] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *Proc. ICCV*, 2011.

[39] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001.

[40] P. Over, G. Awad, J. G. Fiscus, B. Antonishek, M. Michel, W. Kraaij, A. F. Smeaton, and G. Quénot. Trecvid 2010 - an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *TRECVID*, 2010.

[41] M. Pandey and S. Lazebnik. Scene recognition and weakly supervised object localization with deformable part-based models. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1307–1314. IEEE, 2011.

[42] S. Parizi, J. Oberlin, and P. Felzenszwalb. Reconfigurable models for scene recognition. CVPR, 2012.

[43] O. Parkhi, A. Vedaldi, C. V. Jawahar, and A. Zisserman. The truth about cats and dogs. In *Proc. ICCV*, 2011.

[44] A. Payne and S. Singh. Indoor vs. outdoor scene classification in digital photographs. *Pattern Recognition*, 38(10):1533 – 1545, 2005.

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

[46] J. Pfanzagl. Asymptotic expansions in parametric statistical theory. *Developments in statistics*, 3:1–97, 1980.

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

[48] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*. Cambridge, 2000.

[49] A. Quattoni and A. Torralba. Recognizing indoor scenes. In *Proc. CVPR*, 2009.

[50] M. Raptis, I. Kokkinos, and S. Soatto. Discovering discriminative action parts from mid-level video representations. In *Proc. CVPR*, 2012.

[51] F. Sadeghi and M. F. Tappen. Latent pyramidal regions for recognizing scenes. In *Proc. ECCV*, 2012.

[52] G. Salton and M. McGill. Introduction to moderm information retrieval, 1983.

[53] N. Serrano, A. Savakis, and J. Luo. A computationally efficient approach to indoor/outdoor scene classification. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 4, pages 146–149 vol.4, 2002.

[54] N. Serrano, A. E. Savakis, and J. Luo. Improved scene classification using efficient low-level features and semantic cues. *Pattern Recognition*, 37(9):1773 – 1784, 2004.

[55] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th international conference on Machine learning*, pages 807–814. ACM, 2007.

[56] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-GrAdient SOlver for SVM. In *Proc. ICML*, 2007.

[57] G. Sharma and F. Jurie. Learning discriminative spatial representation for image classification. In *Proceedings of the British Machine Vision Conference*, pages 6.1–6.11. BMVA Press, 2011. http://dx.doi.org/10.5244/C.25.6.

[58] S. Singh, A. Gupta, and A. A. Efros. Unsupervised discovery of mid-level discriminative patches. In *Proc. ECCV*, 2012.

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

[60] M. Szummer and R. W. Picard. Indoor-outdoor image classification. In *Content-Based Access of Image and Video Database, 1998. Proceedings., 1998 IEEE International Workshop on*, pages 42–51. IEEE, 1998.

[61] S. Ullman, E. Sali, and M. Vidal-Naquet. A framgent-based approach to object representation and classification. In *Intl. Workshop on Visual Form*, 2001.

[62] A. Vailaya, M. Figueiredo, A. Jain, and H. Zhang. Content-based hierarchical classification of vacation images. In *Multimedia Computing and Systems, 1999. IEEE International Conference on*, volume 1, pages 518–523 vol.1, 1999.

[63] A. Vailaya, M. A. T. Figueiredo, A. Jain, and H.-J. Zhang. Image classification for content-based indexing. *Image Processing, IEEE Transactions on*, 10(1):117–130, 2001.

[64] K. E. A. van de Sande, J. R. R. Ujilings, T. Gevers, and A. W. M. Smeulders. Segmentation as selective search for object recognition. In *Proc. ICCV*, 2011.

[65] J. C. van Gemert, J.-M. Geusebroek, C. J. Veenman, and A. W. M. Smeulders. Kernel codebooks for scene categorization. In *Proc. ECCV*, 2008.

[66] A. Vedaldi and B. Fulkerson. VLFeat – An open and portable library of computer vision algorithms. In *Proc. ACM Int. Conf. on Multimedia*, 2010.

[67] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *Proc. ICCV*, 2009.

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

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

[70] L. Wang, Y. Li, J. Jia, J. Sun, D. Wipf, and J. Rehg. Learning sparse covariance patterns for natural scenes. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2767–2774, 2012.

[71] M. Weber, M. Welling, and P. Perona. Towards automatic discovery of object categories. In *Proc. CVPR*, volume 2, pages 101–108, 2000.

[72] J. Wu and J. Rehg. Centrist: A visual descriptor for scene categorization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(8):1489–1501, 2011.

[73] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3485–3492, 2010.

[74] Youtube. `http://www.youtube.com`.

[75] J. Zhu, L.-J. Li, L. Fei-Fei, and E. Xing. Large margin learning of upstream scene understanding models. In *Proc. NIPS*, 2010.