"Flobject" Analysis: Learning about Static Images from Motion

by

Patrick S. Li

A thesis submitted in conformity with the requirements
for the degree of Masters of Applied Science
Graduate Department of Electrical Engineering
University of Toronto

Abstract

"Flobject" Analysis: Learning about Static Images from Motion

Patrick S. Li

Masters of Applied Science

Graduate Department of Electrical Engineering

University of Toronto

2011

A critical practical problem in the field of object recognition is an insufficient number of labeled training images, as manually labeling images is a time consuming task. For this reason, unsupervised learning techniques are used to take advantage of *unlabeled* training images to extract image representations that are useful for classification. However, unsupervised learning is in general difficult. We propose simplifying the unsupervised training problem considerably by taking the advance of motion information. The output of our method is a model that can generate a vector representation from any static image. However, the model is trained using images with additional motion information. To demonstrate the flobject analysis framework, we extend the latent Dirichlet allocation model to account for word-specific flow vectors. We show that the static image representations extracted using our model achieve higher classification rates and better generalization than standard topic models, spatial pyramid matching, and Gist descriptors.

Acknowledgements

I only managed to complete this thesis because of the support and encouragement of my family and friends. I am especially appreciative of my parents for supporting me through every minute of the process. Thanks Baba for being willing to discuss every minute detail of my thesis with me, and Mama for being willing to *listen* to me discuss every detail with you. Thank you Luca and Emmy for always ensuring that my life was never swallowed by academics and that mathematics and programming were always appropriately balanced by video games and movie nights.

My friends in both Calgary and Toronto have been critical as my sources of inspiration and diversion alike. When research was going well, everyone was quick to tell me how *interesting* it all sounded. And when research was not going well, they were always ready for a camping trip or foosball game. I want to thank my friends Jeff and Lei in particular, off of whom many of my ideas were bounced, incessantly and mercilessly.

In Toronto, Brendan was, in many ways, the best supervisor I could have hoped for. He pushed me when necessary but also gave me the freedom to chase whatever idea I had on any given week. Above all, he put my intellectual development and curiosity ahead of everything else, and I am most grateful for that.

My colleagues in the PSI group were as equally supportive as Brendan. I started with no knowledge or experience in the field at all, and the senior students were always willing to answer whatever questions I had. Ryan, especially, has been a fountain of advice for myself in all matters ranging from MCMC to career advice for the academic fledgling.

Finally, I want to thank Inmar, with whom I did the majority of the work on this thesis, for being an amazingly capable researcher and considerate friend. From her, I learned all I know about affinity propagation, graphical models, the writing of Vonnegut, vegetarian cuisine, and Battlestar Galactica among too many other items to list. Thank you Inmar for your advice, your expertise, your patience with my rather loose adherence to the scientific method, and our many hours of interesting conversation.

Thank you everyone.

-Patrick

Contents

List of Tables

List of Figures

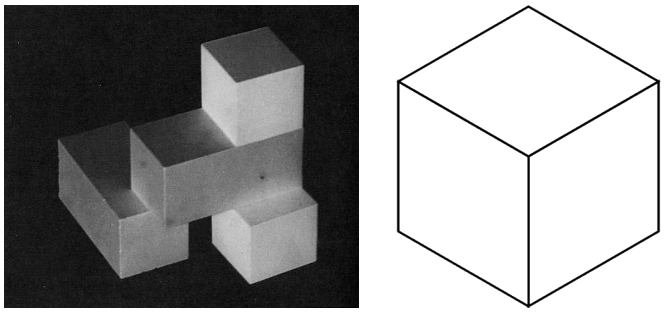
Chapter 1

Introduction

Object recognition is the task of recognizing what type of object is present in an image. For example, to recognize whether or not an image contains a car. This problem has an important application in enabling us to search over a collection of images for particular types of objects. Since the advent of digital photography and the explosion of digital media, this has become an increasingly desired function.

As of yet, there are no automated algorithms capable of reliably identifying several kinds of objects in images, despite how trivial the same task is for people. To simplify the problem, computer object recognition challenges assume that every image contains only a single object of interest, and that the object is an instance of one of a fixed number of known classes. The PASCAL Visual Object Challenge has, for example, twenty classes including dog, car, and motorbike classes, among others. Algorithms are given a collection of images as input, and are asked to classify each image as belonging to one of the twenty classes. Performance is measured by the percentage of images an algorithm classifies correctly.

Object recognition was first approached by attempting to carefully handcraft rules that describe the appearances of different kinds of objects. Blocks World [22] (see Figure 1.1a) was one of the earliest object recognition challenges, and consisted of images



(a) One image from Blocks World.

(b) One possible representation of a cube.

Figure 1.1: Objects in Blocks World were recognized by explicitly searching for lines and intersections.

containing only simple geometric shapes. Algorithms searched for objects by looking for particular arrangements of lines and intersections. A cube, for example, is composed of no more than nine lines as at least three edges are always hidden from view (Figure 1.1b). Such approaches demonstrated initial success but attempts to scale the approach to more complicated objects were unsuccessful. These failures revealed the intractability of handcrafting a set of rigid constraints for describing an object's appearance.

Subsequent progress in object recognition was only achieved via the rise of machine learning techniques in the field. In contrast to the traditional approach of handcrafting rules for classifying an image, machine learning algorithms are *trained* by being shown examples of images and being told the type of object within each. Because of their minimal reliance on manual tuning, machine learning techniques have excelled over the old handcrafted systems.

We refer to an image whose class is known as a *labeled* image, and the set of images dedicated to training a classifier as *training* images. A classifier predicts the class for a *test* image by analyzing its relation to the training images, perhaps using a model as a surrogate for the training images. Because the classifier is told what class each training

image belongs to, we refer to this approach as a *supervised* training framework. This is in constrast to an *unsupervised* training framework, where an algorithm is simply shown a collection of unlabeled images and told to make sense of them.

The limitation of supervised training is that without general knowledge about the contiguity of objects, how the appearance of an object changes with lighting, rotation and translation, how to differentiate between foreground and background, etc., an enormous number of labeled training images is required to attain good classification performance. This is a critical practical problem in object recognition as manually labeling images is extremely time consuming.

The addition of an unsupervised training stage before the supervised training stage is meant to work around the problem of the lack of labeled training images by enabling unlabeled images to also be used for training. Given a sufficient number of images, the unsupervised training stage is meant to learn about properties of objects that are common to all images. These include the previously mentioned traits, the relationship between appearance, lighting, rotation, and translation, and how to differentiate between foreground and background. Knowledge of these common traits of images simplifies the supervised training stage considerably, enabling good classification performance to be achieved with a smaller number of labeled training images.

In practice, however, unsupervised training is difficult and current algorithms for unsupervised training have been unsuccessful in learning about more than basic image properties. For example, there exists no algorithm yet that has learned to differentiate between foreground and background from a set of unlabeled images at a level comparable to human beings.

The focus of this work is on taking advantage of motion information to simplify the unsupervised training stage. We refer to this framework of using motion information to help classify *static* images as "flobject" analysis. There are many algorithms and models that may be used for flobject analysis. This thesis presents one model, an extension of

latent Dirichlet allocation (LDA), and demonstrates that the additional use of motion information during unsupervised training boosts classification accuracy significantly.

Thesis Overview

This thesis is written with two main objectives in mind. One objective is to give an overview of object recognition research and to familiarize the reader with the many parts and pieces that make up an object recognition system. The other is to describe and promote the flobject analysis framework and show how it fits within a larger object recognition system. To this end, this thesis is organized in the following manner:

Chapter 2 will formally define the object recognition problem and establish a consistent notation. It will also define supervised and unsupervised learning and their relations specifically in the context of object recognition. Popular algorithms for supervised and unsupervised learning will be briefly reviewed to prepare for their use in later chapters.

Chapter 3 will give an example of a simple but complete object recognition system. This will solidify the concepts introduced in Chapter 2 as well as describe the fundamental dataflow pipeline that is shared by almost all modern object recognition systems. We will also describe and make use of some techniques that are commonly used by the object recognition community.

Chapter 4 will describe the flobject analysis framework and the flow LDA (FLDA) model. The motivation and properties of the model will be discussed in detail, and we will give an inference method for FLDA based on collapsed Gibbs sampling.

Chapter 5 will detail the experimental setup as well as the CityCars and CityPedestrians classification datasets. We will describe how the output of FLDA is used to create FLDA descriptors, as well as the creation of several other descriptors that we compare against in Chapter 6.

Chapter 6 will show experimental results of applying FLDA to classification on the

CityCars and CityPedestrians datasets. FLDA descriptors are compared to other state-of-the-art descriptors for classification on the CityCars dataset. We will also describe how to explicitly account for spatial locality of objects by creating hierarchical FLDA descriptors (H-FLDA) and applying them towards classification on the CityPedestrians dataset.

Chapter 7 will discuss the existing literature that is most closely related to the work in this thesis. We will touch on the relation between the work on motion analysis and flobject analysis, and how existing models for motion analysis may possibly be applied to flobject analysis.

Chapter 2

The Object Recognition Problem

2.1 Formalizing the Object Recognition Problem

Formally the object recognition problem is a classification problem where inputs are images, and output labels correspond to different classes of objects. We will assume for simplicity that all images contain three color channels, are all of the same width and height, W and H, and that the number of object classes, C, is known. We will further assume that all images contain only one out of the C possible objects, and consequently can be categorized neatly as belonging to a single object class. A classifier, f, is defined to be a function that takes an image, $\vec{x} \in \mathbb{R}^{W \times H \times 3}$, as input and returns a label, $l \in \{1, \ldots, C\}$, indicating the class of the object within the image.

$$\begin{aligned} \text{Given: } & \text{image, } \vec{x} \in \mathbb{R}^{W \times H \times 3}, \\ \text{Predict: } & l = f(\vec{x}) \\ & \text{where } f \text{ is a classifier,} \\ & \quad l \in \{1, \ldots, C\} \text{ is the predicted label for the image.} \end{aligned} \tag{2.1}$$

We assume that every image and its corresponding label comes from an underlying

distribution $p(X = \vec{x}, L = l)$, and our objective is to find a classifier that maximizes the probability of correctly predicting the label for an input image.

$$\text{Find: } \underset{f}{\operatorname{argmax}}\ P(f(\vec{x}) = l) \tag{2.2}$$
$$\text{where } (\vec{x}, l) \sim p(X = \vec{x}, L = l)$$

In practice, we do not have access to the underlying distribution $p(X = \vec{x}, L = l)$. Instead we have a set of samples, $(\vec{x}_1, l_1) \ldots (\vec{x}_N, l_N)$, independently drawn from $p(X = \vec{x}, L = l)$, which we use to find our classifier.

There is the issue that our final classifier might perform well on the limited number of samples we have access to, but poorly on the actual distribution $p(X = \vec{x}, L = l)$. When this happens, we say that our classifier is *overfit* to our data. There are a number of techniques to avoid overfitting, such as cross validation and regularization, but it remains an open problem in model selection. Readers are invited to read [1] for an introduction to the issue.

The approach that will be used in this thesis will be to use random test/train partitions to search over a class of functions. We randomly select half of the data as training data, and use the remaining half as test data. We will define a class of classifiers $\mathcal{F}$, and find the optimal classifier in this class, $f \in \mathcal{F}$, that performs best on the training data. The performance of this classifier, f, is then evaluated on the test data.

If our classifier, f, truely performs well on the underlying distribution $p(X = \vec{x}, L = l)$, then it should have high performance on both the training and test data. However, if f performs well on the training data, but poorly on the test data, then that is evidence that our class of classifiers $\mathcal{F}$ is too rich for the amount of data we have and suffers from overfitting issues.

2.2 Mapping Images to Descriptors

State-of-the-art classifiers do not classify images based on looking directly at the raw color levels of each pixel in the image. They instead first extract higher level image features, such as lines and intersections analogous to the features used in Blocks World, and classify an image based on those. In such cases, it is useful to interpret the classification process as being composed of two stages. In the first stage, higher level image features are extracted from a test image and represented as a *descriptor,* $\vec{d}$, using $\phi(\vec{x})$. In the second stage, we classify the image based on this descriptor.

$$\begin{aligned} &\text{Let } \vec{d} = \phi(\vec{x}), \\ &\text{Predict: } l = f(\vec{d}). \end{aligned} \tag{2.3}$$

We will refer to the space in which descriptors live as *descriptor space*, and the original space in which images live ($\mathbb{R}^{W \times H \times 3}$) as *image space*. The objective of $\phi(\vec{x})$ is to map images to descriptor space, where the separating boundaries between classes are hopefully simpler than in the original image space.

As an example, consider the task of classifying images as one of two categories, images of office scenes ($l = 0$), and images of sunny days ($l = 1$). A classifier that looked only at the percentage of blue pixels in an image might perform very well. In this case,

$$\begin{aligned} &\text{Let } d = \phi(\vec{x}) = \text{ percentage of blue pixels in } \vec{x} \\ &\text{Predict: } l = f(d) = \begin{cases} 1 & \text{if } d > 50\% \\ 0 & \text{otherwise,} \end{cases} \end{aligned} \tag{2.4}$$

where the artificial threshold of 50% was manually chosen, perhaps by a domain expert with an intuition developed from surveying photos of sunny days while in his office.

2.3 Supervised Learning Framework for Object Recognition

As mentioned before, machine learning algorithms defer from traditional approaches in that they are *trained* on a collection of labeled images. In concrete terms, this means that the classifier is provided with a set of M training images, $\vec{x}_1^{\text{train}} \dots \vec{x}_M^{\text{train}}$, and their corresponding labels, $l_1^{\text{train}} \dots l_M^{\text{train}}$, in order to predict the label for a test image.

$$\begin{aligned} \text{Given: } & \text{training images, } \vec{x}_1^{\text{train}} \dots \vec{x}_M^{\text{train}}, \\ & \text{training labels, } l_1^{\text{train}} \dots l_M^{\text{train}}, \\ & \text{test image, } \vec{x}, \\ \text{Let } & \vec{d}, \vec{d}_1^{\text{train}} \dots \vec{d}_M^{\text{train}} = \phi(\vec{x}), \phi(\vec{x}_1^{\text{train}}) \dots \phi(\vec{x}_M^{\text{train}}), \\ \text{Predict: } & l = f(\vec{d}; \vec{d}_1^{\text{train}} \dots \vec{d}_M^{\text{train}}, l_1^{\text{train}} \dots l_M^{\text{train}}). \end{aligned} \tag{2.5}$$

As a trivial example of supervised learning, we may continue to develop the office/sunny day classifier to take advantage of a set of labeled training images in order to avoid having to manually set the threshold of 50%. Instead we will assume that images of sunny days contain a very high percentage of blue pixels relative to images of offices. Therefore we set the threshold, τ, to the minimum percentage of blue pixels found amongst the images of sunny days and accept the possibility of our classifier failing for offices decorated with blue wallpaper.

$$
\begin{aligned}
\text{Given: } & \text{training images, } \vec{x}_1^{\text{train}} \dots \vec{x}_M^{\text{train}}, \\
& \text{training labels, } l_1^{\text{train}} \dots l_M^{\text{train}}, \\
& \text{test image, } \vec{x}, \\
\text{Let } & d, d_1^{\text{train}} \dots d_M^{\text{train}} = \phi(\vec{x}), \phi(\vec{x}_1^{\text{train}}) \dots \phi(\vec{x}_M^{\text{train}}), \\
& \tau = \min\{d_i^{\text{train}} | l_i = 1\} \\
\text{Predict: } & l = f(d; d_1^{\text{train}} \dots d_M^{\text{train}}, l_1^{\text{train}} \dots l_M^{\text{train}}) \\
& = \begin{cases} 1 & \text{if } d > \tau \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}
\tag{2.6}
$$

2.4 Adding Unsupervised Learning to Object Recognition

In the supervised learning framework we introduced the concept of mapping images to descriptors, enabling us to train a classifier in the simpler descriptor space rather than in image space. As a motivating example we discussed the simple case of distinguishing office scenes from sunny days by describing images by the percentage of blue pixels in the image. For other more complicated applications, it is not obvious what mapping function, $\phi(\vec{x})$, is appropriate for mapping images to a space with simple class boundaries.

The supervised learning framework avoided the problem of having to handcraft a set of decision rules by providing a set of labeled training images on which to train a classifier. In a similar fashion, the unsupervised learning framework avoids the problem of having to handcraft a mapping function by providing a set of training images on which to train the mapping function. It is not necessary to provide labels for the training images to the unsupervised stage as it is not concerned with classification.

Thus the mapping function is now a function that takes as input an image, $\vec{x}$, as well

as an additional set of K training images, $\vec{x}_1^{\text{unspv}} \ldots \vec{x}_K^{\text{unspv}}$ ("unspv" being an abbreviation for "unsupervised"), and returns a descriptor, $\vec{d}$, which describes the image in a useful way.

$$\begin{gathered} \text{Let } \vec{d} = \phi(\vec{x}; \vec{x}_1^{\text{unspv}} \ldots \vec{x}_K^{\text{unspv}}) \\ \text{where } \vec{x}_1^{\text{unspv}} \ldots \vec{x}_K^{\text{unspv}} \text{ are training images,} \\ \vec{x} \text{ is an image,} \\ \vec{d} \text{ is a descriptor for } \vec{x}. \end{gathered} \tag{2.7}$$

Note that the set of training images used during the unsupervised stage, $\vec{x}_1^{\text{unspv}} \ldots \vec{x}_K^{\text{unspv}}$, need not be distinct from the set of training images used during the supervised stage, $\vec{x}_1^{\text{train}} \ldots \vec{x}_M^{\text{train}}$. I.e., the images used for supervised training may also be used during unsupervised training.

We will now further develop our office/sunny day classifier by adding an unsupervised learning stage and modifying the mapping function, $\phi(\vec{x})$, to take advantage of training images. We originally defined $\phi(\vec{x})$ to return the percentage of blue pixels in an image. But this scheme requires that the sky be actually blue in all of the sunny day images and that office images contain very little blue. Our classifier would fail if the camera had a sepia tone filter equipped on the lens and the sky in all the images were instead a light reddish color. We will make our classifier robust to such global color tints by assuming that the color of the sky, c, is the most common color amongst our training images. Our mapping function, $\phi(\vec{x})$, will be modified to return the percentage of pixels of color c in an image.

$$\begin{aligned} &\text{Given: unsupervised training images, } \vec{x}_1^{\text{unspv}} \ldots \vec{x}_K^{\text{unspv}}, \\ &\text{Let } d = \phi(\vec{x}; \vec{x}_1^{\text{unspv}} \ldots \vec{x}_K^{\text{unspv}}) \\ &\quad = \text{ percentage of pixels of color } c \text{ in } \vec{x}, \\ &\text{where } c = \text{ most common color in } \vec{x}_1^{\text{unspv}} \ldots \vec{x}_K^{\text{unspv}}. \end{aligned} \tag{2.8}$$

Our complete office/sunny day classifier now contains an unsupervised training stage which infers the most common color, c, from unlabeled training images and then describes each of the labeled training images in terms of the percentage of pixels of color c within each image. The threshold that controls whether an image is of an office or a sunny day is inferred from the set of percentages extracted from the training images. Finally a test image is classified by computing the percentage of pixels of color c within the image and checking whether that is above or below the threshold. The following algorithm details the complete training procedure for our office/sunny day classifier.

$$
\begin{aligned}
\text{Given: } & \text{unsupervised training images, } \vec{x}_1^{\text{unspv}} \dots \vec{x}_K^{\text{unspv}}, \\
& \text{labeled training images, } \vec{x}_1^{\text{train}} \dots \vec{x}_M^{\text{train}}, \\
& \text{with labels, } l_1^{\text{train}} \dots l_M^{\text{train}}, \\
& \text{test image, } \vec{x}, \\
\text{Let } & c = \text{most common color in } \vec{x}_1^{\text{unspv}} \dots \vec{x}_K^{\text{unspv}}, \\
& d = \phi(\vec{x}; \vec{x}_1^{\text{unspv}} \dots \vec{x}_K^{\text{unspv}}), \\
& d_1^{\text{train}} = \phi(\vec{x}_1^{\text{train}}; \vec{x}_1^{\text{unspv}} \dots \vec{x}_K^{\text{unspv}}), \\
& \vdots \qquad\qquad\qquad \vdots \\
& d_M^{\text{train}} = \phi(\vec{x}_M^{\text{train}}; \vec{x}_1^{\text{unspv}} \dots \vec{x}_K^{\text{unspv}}), \\
& \tau = \min\{d_i^{\text{train}} | l_i = 1\}, \\
\text{where } & \phi(\vec{x}; \vec{x}_1^{\text{unspv}} \dots \vec{x}_K^{\text{unspv}}) = \text{percentage of pixels of color } c \text{ in } \vec{x}, \\
\text{Predict: } & l = f(d; d_1^{\text{train}} \dots d_M^{\text{train}}, l_1^{\text{train}} \dots l_M^{\text{train}}) \\
& = \begin{cases} 1 & \text{if } d > \tau \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
\tag{2.9}
$$

2.5 Popular Machine Learning Algorithms

This section will briefly describe some popular machine learning algorithms for supervised and unsupervised learning to prepare for their later use. This listing is by no means comprehensive, and readers are refered to [1] for a more in-depth treatment.

Nearest Neighbours

Nearest neighbours (NN) is one of the simplest classifiers for supervised learning. For a given test image, it predicts that it will have the same label as that of the nearest training image, according to some distance metric, $d(\vec{x}_1, \vec{x}_2)$.

$$\text{nn}(\vec{x}; \vec{x}_1^{\text{train}} \dots \vec{x}_M^{\text{train}}, l_1^{\text{train}} \dots l_M^{\text{train}}) = l_i \tag{2.10}$$
$$\text{where } i = \underset{i \in 1 \dots M}{\text{argmin}}\, d(\vec{x}, \vec{x}_i^{\text{train}})$$

Nearest neighbours may be used with any distance metric.

Logistic Regression

Logistic regression (LR) is a simple linear classifier for binary classification where there are only two object classes, $l = 0$ and $l = 1$. It defines the likelihood of an image being in class $l = 1$ as a logistic function.

$$p(l = 1 | \vec{x}, \vec{w}) = \sigma(\vec{w} \cdot \vec{x}) \tag{2.11}$$

where $\sigma(x) = 1/(1 + e^{-x})$ is the logistic (sigmoid) function.

The likelihood is then fit to the training images, and used to classify a test image.

$$\text{lr}(\vec{x}; \vec{x}_1^{\text{train}} \dots \vec{x}_M^{\text{train}}, l_1^{\text{train}} \dots l_M^{\text{train}}) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.12}$$
$$\text{where } \vec{w} = \underset{\vec{w}}{\text{argmax}} \prod_{i=1}^{M} \sigma(\vec{w} \cdot \vec{x}_i^{\text{train}})^{l_i^{\text{train}}} \left(1 - \sigma(\vec{w} \cdot \vec{x}_i^{\text{train}})\right)^{1 - l_i^{\text{train}}}$$

Support Vector Machine (SVM)

Similar to logistic regression, a support vector machine (SVM) is also a linear classifier for binary classification where there are only two object classes, $l = 0$ and $l = 1$. It assumes there exists a separating linear boundary between the positive and negative classes and attempts to find the boundary that maximizes the margin between the classes. A test image is classified according to which side of the boundary it falls under.

$$\text{svm}(\vec{x}; \vec{x}_1^{\text{train}} \dots \vec{x}_M^{\text{train}}, l_1^{\text{train}} \dots l_M^{\text{train}}) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.13}$$

$$\begin{aligned} \text{where } \vec{w} = \underset{\vec{w}}{\text{argmax}} \big[& \max\{\vec{w} \cdot \vec{x}_i^{\text{train}} | l_i^{\text{train}} = 1\} \\ & -\min\{\vec{w} \cdot \vec{x}_i^{\text{train}} | l_i^{\text{train}} = 0\}\big] \end{aligned} \tag{2.14}$$

$$\text{s.t. } ||\vec{w}|| = 1$$

Similar to how nearest neighbours may be used with different distance metrics, SVMs are also often used with different kernels (inner product spaces). There are extensions to apply SVMs when there does not exist a separating boundary between classes.

K-Means Clustering

K-means clustering is a clustering algorithm that assumes the training images are all grouped around K clusters, $\vec{c}_1 \dots \vec{c}_K$, and returns the set of cluster centers that best describes the data.

$$\begin{aligned} \{\vec{c}_1, \dots, \vec{c}_K\} &= \underset{K}{\text{kmeans}}(\vec{x}_1^{\text{unspv}} \dots \vec{x}_K^{\text{unspv}}) \\ &= \underset{\{\vec{c}_1, \dots, \vec{c}_K\}}{\text{argmin}} \left[\sum_{i=1}^{M} \min_{\vec{c} \in \{\vec{c}_1, \dots, \vec{c}_K\}} ||\vec{x}_i^{\text{unspv}} - \vec{c}||^2 \right] \end{aligned} \tag{2.15}$$

For an image, $\vec{x}$, we can use the index of the nearest cluster center as a low dimensional descriptor for the image. We will refer to this step as "vector quantization".

$$\underset{K}{\text{vquantize}}(\vec{x}; \vec{x}_1^{\text{unspv}} \dots \vec{x}_K^{\text{unspv}}) = \underset{k \in \{1, \dots, K\}}{\text{argmin}} ||\vec{x} - \vec{c}_k||^2 \tag{2.16}$$

$$\text{where } \{\vec{c}_1, \dots, \vec{c}_K\} = \underset{K}{\text{kmeans}}(\vec{x}_1^{\text{unspv}} \dots \vec{x}_K^{\text{unspv}})$$

Note that to vector quantize a *set* of images, $\vec{x}_1 \ldots \vec{x}_N$, k-means is only applied *once* to find the cluster centers. The same set of cluster centers is used to assign indices to *all* images.

$$\underset{K}{\text{vquantize}}(\vec{x}_1 \ldots \vec{x}_N; \vec{x}_1^{\text{unspv}} \ldots \vec{x}_K^{\text{unspv}}) = [w_1 \ldots w_N] \tag{2.17}$$

$$\text{where } w_i = \underset{k \in \{1,\ldots,K\}}{\text{argmin}} ||\vec{x}_i - \vec{c}_k||^2$$

$$\{\vec{c}_1, \ldots, \vec{c}_K\} = \underset{K}{\text{kmeans}}(\vec{x}_1^{\text{unspv}} \ldots \vec{x}_K^{\text{unspv}})$$

Principal Components Analysis (PCA)

PCA is an unsupervised learning algorithm that searches for a D dimensional subspace, created from the span of orthonormal vectors $\vec{e}_1 \ldots \vec{e}_D$, which reconstructs the training images with the least error. We can use the mapping of $\vec{x}$ to the coordinate system defined by $\vec{e}_1 \ldots \vec{e}_D$ as a low dimensional descriptor for an image.

$$\underset{D}{\text{pca}}(\vec{x}; \vec{x}_1^{\text{unspv}} \ldots \vec{x}_K^{\text{unspv}}) = \begin{bmatrix} \vec{e}_1 & \ldots & \vec{e}_D \end{bmatrix}^T \vec{x} \tag{2.18}$$

$$\text{where } \{\vec{e}_1, \ldots, \vec{e}_D\} = \underset{\{\vec{e}_1,\ldots,\vec{e}_D\}}{\text{argmin}} \sum_{i=1}^{K} ||\vec{x}_i^{\text{unspv}} - \underset{\vec{e}_1 \ldots \vec{e}_D}{\text{proj}}(\vec{x}_i^{\text{unspv}})||^2$$

$$\text{s.t. } \vec{e}_1 \perp \vec{e}_2 \perp \ldots \perp \vec{e}_D$$

$$||\vec{e}_i|| = 1 \quad \forall i \in \{1, \ldots, D\} \tag{2.19}$$

Depending upon the specific nature of different datasets and the amount of data available, some of these methods may perform better than others. There is no known method for determining the appropriate method to use for a given dataset save for trying each one. Besides the performance characteristics, different methods have different computational characteristics that make them suitable for different problems. Aspects to keep in mind when choosing a method include:

- Training complexity: Some algorithms require little or no time to train (e.g. nearest neighbours), while others require solving a complex optimization problem (e.g. support vector machines).

- Testing complexity: Some algorithms may take a long time to train but are very fast during testing (e.g. logistic regression and support vector machines) while others take longer during testing (e.g. nearest neighbours).

- Storage requirements: Some algorithms require more storage space than others. For example, for D-dimensional data, a naive implementation of nearest neighbours requires storing the entire training dataset, while logistic regression represents the entire training dataset with a single D-dimensional vector.

- Convergence properties: Some algorithms have stronger convergence guarantees and simpler ways of checking for convergence than others. For example, k-means clustering is guaranteed to converge and checking for convergence is straightforward, whereas support vector machines are guaranteed to converge but checking for convergence depends on the setting of sensitive threshold parameters.

Because of these various aspects of different methods, choosing an appropriate method for a given task is not straightforward. [1] contains in-depth discussion of each of these methods.

Chapter 3

A Simple Object Recognition System

This chapter presents a simple but complete system for object recognition that solidifies the concepts introduced in the previous chapter. We will first describe *how* to extract histograms of oriented gradients (HOG) features from image patches, and then show how to create a HOG histogram descriptor for an image. The final classifier will map images to HOG histogram descriptors and predict class labels using nearest neighbours. Despite its simplicity, this system has been shown to have similar performance to state-of-the-art methods on the CityCars dataset.

3.1 Histograms of Oriented Gradients (HOG) Features

Much of object recognition can be seen as trying to find a similarity metric that reflects our own human notions of similar and dissimilar images. The Euclidean distance between the pixel colors of two images most certainly does *not* reflect our intuition. For example, the Euclidean distance between an image and the same image shifted by one pixel could be very large, and yet a human being would consider the two images almost identical. Histograms of oriented gradients (HOG) is a mapping from image patches to a new space under which Euclidean distance better reflects our own intuition.

HOG features are a commonly used representation for image patches that were de-

signed to be invariant to mild differences in translation, rotation, and lighting. It is highly related to the scale invariant feature transform (SIFT) originally proposed in [15] to solve the image alignment problem, but which has since been applied widely to many vision problems.

HOG features are typically used for grayscale images only. Color images are first transformed to grayscale by averaging the color channels before extracting HOG features. Given a patch extracted from a grayscale image, the HOG feature for the patch is computed by first calculating the intensity gradient for each pixel in the image plane, representing each gradient in polar coordinates with a magnitude and angle. The angles from 0° to 360° are subdivided into eight equally sized angular bins. The HOG feature for a given patch is the set of eight counts that indicates the number of pixels with gradients lying in each bin.

Given a grayscale image where $I(x, y)$ indicates the intensity for the pixel at column x, and row y in the patch, we first estimate its gradient along the horizontal and vertical directions and represent each gradient in polar coordinates.

$$\begin{aligned} \text{Given: } & \text{grayscale image, } I(x,y), \\ \text{Let } & G_x(x,y) = I(x+1,y) - I(x,y) \\ & G_y(x,y) = I(x,y+1) - I(x,y) \\ \text{Compute: } & ||G||(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2} \qquad (3.1) \\ & G_\theta(x,y) = \ \text{angle}(G_x(x,y), G_y(x,y)) \qquad (3.2) \end{aligned}$$

The HOG feature, $\vec{h}$, for a given patch of width W and height H is

$$\vec{h} = \begin{bmatrix} N_0 \\ \vdots \\ N_7 \end{bmatrix} \tag{3.3}$$

$$\text{where } N_k = \sum_{x=1}^{W} \sum_{y=1}^{H} \mathbb{I}((k-0.5)\cdot 45^\circ \leq G_\theta(x,y) < (k+0.5)\cdot 45^\circ), \quad k = 0, 1, \ldots, 7,$$

and $\mathbb{I}(\text{condition})$ is the indicator function

$$\mathbb{I}(\text{condition}) = \begin{cases} 1 & \text{if condition is true,} \\ 0 & \text{otherwise.} \end{cases} \tag{3.4}$$

This expression shows that a HOG feature is essentially a histogram of the orientations of the gradients within the patch. To make the HOG feature more resilient to random noise in the image, the counts, N_k, are also typically weighted by the magnitude of the gradient:

$$\vec{h} = \begin{bmatrix} N_0 \\ \vdots \\ N_7 \end{bmatrix} \tag{3.5}$$

$$\text{where } N_k = \sum_{x=1}^{W} \sum_{y=1}^{H} \left[\|G\|(x,y) \right.$$
$$\left. \cdot \mathbb{I}((k-0.5)\cdot 45^\circ \leq G_\theta(x,y) < (k+0.5)\cdot 45^\circ) \right], \quad k = 0, 1, \ldots, 7.$$

This feature is sufficient for the purposes of this example, but there are other application-specific variants of this feature that are commonly used. These variants differ in the number of angular bins in the histogram (we use eight here), whether the image patch is contrast-normalized before computing the feature, whether the final feature vector is normalized, whether each gradient is greedily assigned to an angular bin (as is done here) or whether it is linearly interpolated amongst its surrounding angular bins, etc. A

"pyramidal" variant of the HOG feature can also be computed by further subdividing the image patch into quadrants, computing a separate feature for each quadrant, and then concatenating these quadrant feature vectors to the overall feature vector. See [5] for a comprehensive treatment. Typical settings for the image patch size, W and H, include 4×4, 8×8, and 16×16.

3.2 HOG Histogram Descriptors

We will map all of our training and test images to HOG histogram-based descriptors before applying nearest neighbours for classification. This is done by first extracting HOG features for every patch in every training image, and then discretizing each HOG feature into an integer index, $w \in \{1, \ldots, V\}$, using vector quantization (2.16). We refer to each of these discretized HOG features as a *visual word*, and each visual word takes on a particular *vocabulary index* between 1 and V. The HOG histogram descriptor, $\vec{d}$, for a given image is the histogram indicating how many instances of each vocabulary index were in the image.

$$
\begin{aligned}
&\text{Given: unsupervised training images, } \vec{x}_1^{\text{unspv}} \dots \vec{x}_M^{\text{unspv}}, \\
&\qquad \text{image, } \vec{x}, \\
&\qquad \text{Let } \vec{h}_1^{\text{unspv}} \dots \vec{h}_N^{\text{unspv}} \text{ be HOG features from every patch in } \vec{x}_1^{\text{unspv}} \dots \vec{x}_M^{\text{unspv}}, \\
&\qquad \vec{h}_1 \dots \vec{h}_S \text{ be HOG features from every patch in } \vec{x}, \\
&\qquad w_1 \dots w_S = \underset{V}{\text{vquantize}}(\vec{h}_1 \dots \vec{h}_S; \vec{h}_1^{\text{unspv}} \dots \vec{h}_N^{\text{unspv}}) \\
&\text{Compute: } \vec{d} = \phi(\vec{x}; \vec{x}_1^{\text{unspv}} \dots \vec{x}_M^{\text{unspv}}) = \begin{bmatrix} N_1 \\ \vdots \\ N_V \end{bmatrix} \\
&\text{where } N_v = \sum_{i=1}^{S} \mathbb{I}(w_i = v).
\end{aligned}
\tag{3.6}
$$

Note that to create HOG histogram descriptors, we typically use the same training images for this unsupervised stage as we do for the later supervised stage. For this reason, and because of its relative simplicity, creating HOG histogram descriptors is generally thought of as a preprocessing step.

3.3 Supervised Training using Nearest Neighbours

Having mapped all images to HOG histogram descriptors, we use a simple nearest neighbours classifier (2.10) to classify a test image.

$$
\begin{aligned}
&\text{Given: HOG histogram descriptors for training images, } \vec{d}_1^{\text{train}} \dots \vec{d}_N^{\text{train}}, \\
&\qquad \text{training labels, } l_1^{\text{train}} \dots l_N^{\text{train}}, \\
&\qquad \text{HOG histogram descriptor for test image, } \vec{d}, \\
&\text{Predict: } l = \text{nn}(\vec{d}; \vec{d}_1^{\text{train}} \dots \vec{d}_N^{\text{train}}, l_1^{\text{train}} \dots l_N^{\text{train}})
\end{aligned}
\tag{3.7}
$$

3.4 Conclusion

The simple object recognition system presented in this chapter has obtained better classification accuracy on the CityCars dataset than many other much more sophisticated systems (Table 5.1). This system outlines a basic pipeline that is followed by the majority of object recognition systems. For specific applications and datasets, better classification results can been obtained by replacing one or more stages in the pipeline. Examples of this include

- Replacing HOG histogram descriptors with a more sophisticated mapping function, $\vec{d} = \phi(\vec{x})$. This mapping function may be carefully handcrafted to exhibit specific properties that are believed to be useful for classification as in the case with Gist descriptors [16]. Alternatively, a mapping function can be trained on data, as in the case of PCA [21], autoencoders [9], restricted Boltzmann machines [8], and the FLDA algorithm introduced in the following chapter.

- Replacing the nearest neighbours classifier with a more sophisticated classifier. This includes neural networks [13], convolutional networks [14], logistic regression, Gaussian process classifiers [11], etc. Support vector machines (SVM) with different kernels [12], are an exceedingly common classifier in object recognition systems that have been used to obtain good classification performance for specific datasets.

Chapter 4

Flobject Analysis

The previous section presented the components of a simple object recognition system. In this chapter, we will motivate and describe our flobject analysis framework, which takes advantage of motion information to help classify static images. We will also present one model that may be applied towards flobject analysis called the flow latent Dirichlet allocation (FLDA) model. An inference algorithm for FLDA based on collapsed Gibbs sampling will be derived in detail. Later, we will use the output of the FLDA algorithm to derive FLDA descriptors for images which will replace the HOG histogram mapping function $\phi(\vec{x})$ in the simple pipeline.

4.1 Key Concept

Flobject analysis is an unsupervised training framework that takes advantage of motion information to help classify *static* images. Specifically, it is a mapping function which takes as input a static image, $\vec{x}$, as well as a collection of training images, $\vec{x}_1^{\text{unspv}} \dots \vec{x}_M^{\text{unspv}}$, each with a corresponding *flow field,* $\vec{f}_1^{unspv} \dots \vec{f}_M^{unspv}$. The flow fields are used when calculating the descriptor, $\vec{d}$, for $\vec{x}$.

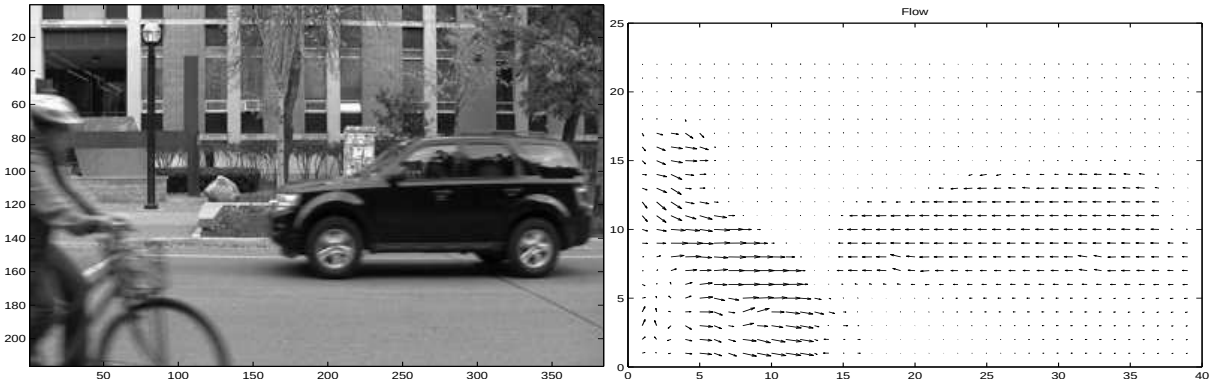
$$\begin{aligned} \text{Given: } & \text{unlabeled training images, } \vec{x}_1^{\text{unspv}} \dots \vec{x}_M^{\text{unspv}}, \\ & \text{flow fields, } \vec{f}_1^{\text{unspv}} \dots \vec{f}_M^{\text{unspv}}, \\ & \text{image, } \vec{x}, \\ \text{Compute: } & \vec{d} = \phi(\vec{x}; \vec{x}_1^{\text{unspv}} \dots \vec{x}_M^{\text{unspv}}, \vec{f}_1^{\text{unspv}} \dots \vec{f}_M^{\text{unspv}}) \end{aligned} \tag{4.1}$$

For an image with width W and height H, its corresponding flow field may be thought of as a $W \times H$ array of two dimensional vectors, each representing the motion of that particular pixel. Flow fields may be calculated using an existing optical flow algorithm, which takes as input two consecutive video frames and outputs a two dimensional flow vector for each pixel in the first frame. See Figure 4.1 for an example of a computed flow field. As a visualization tool, flow fields are often displayed by mapping $\mathbb{R}^2$ to a colormap, and representing the flow for each pixel as a color. See Figure 4.2 for an example of the same flow field visualized in this way.

One point to stress is that our objective is to improve classification performance on *static* images, i.e., the flow fields are not available during testing. Performance is improved by extracting a more sophisticated and useful descriptor for each static image. The training images and their flow fields are used to learn *how* to extract these descriptors.

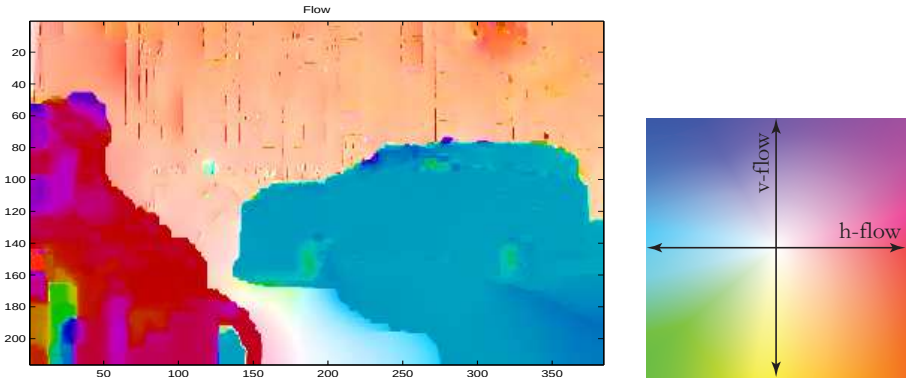
4.2 Motivation

As mentioned previously, a critical practical problem in object recognition is the lack of a sufficient number of labeled training images. This naturally motivates an unsupervised training framework for taking advantage of *unlabeled* training images to map images to a simpler descriptor space where classification is easy.



(a) Image (b) Flow Field

Figure 4.1: A training image from the CityCars dataset, shown with its corresponding flow field.



(a) Flow Field (b) Color Map

Figure 4.2: The same flow field as in Figure 4.1 visualized using a color map.

We will motivate the flobject analysis framework by considering specifically the problem of differentiating the object of interest in an image from the background. Because the class of an image is dependent primarily on the object of interest and to a lesser extent on the background, we should expect a useful descriptor for classification, $\vec{d}$, to be dependent on only the foreground pixels in an image and relatively invariant to changes in the background pixels. At the very least, the descriptor $\vec{d}$ should be factored such that a subset of the dimensions depends only on the foreground pixels, and the remaining dimensions depend only the background, so that during the supervised training stage the classifier may learn to ignore the proper dimensions. Thus, the mapping function must analyze the set of training images, $\vec{x}_1 \ldots \vec{x}_M$, to learn how to differentiate the foreground from the background pixels for a new image, $\vec{x}$ (See Figure 4.3).

In general, segmenting foreground from background is an extremely difficult task. The appearance of a specific object may vary widely from image to image due to out-of-plane rotations, changes in lighting, deformations, etc., and images of the same object may be shot in front of many different backgrounds. Indeed, there is an entire branch of computer vision dedicated to solving the segmentation problem.

The additional flow information for the training images simplifies the problem considerably by providing a strong clue as to the proper segmentation of an image. As in Figure 4.2, in the presence of a sharp discontinuity in the flow field, it is highly unlikely that pixels on opposite sides of the boundary belong to the same object. In a way, the flow field may be thought of as a crude surrogate for the ground-truth segmentation of an image. The advantage of using flow fields instead of ground-truth segmentations for training is that flow fields can be obtained cheaply using optical flow algorithms, whereas segmentations require manual hand labeling of images.

From a biological perspective, there is also extensive literature that suggests that

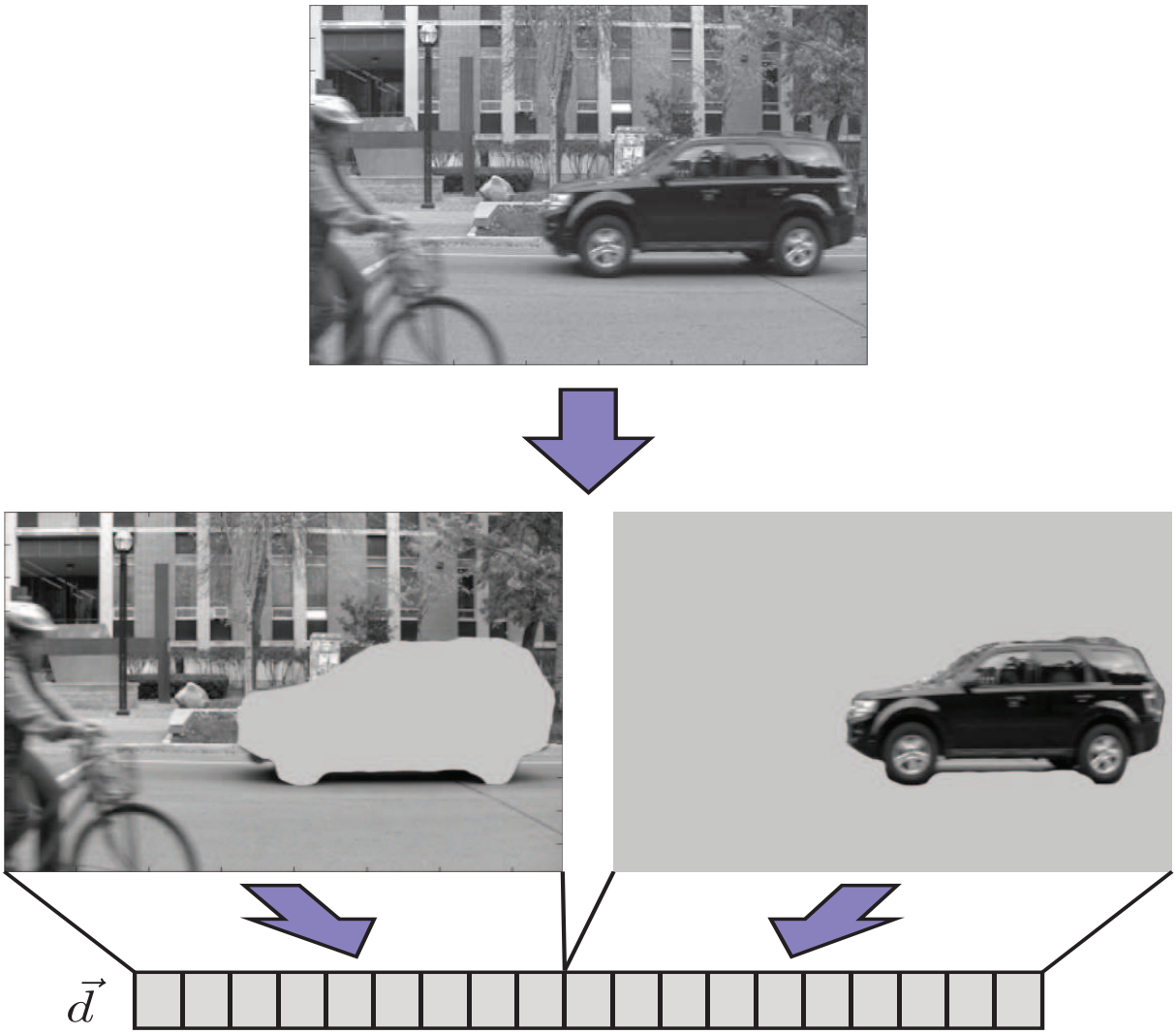


Figure 4.3: An ideal descriptor should be factored into the contribution from the foreground pixels and the background pixels.

motion cues and disparity information are critically important even for human beings for developing static image representations [17].

4.3 Latent Dirichlet Allocation (LDA)

There are many models and algorithms that are suitable for performing flobject analysis. The motion and appearance information can be jointly modeled using, for example, extensions of topic models, neural networks [13], restricted Boltzmann machines [8], Bayesian networks, etc. In this work we consider a topic modeling approach by extending the latent Dirichlet allocation (LDA) model [2].

Before describing our extension to account for flow vectors, we will first review the standard latent Dirichlet allocation (LDA) model. LDA [2] was originally proposed as a statistical topic model for modeling text data, where the objective is to discover underlying topics in a corpus containing many documents. All unique words in the corpus are assigned an index between 1 and V, and each document, d, is represented as a list of N_d integers, $\vec{w}_d \in \{1, \ldots, V\}^{N_d}$.

Topics are modeled as a distribution over the corpus vocabulary, and each document is modeled as a histogram over the vocabulary, thus ignoring all ordering information between words. LDA assumes that there is a finite number of topics that are shared by all documents in the corpus, and that each document is composed of a mixture of topics.

Figure 4.4 shows the generative model for LDA using plate notation (Figure 4.5). To generate a corpus of documents, we first generate a set of T topics, $\Phi = \{\vec{\phi}_1, \ldots, \vec{\phi}_T\}$, from a symmetric Dirichlet prior parameterized by β, and we generate the topic mixing proportions for each document, $\Theta = \{\vec{\theta}_1, \ldots, \vec{\theta}_D\}$, from a symmetric Dirichlet prior

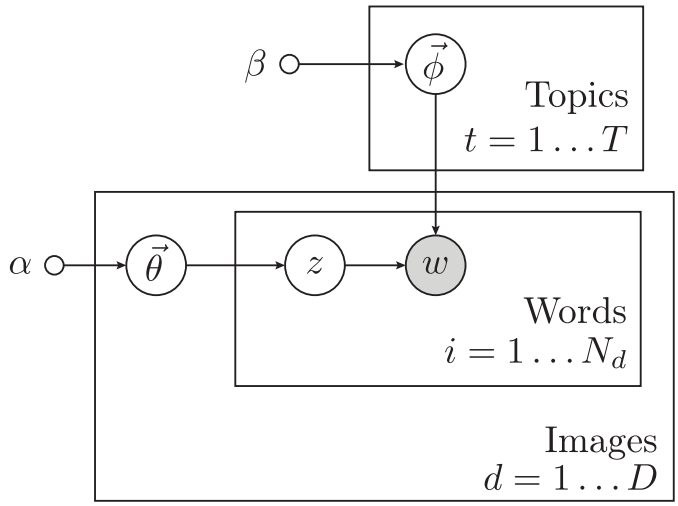


Figure 4.4: LDA Graphical Model

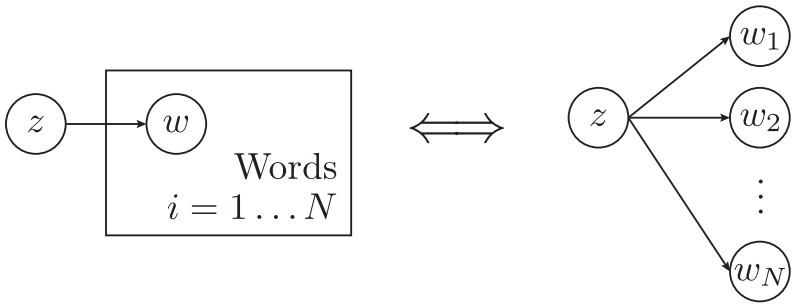


Figure 4.5: Plate Notation: Variables inside plates are repeated for the specified number of times.

parameterized by α. Each individual word in a document, d, is generated by drawing a topic assignment, $z_i \in \{1, \ldots, T\}$, from $\vec{\theta}_d$, and then drawing its vocabulary index, $w_i \in \{1, \ldots, V\}$, from the appropriate topic, $\vec{\phi}_{z_i}$.

For notational convenience we will use d_i to denote the document containing word w_i. The complete joint probability of data, $\vec{w}$, and parameters, $\vec{z}$, Φ, Θ, can then be expressed as

$$\begin{aligned}
&p(\vec{w}, \vec{z}, \Theta, \Phi) \\
&= p(\Theta)p(\Phi)p(\vec{z}|\Theta)p(\vec{w}|\vec{z}, \Phi) \\
&= \prod_d^D p(\vec{\theta}_d) \cdot \prod_t^T p(\vec{\phi}_t) \\
&\quad \cdot \prod_d^D p(\{z_k|d_k = d\}|\vec{\theta}_d) \cdot \prod_t^T p(\{w_k|z_k = t\}|\vec{\phi}_t) \\
&= \prod_d^D \mathrm{Dir}(\vec{\theta}_d|\alpha) \cdot \prod_t^T \mathrm{Dir}(\vec{\phi}_t|\beta) \\
&\quad \cdot \prod_d^D \mathrm{Discrete}(\{z_k|d_k = d\}|\vec{\theta}_d) \cdot \prod_t^T \mathrm{Discrete}(\{w_k|z_k = t\}|\vec{\phi}_t)
\end{aligned} \tag{4.2}$$

where the symmetric Dirichlet and Discrete distributions are defined as

$$\mathrm{Dir}(\vec{\theta}|\alpha) = \frac{\Gamma(T\alpha)}{\Gamma(\alpha)^T} \prod_t^T \theta_t^{\alpha-1} \tag{4.3}$$

$$\mathrm{Discrete}(\{w_i, \ldots, w_N\}|\vec{\theta}) = \prod_i^N \theta_{w_i}. \tag{4.4}$$

Having observed a corpus of documents, $\vec{w}$, we must marginalize over the latent variables Θ, and z, in order to discover the underlying topics, $p(\Phi|\vec{w})$.

$$p(\Phi|\vec{w}) = \iint p(\vec{z}, \Phi, \Theta|\vec{w}) d\vec{z} d\Theta \tag{4.5}$$

However this integral is intractable to compute exactly in general. Blei *et al* [2] originally proposed an approximate variational inference method for inferring the posterior. Here

we will describe an alternative inference method based on collapsed Gibbs sampling originally proposed by Griffiths *et al* [7] because of its simplicity and ease of extension to our own FLDA model. Before deriving the conditional distributions for Gibbs sampling we will list some properties of Dirichlet distributions that will be used later.

We will denote by $\mathbb{N}(\text{condition})$ the number of items from $k = 1, \ldots, N$ which satisfy the condition. More precisely,

$$\begin{aligned} \mathbb{N}(\text{condition}) &= \sum_{k=1}^{N} \mathbb{I}(\text{condition}); \\ \textit{e.g.}, \ \mathbb{N}(w_k = 1) &= \sum_{k=1}^{N} \mathbb{I}(w_k = 1), \end{aligned} \tag{4.6}$$

where $\mathbb{I}(\text{condition})$ is the indicator function defined in (3.4).

Having observed words $\vec{w} \in \{1, \ldots, W\}^N$, the *maximum a posteriori* (MAP) estimate for the parameters of the underlying discrete distribution, $p(\vec{\theta}|\vec{w})$, is

$$\begin{aligned} \vec{\theta}_{\text{MAP}} &= \underset{\vec{\theta}}{\text{argmax}}\ p(\vec{\theta}|\vec{w}) \\ &= \underset{\vec{\theta}}{\text{argmax}}\ p(\vec{\theta})p(\vec{w}|\vec{\theta}) \\ &= \underset{\vec{\theta}}{\text{argmax}}\ \text{Dir}(\vec{\theta}|\alpha) \cdot \text{Discrete}(\vec{w}|\vec{\theta}) \\ &= \frac{1}{N + W\alpha} \begin{bmatrix} \mathbb{N}(w_k = 1) + \alpha \\ \vdots \\ \mathbb{N}(w_k = W) + \alpha \end{bmatrix}, \end{aligned} \tag{4.7}$$

if we assume a symmetric Dirichlet prior over the parameter $\vec{\theta}$. The predictive distribution for a new word, w, having observed a set of words, $w_1 \ldots w_N$, is a discrete distribution parameterized by $\vec{\theta}_{\text{MAP}}$.

$$\begin{aligned} p(w|w_1 \dots w_N) &= \int p(\vec{\theta}|w_1 \dots w_N) p(w|\vec{\theta}) d\vec{\theta} \\ &= \text{Discrete}(w|\vec{\theta}_{\text{MAP}}) \\ &= \frac{\mathbb{N}(w_k = w) + \alpha}{N + W\alpha}. \end{aligned} \tag{4.8}$$

To infer a set of topics, Φ, for an observed corpus, $\vec{w}$, we will first use collapsed Gibbs sampling to obtain a typical sample from the posterior, from which we get the MAP estimate for Φ. The collapsed Gibbs sampler requires the conditional distribution of a single assignment, z_i, conditioned upon all other topic assignments, $\vec{z}_{\setminus i} = \{z_k | k \neq i\}$. Because LDA uses conjugate priors over its parameters, we can write down a closed form expression for this distribution, $p(z_i|\vec{z}_{\setminus i}, \vec{w})$.

$$\begin{aligned} p(z_i|\vec{z}_{\setminus i}, \vec{w}) &\propto p(\vec{z}, \vec{w}) \\ &= p(\vec{z}_{\setminus i}, \vec{w}_{\setminus i}) p(z_i, w_i|\vec{z}_{\setminus i}, \vec{w}_{\setminus i}) \\ &= p(\vec{z}_{\setminus i}, \vec{w}_{\setminus i}) p(z_i|\vec{z}_{\setminus i}, \vec{w}_{\setminus i}) p(w_i|\vec{w}_{\setminus i}, \vec{z}) \\ &\propto p(z_i|\vec{z}_{\setminus i}) p(w_i|\vec{w}_{\setminus i}, \vec{z}). \end{aligned} \tag{4.9}$$

Considering $p(z_i|\vec{z}_{\setminus i})$ first, we note that topic assignment z_i is dependent only on topic assignments that belong to the same document.

$$\begin{aligned} p(z_i|\vec{z}_{\setminus i}) &= p(z_i|\{z_k|k \neq i\}) \\ &= p(z_i|\{z_k|d_k = d_i \cap k \neq i\}) \\ &= \int p(\vec{\theta}_{d_i}|\{z_k|d_k = d_i \cap k \neq i\}) p(z_i|\vec{\theta}_{d_i}) d\vec{\theta}_{d_i} \end{aligned} \tag{4.10}$$

We see from comparing expression (4.10) to (4.8) that $p(z_i|\vec{z}_{\setminus i})$ is the predictive distribution for a new topic assignment. Hence,

$$p(z_i|\vec{z}_{\setminus i}) = \frac{\mathbb{N}(z_k = z_i \cap d_k = d_i \cap k \neq i) + \alpha}{\mathbb{N}(d_k = d_i \cap k \neq i) + T\alpha}. \tag{4.11}$$

To simplify $p(w_i|\vec{w}_{\setminus i}, \vec{z})$, we note that w_i only depends on the other words that are assigned to the same topic.

$$\begin{aligned} p(w_i|\vec{w}_{\setminus i}, \vec{z}) &= p(w_i|\{w_k| \cap k \neq i\}, \vec{z}) \\ &= p(w_i|\{w_k|z_k = z_i \cap k \neq i\}) \\ &= \int p(\vec{\phi}_{z_i}|\{w_k|z_k = z_i \cap k \neq i\}) p(w_i|\vec{\phi}_{z_i}) d\vec{\phi}_{z_i} \end{aligned} \tag{4.12}$$

Again, we compare expression (4.12) to (4.8) to see that $p(w_i|\vec{w}_{\setminus i}, \vec{z})$ is the predictive distribution for a new word under topic z_i. Hence,

$$p(w_i|\vec{w}_{\setminus i}, \vec{z}) = \frac{\mathbb{N}(w_k = w_i \cap z_k = z_i \cap k \neq i) + \beta}{\mathbb{N}(z_k = z_i \cap k \neq i) + W\beta}. \tag{4.13}$$

Combining the expressions for $p(z_i|\vec{z}_{\setminus i})$ and $p(w_i|\vec{w}_{\setminus i}, \vec{z})$, the final conditional distribution for $p(z_i|\vec{z}_{\setminus i}, \vec{w})$ is

$$\begin{aligned} &p(z_i|\vec{z}_{\setminus i}, \vec{w}) \\ &\propto p(z_i|\vec{z}_{\setminus i}) p(w_i|\vec{w}_{\setminus i}, \vec{z}) \\ &= \frac{\mathbb{N}(z_k = z_i \cap d_k = d_i \cap k \neq i) + \alpha}{\mathbb{N}(d_k = d_i \cap k \neq i) + T\alpha} \cdot \frac{\mathbb{N}(w_k = w_i \cap z_k = z_i \cap k \neq i) + \beta}{\mathbb{N}(z_k = z_i \cap k \neq i) + W\beta}. \end{aligned} \tag{4.14}$$

Thus to run the collapsed Gibbs sampler, we initialize $\vec{z}$ to random topics between 1 and T. Then, for each word in turn, we compute $p(z_i|\vec{z}_{\setminus i}, \vec{w})$ for $z_i \in \{1, \ldots, T\}$ and sample a z_i^*. After a number of iterations through all the words in the corpus, we use a single sample, $\vec{z}*$, to obtain the MAP estimate for the topics Φ using (4.7).

$$\Phi = \{\vec{\phi}_1 \dots \vec{\phi}_T\}$$
$$\text{where } \vec{\phi}_{t,v} = \frac{\mathbb{N}(w_k = v \cap z_k^* = t) + \beta}{\mathbb{N}(z_k^* = t) + W\beta} \tag{4.15}$$

4.4 Flow Latent Dirichlet Allocation (FLDA)

In order to apply the LDA framework towards flobject analysis, we make the analogy of treating images as documents of visual words. This can be done with a simple preprocessing step wherein HOG features are extracted from an input image and discretized, as was done to create the HOG histogram descriptors (Section 3.2). The corresponding flow for each visual word may be computed by averaging the dense per-pixel optical flow over the patch used to extract the each HOG feature. Thus flow latent Dirichlet allocation (FLDA) is a model for a collection of images with flows, each represented as a set of visual words with corresponding flow vectors. The output of FLDA will be the learned topics for the image collection, where we hope that different topics will correspond to different classes of objects.

Input: visual words in corpus, $\vec{w} = [w_1 \dots w_N]$,
document index of each word, $\vec{d} = [d_1 \dots d_N]$,
flow for each word, $\vec{f} = [\vec{f}_1 \dots \vec{f}_N]$,
where $w_i \in 1 \dots V$
$d_i \in 1 \dots D$
$\vec{f}_i \in \mathbb{R}^2$
Output: $\Phi = \{\vec{\phi}_1, \dots, \vec{\phi}_T\}$

We desire to use the flow to help guide the creation of meaningful topics, and therefore want our model to exhibit the following intuitive properties:

- Words with vastly different flows should prefer to belong to different topics. Consider an image with uniformly stationary flow everywhere except in a small concentrated region with very high flow. Depending on the appearance of the region, there is reason to consider assigning words within the region and words outside the region to different topics. The high flow region could be a baseball, for example.

- To counterbalance the previous point, when the appearance is sufficiently similar, words with vastly different flows should still be able to be assigned to the same topic. There could be two baseballs in an image, for example, one flying left and one flying right, and we still wish for them to be assigned to the same topic.

- When a set of words all possess similar flows, there should be *no additional* pressure from the flow to assign them to the same topic. In this case, only appearance should guide the separation of topics. This is because we do not want to assume that words with the same flow necessarily belong to the same object. Consider the case of a stationary background. There may be many different objects present, but they all have the same flow.

FLDA is an extension to LDA to account for word specific flows in such a way as to exhibit these desired properties. Appearance is modeled through the topics in standard LDA, and we account for the word specific flows by modeling all the flows within a document as a mixture of Gaussians. We refer to each Gaussian in the mixture as a *flow component*. Each word, w_i, will now be associated with a latent variable, z_i, that indicates its topic assignment, as well as an additional latent variable, c_i, that indicates its flow component assignment. Each topic in a document is associated with different mixing proportions, $\vec{\pi}_{d,t}$, over the flow components.

Figure 4.6 shows the generative process for FLDA. The visual words, $\vec{w}$, are generated identically to how they were in LDA. The set of T topics, $\Phi = \{\vec{\phi}_1, \ldots, \vec{\phi}_T\}$, are

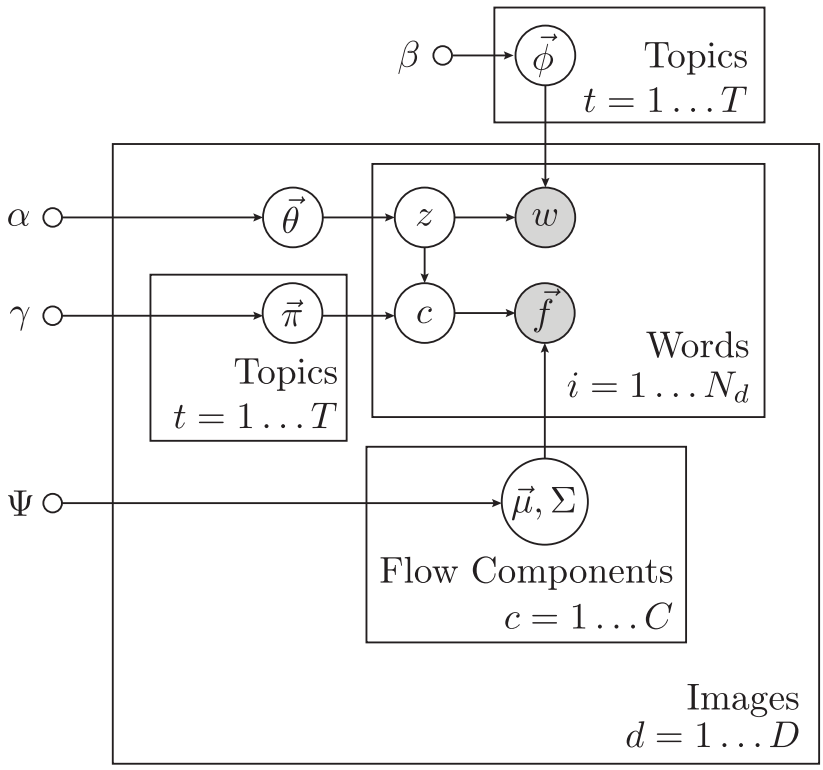


Figure 4.6: FLDA Graphical Model

drawn from a symmetric Dirichlet prior parameterized by β, and the topic mixing proportions for each document, $\Theta = \{\vec{\theta}_1, \ldots, \vec{\theta}_D\}$, are drawn from a symmetric Dirichlet prior parameterized by α. Each individual word in a document, d, is generated by drawing a topic assignment, $z_i \in \{1, \ldots, T\}$, from $\vec{\theta}_d$, and then drawing its vocabulary index, $w_i \in \{1, \ldots, V\}$, from the appropriate topic, $\vec{\phi}_{z_i}$. The new variables, $\vec{c}$, $\boldsymbol{\pi}$, $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$, are introduced to model the flows contained in an image. The set of C flow components for a document, d, parameterized by $\vec{\mu}_{d,1} \ldots \vec{\mu}_{d,C}$, and $\Sigma_{d,1} \ldots \Sigma_{d,C}$, are drawn from a Normal Inverse Wishart prior controlled by the set of parameters $\Psi = \{\vec{\mu}_0, \Lambda_0, \nu_0, \kappa_0\}$. The mixing proportion over flow components for each topic, $\vec{\pi}_{d,t}$, are drawn from a symmetric Dirichlet prior parameterized by γ. The corresponding flow for a word is generated by drawing a flow component assignment, $c_i \in \{1, \ldots, C\}$, from $\vec{\pi}_{d,z_i}$, and then drawing its flow vector, $\vec{f}_i \in \mathbb{R}^2$, from the appropriate flow component, $\mathcal{N}(\vec{f}_i | \vec{\mu}_{d,c_i}, \Sigma_{d,c_i})$.

The introduction of the topic specific mixing proportions over flow components $\vec{\pi}_{d,t}$

allows FLDA to exhibit our desired properties. Words with vastly different flows will be assigned to different flow components. Then depending on the similarity of their appearance, the words will be either encouraged or discouraged to be assigned to different topics. In the case where all the flows are the same and the c are identical, as in the example of a stationary background, the FLDA model is equivalent to the LDA model and topics are guided solely by the appearance.

The complete joint probability of data, $\vec{w}$, $\vec{f}$, and parameters, $\vec{z}$, $\vec{c}$, Φ, Θ, $\boldsymbol{\pi}$, $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$, can be expressed as

$$
\begin{aligned}
&p(\vec{w}, \vec{f}, \vec{z}, \vec{c}, \Phi, \Theta, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \\
&= p(\Theta)p(\Phi)p(\vec{z}|\Theta)p(\vec{w}|\vec{z}, \Phi)p(\boldsymbol{\mu}, \boldsymbol{\Sigma})p(\boldsymbol{\pi})p(\vec{c}|\vec{z}, \boldsymbol{\pi})p(\vec{f}|\vec{c}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \\
&= \prod_d^D p(\vec{\theta}_d) \cdot \prod_t^T p(\vec{\phi}_t) \cdot \prod_d^D p(\{z_k|d_k = d\}|\vec{\theta}_d) \cdot \prod_t^T p(\{w_k|z_k = t\}|\vec{\phi}_t) \\
&\quad \cdot \prod_d^D \prod_c^C p(\vec{\mu}_{d,c}, \Sigma_{d,c}) \cdot \prod_d^D \prod_t^T p(\{c_k|d_k = d \cap z_k = t\}|\vec{\pi}_{d,t}) \\
&\quad \cdot \prod_d^D \prod_c^C p(\{\vec{f}_k|d_k = d \cap c_k = c\}|\vec{\mu}_{d,c}, \Sigma_{d,c}) \\
&= \prod_d^D \mathrm{Dir}(\vec{\theta}_d|\alpha) \cdot \prod_t^T \mathrm{Dir}(\vec{\phi}_t|\beta) \\
&\quad \cdot \prod_d^D \mathrm{Discrete}(\{z_k|d_k = d\}|\vec{\theta}_d) \cdot \prod_t^T \mathrm{Discrete}(\{w_k|z_k = t\}|\vec{\phi}_t) \\
&\quad \cdot \prod_d^D \prod_c^C \text{N. Inv. Wishart}(\vec{\mu}_{d,c}, \Sigma_{d,c}|\Psi) \cdot \prod_d^D \prod_c^C \mathrm{Dir}(\vec{\pi}_{d,c}|\gamma) \\
&\quad \cdot \prod_d^D \prod_t^T \mathrm{Discrete}(\{c_k|d_k = d \cap z_k = t\}|\vec{\pi}_{d,t}) \\
&\quad \cdot \prod_d^D \prod_c^C \mathcal{N}(\{\vec{f}_k|d_k = d \cap c_k = c\}|\vec{\mu}_{d,c}, \Sigma_{d,c}).
\end{aligned} \tag{4.16}
$$

where the Normal Inverse Wishart distribution is defined as

$$\begin{aligned}
&\text{N. Inv. Wishart}(\vec{\mu}, \Sigma | \vec{\mu}_0, \Lambda_0, \nu_0, \kappa_0) \\
&= \frac{1}{Z}|\Sigma|^{-\frac{\nu_0}{2}-2} \exp\left(-\frac{1}{2}\text{tr}(\Lambda_0\Sigma^{-1}) - \frac{\kappa_0}{2}(\vec{\mu}-\vec{\mu}_0)^T\Sigma^{-1}(\vec{\mu}-\vec{\mu}_0)\right) \qquad (4.17) \\
&\text{where } Z = \frac{2^{\nu_0}}{|\Lambda_0|^{\nu_0/2}}\Gamma(\frac{\nu_0}{2})\Gamma(\frac{\nu_0-1}{2})(\frac{2\pi^{3/2}}{\kappa_0}).
\end{aligned}$$

As with standard LDA, to perform inference, we will use collapsed Gibbs sampling to obtain samples of the posterior $p(\vec{z}, \vec{c}|\vec{w}, \vec{f})$ and estimate the topics, Φ, from a single sample. The necessary conditional distributions for Gibbs sampling, $p(z_i|\vec{z}_{\setminus i}, \vec{c}, \vec{w}, \vec{f})$, and $p(c_i|\vec{c}_{\setminus i}, \vec{z}, \vec{w}, \vec{f})$, are derived in an analogous way to LDA.

$$\begin{aligned}
p(z_i|\vec{z}_{\setminus i}, \vec{c}, \vec{w}, \vec{f}) &\propto p(\vec{z}, \vec{c}, \vec{w}, \vec{f}) \\
&= p(\vec{z}_{\setminus i}, \vec{c}_{\setminus i}, \vec{w}_{\setminus i}, \vec{f}_{\setminus i}) p(z_i, c_i, w_i, \vec{f}_i | \vec{z}_{\setminus i}, \vec{c}_{\setminus i}, \vec{w}_{\setminus i}, \vec{f}_{\setminus i}) \\
&= p(\vec{z}_{\setminus i}, \vec{c}_{\setminus i}, \vec{w}_{\setminus i}, \vec{f}_{\setminus i}) p(z_i | \vec{z}_{\setminus i}, \vec{c}_{\setminus i}, \vec{w}_{\setminus i}, \vec{f}_{\setminus i}) \\
&\quad p(w_i | \vec{z}, \vec{c}_{\setminus i}, \vec{w}_{\setminus i}, \vec{f}_{\setminus i}) p(c_i | \vec{z}, \vec{c}_{\setminus i}, \vec{w}, \vec{f}_{\setminus i}) p(f_i | \vec{z}, \vec{c}, \vec{w}, \vec{f}_{\setminus i}) \\
&\propto p(z_i|\vec{z}_{\setminus i}) p(w_i | \vec{w}_{\setminus i}, \vec{z}) p(c_i | \vec{c}_{\setminus i}, \vec{z}) p(f_i | \vec{c}, \vec{f}_{\setminus i}) \\
&\propto p(z_i|\vec{z}_{\setminus i}) p(w_i | \vec{w}_{\setminus i}, \vec{z}) p(c_i | \vec{c}_{\setminus i}, \vec{z}) \qquad (4.18)
\end{aligned}$$

The expressions for $p(z_i|\vec{z}_{\setminus i})$ and $p(w_i|\vec{w}_{\setminus i}, \vec{z})$ remain identical to that of standard LDA. The simplication for $p(c_i|\vec{c}_{\setminus i}, \vec{z})$ is similar to the simplification for $p(w_i|\vec{w}_{\setminus i}, \vec{z})$. We note that the flow component assignment c_i is only dependent on the component assignments for words in the same document and topic.

$$\begin{aligned}
&p(c_i|\vec{c}_{\setminus i}, \vec{z}) \\
&= p(c_i | \{\vec{c}_k | k \neq i\}, \vec{z}) \qquad (4.19) \\
&= p(c_i | \{\vec{c}_k | d_k = d_i \cap z_k = z_i \cap k \neq i\}) \\
&= \int p(\vec{\pi}_{d_i,z_i} | \{c_k | z_k = z_i \cap d_k = d_i \cap k \neq i\}) p(c_i | \vec{\pi}_{d_i,z_i}) d\vec{\pi}_{d_i,z_i} \qquad (4.20)
\end{aligned}$$

We see from comparing (4.20) to (4.8) that $p(c_i|\vec{c}_{\setminus i}, \vec{z})$ is proportional to the predictive distribution of a new flow component assignment.

$$p(c_i|\vec{c}_{\setminus i}, \vec{z}) \propto \frac{\mathbb{N}(c_k = c_i \cap z_k = z_i \cap d_k = d_i \cap k \neq i) + \gamma}{\mathbb{N}(z_k = z_i \cap d_k = d_i \cap k \neq i) + C\gamma} \tag{4.21}$$

Combining expressions (4.14) and (4.21) gives

$$\begin{aligned}
&p(z_i|\vec{z}_{\setminus i}, \vec{c}, \vec{w}, \vec{f}) \\
&\propto p(z_i|\vec{z}_{\setminus i})p(w_i|\vec{w}_{\setminus i}, \vec{z})p(c_i|\vec{c}_{\setminus i}, \vec{z}) \\
&= \frac{\mathbb{N}(z_k = z_i \cap d_k = d_i \cap k \neq i) + \alpha}{\mathbb{N}(d_k = d_i \cap k \neq i) + T\alpha} \cdot \frac{\mathbb{N}(w_k = w_i \cap z_k = z_i \cap k \neq i) + \beta}{\mathbb{N}(z_k = z_i \cap k \neq i) + W\beta} \\
&\quad \cdot \frac{\mathbb{N}(c_k = c_i \cap z_k = z_i \cap d_k = d_i \cap k \neq i) + \gamma}{\mathbb{N}(z_k = z_i \cap d_k = d_i \cap k \neq i) + C\gamma}.
\end{aligned} \tag{4.22}$$

The above expression allows us to sample a single topic assignment z_i conditioned upon all other topic assignments and all of the flow component assignments. The next expression $p(c_i|\vec{c}_{\setminus i}, \vec{z}, \vec{w}, \vec{f})$ will allow us to sample a single flow component assignment c_i conditioned upon all other flow component assignments and all of the topic assignments.

$$\begin{aligned}
p(c_i|\vec{c}_{\setminus i}, \vec{z}, \vec{w}, \vec{f}) &\propto p(\vec{z}, \vec{c}, \vec{w}, \vec{f}) \\
&\propto p(z_i|\vec{z}_{\setminus i})p(w_i|\vec{w}_{\setminus i}, \vec{z})p(c_i|\vec{c}_{\setminus i}, \vec{z})p(f_i|\vec{c}, \vec{f}_{\setminus i}) \\
&\propto p(c_i|\vec{c}_{\setminus i}, \vec{z})p(f_i|\vec{c}, \vec{f}_{\setminus i})
\end{aligned} \tag{4.23}$$

The distribution $p(c_i|\vec{c}_{\setminus i}, \vec{z})$ has already been derived for $p(z_i|\vec{z}_{\setminus i}, \vec{c}, \vec{w}, \vec{f})$. The latter term $p(f_i|\vec{c}, \vec{f}_{\setminus i})$ is simplified by noting that the current flow vector $\vec{f}_i$ is dependent on only the other flow vectors that belong to the same component in the same document.

$$
\begin{aligned}
p(f_i|\vec{c},\vec{f}_{\setminus i}) &= p(f_i|\{\vec{f_k}|k\neq i\},\vec{c}) \\
&= p(\vec{f_i}|\{\vec{f_k}|c_k = c_i \cap d_k = d_i \cap k \neq i\}) \\
&= \iint p(\vec{\mu}_{d_i,c_i},\Sigma_{d_i,c_i}|\{\vec{f_k}|c_k = c_i \cap d_k = d_i \cap k \neq i\}) \\
&\qquad p(\vec{f_i}|\vec{\mu}_{d_i,c_i},\Sigma_{d_i,c_i}) d\vec{\mu}_{d_i,c_i} d\Sigma_{d_i,c_i}
\end{aligned} \tag{4.24}
$$

This last expression corresponds to the predictive distribution of a Gaussian likelihood function with a conjugate Normal Inverse Wishart prior on the mean and covariance. Though looking slightly cluttered, this result can be looked up in a standard probability textbook and is equal to a Student-t distribution, $t_{\text{dof}}(m, S)$, with dof degrees of freedom, mean m, and scale S.

$$
p(f_i|\vec{c},\vec{f}_{\setminus i}) = t_{\nu_n - 1}(\vec{\mu}_n, \frac{(\kappa_n+1)\Lambda_n}{\kappa_n(\nu_n-1)}) \tag{4.25}
$$

$$
\text{where } \vec{\mu}_n = \frac{\kappa_0}{\kappa_0+n}\vec{\mu}_0 + \frac{n}{\kappa_0+n}\vec{\mu}_{\text{ML}}
$$

$$
\Lambda_n = \Lambda_0 + S + \frac{\kappa_0 n}{\kappa_0+n}(\vec{\mu}_{\text{ML}} - \vec{\mu}_0)(\vec{\mu}_{\text{ML}} - \vec{\mu}_0)^T
$$

$$
\kappa_n = \kappa_0 + n
$$

$$
\nu_n = \nu_0 + n
$$

$$
\vec{\mu}_{\text{ML}} = \frac{1}{n}\sum_{k=1}^{N} \mathbb{I}(c_k = c_i \cap d_k = d_i \cap k \neq i)\vec{f_k}
$$

$$
S = \sum_{k=1}^{N} \mathbb{I}(c_k = c_i \cap d_k = d_i \cap k \neq i)(\vec{f_k} - \vec{\mu}_{\text{ML}})(\vec{f_k} - \vec{\mu}_{\text{ML}})^T
$$

$$
n = \mathbb{N}(c_k = c_i \cap d_k = d_i \cap k \neq i)
$$

Combining expression (4.21) and (4.25) gives us a closed form expression for the final conditional distribution $p(c_i|\vec{c}_{\setminus i}, \vec{z}, \vec{w}, \vec{f})$.

$$
\begin{aligned}
& p(c_i|\vec{c}_{\setminus i}, \vec{z}, \vec{w}, \vec{f}) \\
& \propto p(c_i|\vec{c}_{\setminus i}, \vec{z}) p(f_i|\vec{c}, \vec{f}_{\setminus i}) \\
& \propto \frac{\mathbb{N}(c_k = c_i \cap z_k = z_i \cap d_k = d_i \cap k \neq i) + \gamma}{\mathbb{N}(z_k = z_i \cap d_k = d_i \cap k \neq i) + C\gamma} \\
& \quad \cdot t_{\nu_n - 1}(\vec{\mu}_n, \frac{(\kappa_n + 1)\Lambda_n}{\kappa_n(\nu_n - 1)})
\end{aligned} \tag{4.26}
$$

The conditional distributions $p(z_i|\vec{z}_{\setminus i}, \vec{c}, \vec{w}, \vec{f})$ and $p(c_i|\vec{c}_{\setminus i}, \vec{z}, \vec{w}, \vec{f})$ are sufficient to define the Gibbs sampler. As with standard LDA, we first initialize $\vec{z}$ and $\vec{c}$ to random values. Then for each word in turn, we sample a $z_i^* \sim p(z_i|\vec{z}_{\setminus i}, \vec{c}, \vec{w}, \vec{f})$ and a $c_i^* \sim p(c_i|\vec{c}_{\setminus i}, \vec{z}, \vec{w}, \vec{f})$. After a number of iterations through all the words in the corpus, we use a single sample to estimate Φ using equation (4.15).

One useful property of the FLDA model is that even though it defines a distribution for images with flow, it naturally also models *static* images, i.e. images without observed flow. This is done by marginalizing over $\vec{f}$ in the model, which collapses FLDA back to standard LDA. We will make use of this property later when we create FLDA descriptors for static images during classification.

Chapter 5

Experimental Setup

For the following experiments, we show classification results on the CityCars and CityPedestrians dataset. The datasets are first preprocessed and transformed to a suitable representation for FLDA. The unsupervised FLDA stage then creates an FLDA descriptor for static training images which are input into a nearest neighbours algorithm for classification. This section will detail the preprocessing steps, and the creation of the descriptors used for later comparisons.

5.1 Dataset

We created the CityCars and the CityPedestrians dataset as a benchmark classification dataset for flobject analysis. The CityCars dataset is built for the binary classification task of determining whether or not a test image contains a car. The dataset contains 315 pairs of consecutive video frames containing side views of moving cars in an urban environment, and 338 static images without cars. Care was taken to ensure that the negative images were shot in identical locations as the positive examples. This prevents classifiers from using clues from the background to help classify images. See Figure 5.1.

We intentionally constructed the CityCars dataset so that it would pose a more realistic and challenging static image classification task. To demonstrate this, we used

	L2 NN	IK NN	IK SVM
CityCars	**65%** (3%)	55% (1%)	58% (2%)
CaltechCars	93% (3%)	98% (1%)	**99%** (1%)

Table 5.1: Comparisons shows that classifying cars in the CityCars dataset is much more difficult than in the Caltech101 dataset (CaltechCars). Classification was performed using spatial pyramid HOG (SPHOG) descriptors with nearest neighbour (NN) and SVM classifiers, using the intersection (IK) and L2 kernels. One std. dev. is shown in brackets.

state-of-the-art methods to compare classification performance on the CityCars dataset with that obtained on a dataset containing 123 Caltech images [6] containing side views of cars and 123 randomly selected images from other categories (we refer to this as the "CaltechCars" dataset). For both datasets, we randomly partitioned the data into one half for training and one half for testing and repeated each experiment 20 times to estimate confidence intervals.

The results for several classifiers that use the spatial pyramid HOG descriptor (SPHOG) are summarized in Table 5.1 and clearly support two observations. First, the CityCars dataset poses a much more difficult classification challenge than the CaltechCars dataset. This may be because the backgrounds, which take up most pixels in each image, are similar in the positive and negative examples. Second, whereas the intersection kernel (IK) SVM clearly outperforms simpler methods on the CaltechCars dataset, on the CityCars dataset the advantages of both the intersection kernel and the SVM disappear: the simple L2 nearest neighbour method outperforms SVM and IK-based methods. This result is explained by the fact that while the feature-AND operation of the intersection kernel enables high SVM accuracy when positive and negative examples do not share many features, it fails when they have many features in common, such as features derived

Figure 5.1: Positive (top) and negative (bottom) training (left) and test (right) images from the CityCars dataset.

Figure 5.2: Positive (top) and negative (bottom) training (left) and test (right) images from the CityPedestrians dataset.

from similar backgrounds. These results further support the conclusions presented in [18] regarding the inherent problems in benchmark datasets such as Caltech.

The CityPedestrians dataset is built for the task of determining whether or not a test image contains a pedestrian. The dataset contains 938 pairs of consecutive video frames containing side views of pedestrians, and 456 static images without pedestrians. For the same reason as with the CityCars dataset, we shot the negative images in identical locations as the positive examples. See Figure 5.2.

5.2 Dataset Preprocessing

Before analysis, all image pairs in CityCars and CityPedestrians are preprocessed and represented as sets of visual words with corresponding flow vectors, and all static images

are represented as sets of visual words. Visual words are represented as integer indices between 1 and W, and flow vectors are represented as two dimensional real valued vectors. Visual words are extracted using the method outlined in Section 3.2.

For every image, HOG features are extracted from every 16×16 pixel patch whose center lies on a 6×6 pixel grid. The extracted HOG features are then discretized into an integer index between 1 and W using k-means clustering. Flow vectors are extracted by first computing a dense per-pixel optical flow field from the two images in the image pair using the technique outlined in [3]. The flow vector corresponding to a specific visual word is computed by averaging the per-pixel optical flow over the word's 16×16 pixel patch with a Gaussian weighting window. Thus every image pair is represented as a set of visual words (an array of integers between 1 and W), and a corresponding set of flow vectors (an array of two dimensional real vectors).

For our experiments, the vocabulary size W for the CityCars dataset was chosen to be 1000 as is standard in the literature (*e.g.* [19]). The vocabulary size W for the CityPedestrians dataset was chosen to be 400 using cross validation over $W = 200$, 400, 1000, and 2000.

5.3 FLDA and LDA Descriptors

Given a set of training image pairs, each represented as a set of visual words with flow vectors (an array of integers from 1 to W, and a corresponding array of two dimensional vectors), we run Gibbs sampling on the FLDA model for 32000 iterations and obtain a *maximum a posteriori* estimate of the topics Φ from the last sample.

For a single static image, we obtain topic assignments for each word by running Gibbs sampling over the FLDA model while marginalizing over the flow. As mentioned previously, marginalizing over the flow collapses FLDA to standard LDA. Gibbs sampling is run for 600 iterations with the topics Φ fixed to its MAP estimate. We assume that the

last sample, $\vec{z}^*$, is a typical sample from the posterior distribution, $p(\vec{z}|\vec{w}, \Phi)$, and use this to compute T separate topic specific HOG histogram descriptors, $\vec{h}_1 \ldots \vec{h}_T$. The FLDA descriptor, $\vec{d}$, is created by concatenating the topic specific HOG histogram descriptors with the overall HOG histogram descriptor described in Section 3.2.

$$\begin{gathered}\text{Given: visual words in image, } \vec{w},\\ \text{topic assignments, } \vec{z}^*,\\ \text{Compute: } \vec{d} = \begin{bmatrix} \vec{h}_{\text{overall}} \\ \vec{h}_1 \\ \vdots \\ \vec{h}_T \end{bmatrix} \\ \text{where } h_{t,v} = \mathbb{N}(w_k = v \cap z_k^* = t)\\ h_{\text{overall},v} = \mathbb{N}(w_k = v)\end{gathered} \tag{5.1}$$

The individual histograms, $\vec{h}_{\text{overall}}$, $\vec{h}_1 \ldots \vec{h}_T$, may be individually normalized before concatenation. We refer to this as the *topic normalization*, and we test the descriptors under no topic normalization, L1, and L2 topic normalizations. After concatenation, the overall descriptor, $\vec{d}$, is normalized again. We refer to this as the *global normalization*.

LDA descriptors are created in an identical fashion to FLDA descriptors except that during the unsupervised stage we use LDA instead of FLDA to train the topics Φ.

5.4 GIST Descriptors

GIST Descriptors [16] were invented by Oliva and Torralba for the purpose of whole scene recognition. Oliva defines a variety of image filters that correspond intuitively to eight different global properties in an image. These properties include "Naturalness", "Openness", "Perspective", "Symmetry" among four others. For a given static image, its

GIST descriptor is calculated by filtering the image with each of the eight filters and concatenating the filter responses.

5.5 Spatial Pyramid HOG (SPHOG) Descriptors

Given a single image where HOG features have been extracted from every patch in the image and discretized to an integer index between 1 and W, the SPHOG descriptor is created by first dividing the image into four equal sized quadrants, the northeast (quadrant 1), northwest (quadrant 2), southwest (quadrant 3), and southeast (quadrant 4) quadrants. Quadrant-specific HOG histogram descriptors, $\vec{h}_1 \ldots \vec{h}_4$, are extracted for each quadrant and then concatenated with the overall HOG histogram descriptor, $\vec{h}_{\text{overall}}$, to form the SPHOG descriptor.

$$\text{Given: visual words in image, } \vec{w},$$

$$\text{Compute: } \vec{d} = \begin{bmatrix} \vec{h}_{\text{overall}} \\ \vec{h}_1 \\ \vdots \\ \vec{h}_4 \end{bmatrix} \tag{5.2}$$

$$\text{where } h_{q,v} = \mathbb{N}(w_k = v \cap k \in \text{quadrant } q)$$

$$h_{\text{overall},v} = \mathbb{N}(w_k = v)$$

5.6 Intersection Kernel

The intersection kernel is a similarity metric for comparing discrete distributions often used in the object recognition community. It is defined intuitively as the amount of overlap between distributions. Given two N-dimensional discrete distributions, $\vec{h}_1$, and $\vec{h}_2$, we define the intersection kernel $k(\vec{h}_1, \vec{h}_2)$ as

$$k(\vec{h}_1, \vec{h}_2) = \sum_{k=1}^{N} \min(h_{1,k}, h_{2,k}). \tag{5.3}$$

One of the current state-of-the-art classifiers for the Pascal VOC challenge is a support vector machine (SVM) in conjunction with the spatial pyramid match kernel [12]. The spatial pyramid match kernel is equivalent to applying the intersection kernel on spatial pyramid HOG descriptors.

Chapter 6

Experimental Results

The experiments we report investigate the usefulness of flobject analysis, and specifically of the FLDA model, for mapping static images to descriptors for object recognition. We show classification results on the CityCars dataset, compare inter-dataset generalization capability, and explore properties of our method. Using the topics already learned from FLDA, we also develop a spatial hierarchical FLDA (H-FLDA) descriptor for analyzing articulated objects. The H-FLDA descriptors are demonstrated on a classification task on the CityPedestrians dataset.

6.1 Comparisons on the CityCars Dataset

We compared the FLDA descriptors (3000 dimensional, based on learning two topics) to four alternative descriptors: a 1000 dimensional HOG histogram descriptor, a 5000 dimensional spatial pyramid HOG (SPHOG) descriptor, a 960 dimensional Gist descriptor, and a 3000 dimensional descriptor obtained from standard LDA (see Chapter 5). Using the nearest neighbour classifier (NN), we experimented with both the Euclidean (L2) distance and the intersection kernel (IK) similarity metric. Furthermore, we explored various normalization schemes for the descriptors. For the SPHOG, LDA, and FLDA descriptors we demonstrate performance for no normalization, L1 topic normalization,

L2 NN	None	L1	L2
HOG	**65%** (5%)	60% (6%)	54% (2%)
SPHOG	**65%** (7%)	64% (6%)	54% (4%)
Gist	69% (5%)	69% (4%)	**70%** (5%)
LDA	62% (5%)	**64%** (5%)	59% (4%)
FLDA	61% (7%)	**82%** (4%)	73% (5%)

Table 6.1: L2 nearest neighbour classification accuracy on the CityCars dataset for various descriptors and normalization schemes. One std. dev. is shown in brackets.

and L2 topic normalization. For the HOG, and Gist descriptors we demonstrate performance for no normalization, L1 global normalization, and L2 global normalization. The SPHOG, LDA, and FLDA descriptors are always globally L1 normalized. Results using Euclidean distance are shown in Table 6.1, and those for the intersection kernel in Table 6.2. The data is partitioned into one half for training and one half for testing, and each experiment is repeated 20 times to estimate confidence intervals.

The FLDA descriptor achieves the best overall classification accuracy, outperforming other descriptors under both Euclidean distance and intersection kernel similarity. This indicates that FLDA topic-based factorization of the histograms is beneficial to classification. The LDA descriptor performs similarly to SPHOG, showing that topics inferred without motion coherence do not help with classification. Finally, we note that normalization affects performance, and that the effect is not consistent across different distance metrics, indicating that normalization plays an important role. Based on these results, we use the Euclidean distance nearest neighbours classifier with L1 topic normalization and L1 global normalization for the FLDA descriptor in later experiments.

IK NN	None	L1	L2
HOG	57% (4%)	56% (4%)	**67%** (5%)
SPHOG	56% (4%)	57% (4%)	**63%** (6%)
Gist	61% (9%)	**63%** (9%)	60% (7%)
LDA	56% (3%)	57% (4%)	**70%** (6%)
FLDA	56% (3%)	66% (7%)	**79%** (4%)

Table 6.2: Intersection kernel nearest neighbour classification accuracy on the CityCars dataset for various descriptors and normalization schemes. One std. dev. is shown in brackets.

Figure 6.1 compares the nearest neighbours for some test images for the FLDA and the SPHOG descriptors. The SPHOG descriptors are more likely to confuse the background features with the object features.

6.2 Inter-Dataset Generalization

We next investigated to what extent the FLDA descriptors generalizes across datasets as compared to SPHOG descriptors. To demonstrate this, we extract FLDA descriptors and SPHOG descriptors for the CaltechCars and CityCars dataset, and divide both datasets into training and test sets. We then show the classification performance on the test set of a single dataset while using the training set of the other. The full cross comparisons for the CaltechCars and CityCars dataset for the SPHOG and the FLDA descriptors are shown in Table 6.3 and Table 6.4 respectively. FLDA descriptors demonstrate significantly better generalization capabilities than SPHOG descriptors. FLDA descriptors outperform SPHOG on two of the four categories, and compare similarly for the other two. Note that using FLDA descriptors trained on CityCars to classify CaltechCars

Figure 6.1: Test images (left column) along with the nearest training image using FLDA descriptors (middle column) and SPHOG descriptors (right column).

SPHOG		Testing	
		CityCars	CaltechCars
Training	CityCars	65% (3%)	63% (6%)
	CaltechCars	62% (3%)	93% (3%)

Table 6.3: Inter-dataset generalization (classification accuracy) using SPHOG descriptors.

FLDA		Testing	
		CityCars	CaltechCars
Training	CityCars	82% (4%)	73% (3%)
	CaltechCars	63% (2%)	93% (2%)

Table 6.4: Inter-dataset generalization (classification accuracy) using FLDA descriptors.

yields significantly better results than using SPHOG descriptors trained on CaltechCars to classify CityCars (73% vs. 62%).

6.3 Exploration of Training Conditions

We investigated the sensitivity of FLDA descriptors to various training parameters. Figure 6.2 compares how the classification accuracy is affected by the number of labeled training examples for FLDA, SPHOG, and HOG descriptors. We see that the classification accuracy for SPHOG and HOG descriptors saturates much earlier and at a lower point than that for FLDA descriptors.

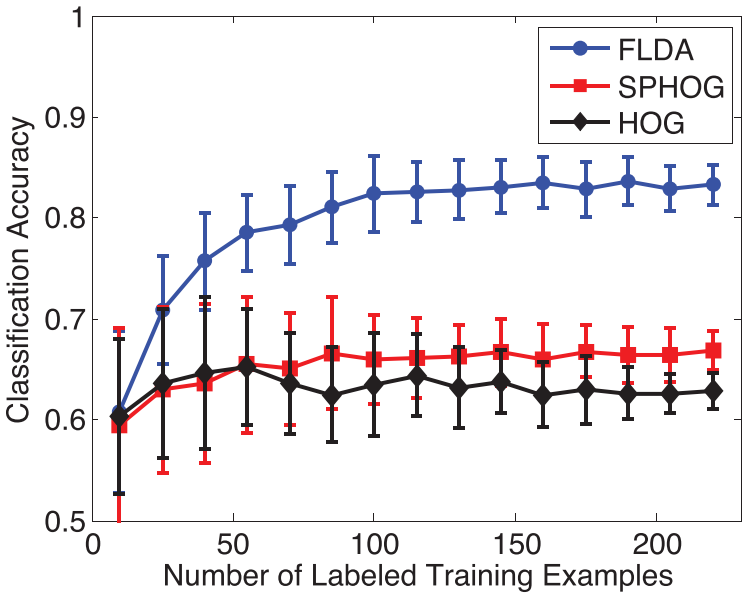


Figure 6.2: Classification accuracy as a function of the number of labeled images used during supervised training.

Optical flow fields are often noisy due to out-of-plane rotations, optical flow aperture problems, and reflections from specular surfaces. To investigate how sensitive the classification performance is to noise in the flow, we artificially add increasing amounts of Gaussian noise to the flow and note its effect on classification accuracy. Figure 6.3 shows that as the noise increases, the performance degrades smoothly towards the level of SPHOG performance.

Figure 6.4 shows how classification accuracy is affected when we adjust the number of topics, T, learned by FLDA. We see that the classification accuracy peaks sharply at two topics. This might be due to measuring accuracy on a binary classification task, or because the dataset strongly supports only two topics, a car topic and a background topic.

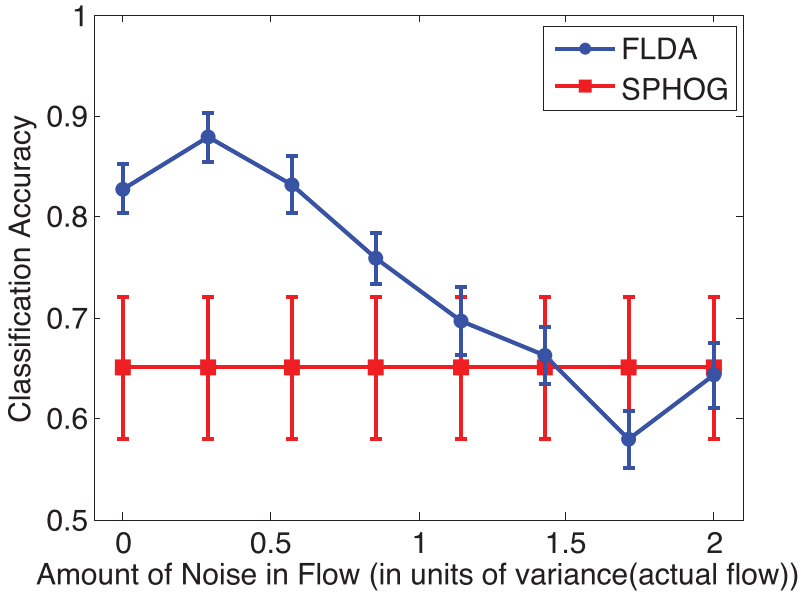


Figure 6.3: Effect of adding Gaussian noise to flow on classification accuracy.

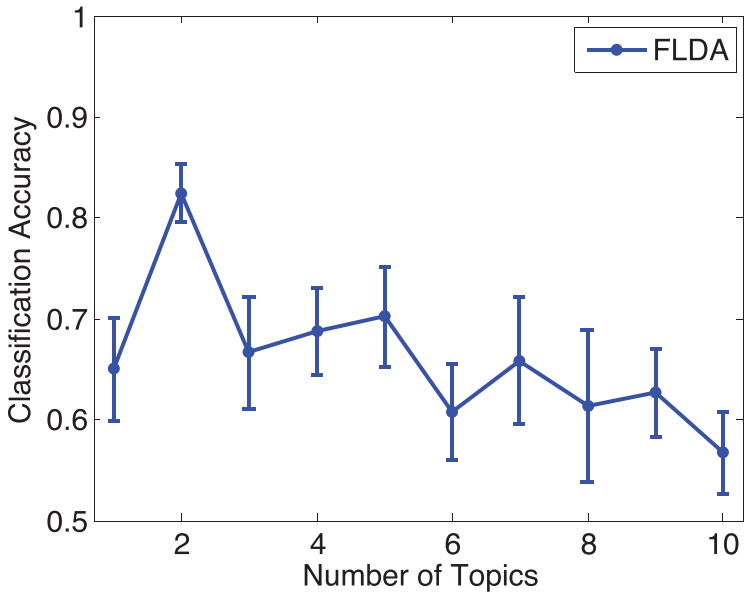


Figure 6.4: Effect of adjusting the number of topics on classification accuracy.

6.4 Analyzing Spatially Localized Objects using Hierarchical FLDA Descriptors

Objects are frequently spatially localized in an image, but if there is enough training data or the appearance features on an object are sufficiently distinctive then we can still achieve good classification performance without having to explicitly model spatial locality. For example, the FLDA-based descriptors described in Section 5.3 perform well despite not accounting for the locations of visual words in an image. However, in the case of insufficient amounts of training data, or where the appearance of an object may vary significantly, then it is helpful to explicitly model spatial locality. Here we show how the topics learned using FLDA can be used to construct hierarchical FLDA (H-FLDA) descriptors which improves classification performance by taking advantage of spatial locality.

To investigate the advantages of explicitly modeling spatial locality, we constructed the CityPedestrians dataset, which consists of images of side views of pedestrians walking in an urban environment. Pedestrians typically take up only a small region of each image, and also vary significantly in appearance between images because of differences in attire, posture, gender, etc. The CityPedestrians dataset is designed so that an explicit model for spatial localization would be critical for achieving good classification performance. Indeed, the classification accuracies for several previously described descriptors are shown at the top of Table 6.5 and are comparable to random guessing (50%).

One advantage of flobject analysis is that once the flow-based topics have been learned, they may be used in many different ways to create descriptors for static images. We previously show how to create FLDA descriptors from the learned topics. Here we will show how to create another descriptor, hierarchical FLDA (H-FLDA) descriptors, which

L2 NN	None	L1	L2
HOG	**55%** (1%)	55% (1%)	51% (1%)
SPHOG	**53%** (1%)	53% (1%)	53% (1%)
LDA	**55%** (1%)	53% (1%)	53% (1%)
FLDA	**55%** (1%)	52% (1%)	52% (1%)

	Hierarchical Descriptors
H-LDA	57% (1%)
H-FLDA	**68%** (1%)

Table 6.5: L2 nearest neighbour classification accuracy of hierarchical descriptors on the CityPedestrians dataset.

take spatial locations into account and combines visual words in a hierarchical fashion. For a given image, the H-FLDA descriptor is a T dimensional vector that sums to 1, where T is the number of topics obtained during FLDA. This descriptor is created by scanning a 10×10 window over the image, computing the histogram over visual words for each window, and assigning each window to the nearest topic (using L2). The H-FLDA descriptor is formed by normalizing the histogram over the total number of windows assigned to each topic. For comparison, H-LDA descriptors are created in the same fashion but with LDA-derived topics.

The bottom part of Table 6.5 demonstrates that the hierarchical H-FLDA descriptor improves performance over the FLDA descriptor, H-LDA descriptor, and other descriptors on the CityPedestrians dataset.

Chapter 7

Related Work

Methods for motion and activity modeling, and video summarization are relevant to our work. However, while the literature in this area is extensive, to our knowledge none of that work is directed towards training methods that extract good representations for *static* images. An interesting avenue for further research is to examine previously described methods for jointly modeling appearance and motion and consider integrating out the motion portion of the model after training. This approach could lead to different methods for flobject analysis.

Regarding our extension of LDA to model word-specific optical flow vectors, while there is no previous work in this area, our extended model is most similar in spirit to the work of Sudderth *et al* [20]. They extend LDA hierarchically to allow for variable spatial layouts of visual words. If spatial coordinates in their model were replaced with flow vectors, their model could be used for flobject analysis. However, it is not clear how well this approach would work since their model was not tested in the absence of spatial information. Others have pre-clustered visual words according to spatial layout and then applied LDA using either subregion-defined words [4] or 'doublet' words that encode spatially proximal visual words [19]. A similar approach could be used to pre-cluster visual words according to similar optical flow. However, it is not clear how optical

flow should then be integrated out for static image analysis.

Jojic and Frey [10] proposed a method for learning layers of flexible sprites from a video sequence. Their method addresses segmenting images based on motion but did not use optical flow. The objective of the method is also different than that of flobject analysis. The flexible sprites method tries to identify a small number of sprites in order to accurately model a single long video sequence, whereas flobject analysis tries to identify a large collection of objects existing among many short video sequences. The flexible sprites method is more suitable for video compression whereas flobject analysis is more suitable for object recognition.

Chapter 8

Conclusion

Flobject analysis takes advantage of training images with corresponding motion information to infer good representations for *static* images. We constructed the CityCars and CityPedestrians datasets, which includes pairs of consecutive video frames containing moving cars and people. A flow-based extension of LDA was developed for doing flobject analysis, and we demonstrated that the additional motion information used during unsupervised training significantly improved classification performance over other methods which only train on static images.

The additional flow information simplifies the unsupervised training stage considerably. The proposed topic modeling approach for flobject analysis was simple but demonstrated good performance on a binary classification task. This initial result motivates further research into modifying existing state-of-the-art unsupervised training techniques to take advantage of flow information. We hope to eventually scale the method up to be able to learn about hundreds of different classes of objects from a large video collection.

The framework we described can be improved in several ways, and altogether different kinds of models can be used for flobject analysis, such as those that decompose the image into a hierarchy of parts or use layers of variables to account for high-order statistics. Importantly, while the framework requires moving objects, this can be achieved for static

objects by panning a camera or using stereo data. Also, while here we tested the flobject analysis framework for object recognition tasks, future work could include applications to image segmentation and object localization.

Bibliography

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, August 2006.

[2] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3, 2003.

[3] A. Bruhn, J. Weickert, and C. Schnörr. Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61, 2005.

[4] L. Cao and L. Fei-Fei. Spatially coherent latent topic model for concurrent object segmentation and classification. In *ICCV*, 2007.

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

[6] 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*, 2004.

[7] Thomas L. Griffiths and Mark Steyvers. Finding Scientific Topics. *PNAS*, 101, 2004.

[8] G.E. Hinton, S. Osindero, and Y.W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18, 2006.

[9] G.E. Hinton and R.R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313, 2006.

[10] N. Jojic and B.J. Frey. Learning flexible sprites in video layers. In *CVPR*, 2001.

[11] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Active learning with gaussian processes for object categorization. In *ICCV*, 2007.

[12] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[13] B.B. Le Cun, JS Denker, D. Henderson, RE Howard, W. Hubbard, and LD Jackel. Handwritten digit recognition with a back-propagation network. In *NIPS*, 1990.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86, 1998.

[15] D.G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.

[16] A. Oliva and A. Torralba. Building the Gist of a Scene: The Role of Global Image Features in Recognition. *Progress in Brain Research*, 155, 2006.

[17] Y. Ostrovsky, E. Meyers, S. Ganesh, U. Mathur, and P. Sinha. Visual parsing after recovery from blindness. *Psychological Science*, 20, 2009.

[18] N. Pinto, D.D. Cox, and J.J. DiCarlo. Why is real-world visual object recognition hard? *PLoS computational biology*, 4, 2008.

[19] J. Sivic, B.C. Russell, A.A. Efros, A. Zisserman, and W.T. Freeman. Discovering objects and their location in images. In *ICCV*, 2005.

[20] E.B. Sudderth, A. Torralba, W.T. Freeman, and A.S. Willsky. Describing visual scenes using transformed objects and parts. *International Journal of Computer Vision*, 77, 2008.

[21] M.A. Turk and A.P. Pentland. Face recognition using eigenfaces. In *CVPR*, 1991.

[22] D. Waltz. Understanding line drawings of scenes with shadows. In *The Psychology of Computer Vision*, 1975.