Discriminative vs. Generative Object Recognition: Objects, Faces, and the Web

Thesis by

Alex Holub

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

2007

(Defended April 30, 2007)

Acknowledgements

One adage that I have found true of completing a thesis: "the thesis is a marathon not a sprint". As in a marathon the thesis can be both frustrating at moments and extremely exhilarating at other moments. One must push through the lulls and embrace the apogees. I feel extremely blessed to not only have had the chance to be at Caltech for my PhD, but also to have worked in the the Vision Laboratory. The social and intellectual atmosphere were amazing and I have made friendships which will last a lifetime. Students at Caltech have the freedom to pursue their own ideas and research agendas, and they will find intellectual peers among their colleagues who are equally as curious and motivated as they are. Every effort is made to remove distractions from the lives of students and allow them to focus on research and learning. There are undoubtedly few institutions in the world which can offer so much to students.

As I begin to unfold the next chapter in my life, it becomes compelling to reflect back all those years to when I began the marathon. My first project at Caltech was a collaboration between Professor Pietro Perona and Professor Gilles Laurent and involved modeling locust olfaction. My intellectual growth becomes apparent when I think of the difference between how I approached research questions and new projects then, and how I approach them now. A significant part of the growth I can attribute to Professor Perona. He taught me, among other things, how to think about research questions, how to pick interesting and worthwhile directions for research, how to convey complex ideas simply and clearly, and, above all, how to think critically. He gave me confidence when I felt down, and gave me guidance when I was stumbling. He is a mentor and a friend, and I would like to thank him for supporting me and allowing me the freedom to grow and pursue ideas while in his lab.

During my PhD I had the opportunity to collaborate with numerous amazing researchers, too numerous to completely mention here. But one collaborator (and in essence a co-advisor) who stands out is Professor Max Welling. I thank him for investing time in me

and for being one of the most exciting collaborators I have had a chance to work with. He is full of ideas and has a contagious passion for research which can only inspire those around him. In addition I would like to thank Dr. Michael C. Burl with whom I have recently begun collaborating. Finally, I would like to thank a fellow graduate student with whom I also began collaborating recently, Pierre Moreels. He has not only been a loyal friend for the past years, but is also one of those people who it is truly enjoyable to work with.

Next I would like to briefly acknowledge and thank all those members of the lab who have helped me over the years, from minor technical problems, to bringing me up to speed on the latest technological gadgets. There were so many great moments we shared together, and, the little community that is the Vision lab will be one of the things I miss most about Caltech. Marco helped me stumble through numerous latex questions and his dry (and pointed) sense of humor has brightened many late-nights in the lab. Marco is one of those people who truly enjoys helping others and teaching others, and I can say that I personally benefited a great deal from this proclivity. Claudio helped me with everything from writing C-code to installing cable TV at my house. He was always willing to take time out of his schedule to help in any way he could, and he would often spend hours helping me install software and guiding my electronic purchases. Lihi acted as a "big-sister" in the lab, reading over papers and being supportive of whatever idea I had at the time. And I have also had the pleasure of getting to know some of the newer additions to the lab. In particular, Merrielle, who has a contagious smile and affable personality. She brightens up the lab merely with her presence and makes one look forward to being in lab every day. And Ryan, who is one of the brightest and knowledgable people I have met at Caltech, someone who is always willing and excited to engage in a scientific conversation and is knowledgable about a vast spectrum of things. Finally I would also like to thank Andrea, who has been a great friend and confidant to me over the years. And I would like to thank all the others in the lab, both from the past and the present, thank you!

I have been extremely fortunate to have made so many friendships at Caltech which have made this one of the best periods of my life. I am not going to explicitly mention most of these wonderful people here, but you know who you are, and you know how much I will miss all of you. A special thanks goes out to Kimberly, who has been unfailingly supportive of me for the past year, one of the craziest and most stressful of my life. I cannot express my gratitude in words. My father, although far away in distance, has given me sage advice over

the years and has kept me grounded and given me the strength to shake off the unavoidable setbacks inherent in any life. And my mother, who has been the rock in my life since I was a young child and who I admire and respect more than anyone in this world.

Thank you all.

Alex Holub

Abstract

The ability to automatically identify and recognize objects in images remains one of the most challenging and potentially useful problems in computer vision. Despite significant progress over the past decade computers are not yet close to matching human performance. This thesis develops various machine learning approaches for improving the ability of computers to recognize object categories. In particular, it focuses on approaches which are able to distinguish between object categories which are visually similar to one another. Examples of similar visual object categories are Motorcycles and Bicycles, and Lions and Cougars. Distinguishing between similar object categories may require different algorithms than distinguishing between different categories. We explore two common machine learning paradigms, generative and discriminative learning, and analyze their respective abilities to distinguish between different sets of object categories. One set of object categories which we are exposed to on a daily basis are face images, and a significant portion of this thesis is spent analyzing different methods for accurately representing and discriminating between faces. We also address a key issue related to the discriminative learning paradigms, namely how to collect the large training set of images necessary to accurately learn discriminative models. In particular, we suggest a novel active learning which intelligently chooses the most informative image to label and thus drastically reduces (up to 10×) the time required to collect a training set. We validate and analyze our algorithms on large data-sets collected from the web and show how using hybrid generative-discriminative techniques can drastically outperform previous algorithms. In addition, we show how to use our techniques in practical applications such as finding similar-looking individuals within large data-sets of faces, discriminating between large sets of visual categories, and increasing the efficiency and speed of web-image searching.

Contents

Chapter 1

Introduction

1.1 The Information Age

Over the past century humanity has witnessed the emergence of the information age. Information exists everywhere. We have unprecedented access to it via a variety of modalities. Personal computers, whose memory systems were capable of holding at most kilobytes of data decades ago are now capable of storing up to terabytes of information. However, the information age is being driven only in small part by the increase in size of personal computer storage devices. The main thrust for the information age has been the nearly ubiquitous emergence of the internet and the World Wide Web (WWW or "web"). The internet allows personal computers, once isolated islands in the sea of humanity, to connect, talk to one another, and share information, thereby making entire domains of knowledge and media available literally at the click of a mouse. In 2005 it was estimated that there were over 11.5 billion web-pages, up from 550 million in 2001 (this information was obtained from the online encyclopedia wikipedia at http://www.wikipedia.com).

As the amount of information available to us increases at an exponential rate we face a very real and pragmatic problem: how can we sift through all the information in order to retrieve only the information we are interested in? It seems akin to the perennial aphorism of finding a needle in a haystack. It is not practical, nor feasible, to have humans sort and organize these vast repositories of information. One solution, being utilized more and more is to use intelligent computer programs to search, mine, and extract the information we require. Companies like Google, America Online, and Yahoo! are in the business of providing the user with the information they require, and these companies are increasingly relying on sophisticated algorithms to help them retrieve and provide the information. As

the sophistication of these algorithms increases, so does their reach into different media modalities such as audio, video, and images.

This thesis explores how intelligent computer algorithms can be used to help process visual information. In particular we are interested in processing information about object categories contained in images. In the process we show how these techniques can be used to help with some very real problems facing users of the web today, such as how to efficiently search for images using the Google search engine, or how to recognize and find similar-looking faces to a particular person.

1.2 Learning and Computer Vision

The earliest significant forays into computer vision can be dated to the late 1960's. Since then there has been a great deal of progress, although, notably, very few computer vision algorithms are being used by the general public today. Most computer algorithms benefit very specialized markets such as assembly line robotics or digital character recognition by the postal service. Is computer vision ready to make the leap into the mainstream and significantly impact society?

Recently researchers have begun to tackle one of the most interesting and potentially useful problems in computer vision, namely how to automatically recognize and discriminate between object categories contained in images. For example, if shown an image of a "shoe", a computer should be able to discern that there is indeed a shoe in the image. These sorts of algorithms could impact web search, surveillance, mobile robotics, and the organization of visual information on our computers. Indeed, successful algorithms in the object recognition domain seem to have the potential of fundamentally changing our society.

How might a computer be able to recognize the face of Tom Cruise in our personal photo collection? There are two broad paradigms we could use to find Tom which we introduce next. The first paradigm, which we call building a model, specifies, down to pixel precision, what Tom looks like. This includes his prominent nose and his broad smile. At this point we might be able to compare all the image on our computer and find the image which was most correlated with the model of Tom we have specified. Of course we would also need to specify every possible deformation of Tom's face which occurs when he smiles, turns slightly away from the camera, or when the pictures is taken in different lighting conditions. The

number of possible pixel combinations we need to specify sky-rockets and rapidly becomes intractable.

The second paradigm we explore involves allowing the computer to learn all possible deformations of Tom's face. For example, consider presenting the computer with various examples of Tom, taken under different conditions. If presented with enough examples it could learn the typical deformations and then be able to recognize Tom in a new image. The implementor of the algorithm must then provide two different kinds of information: (1) an effective learning algorithm, and (2) the appropriate training examples of Tom's face. What is appealing about this "learning" paradigm is that it is flexible. Once we have an algorithm which can learn to recognize Tom, it should be able to learn a representations for Julia Roberts, Brad Pitt, etc, provided we give it the appropriate training examples.

There are many approaches which combine aspects of both approaches by introducing prior information into a learning approach. For instance one could imagine specifying broadly the deformations possible, but learning the exact parameters for a particular face via learning. In fact, most learning methods are parametric in nature, thereby implicitly encompassing prior information via the parameters of the model.

In this thesis we focus on learning object categories. Our algorithms for learning implicitly make use of prior information about what we expect to find in the images. For instance, when recognizing and distinguishing between faces, we assume as a prior that we will find eyes, a nose, and a mouth within the image. Nevertheless, the main thrust of our approach is to use learning rather than hand-crafted prior models.

1.3 Discriminative and Generative Learning

Learning object categories from exemplar images seems like a great paradigm in the abstract. However, as with most great ideas, the devil is in the details. How do we construct algorithms which allow computers to learn representations of these object categories from images? The field of Machine Learning is filled with algorithms and techniques for learning complex representations and models from training data. The choice of what learning algorithm to use is somewhat dependent on the task which we would like the computer to solve. One natural task in object recognition is to ask the computer to distinguish between two objects. These two objects might be Airplanes and Bicycles, or John Kerry and George

Bush.

We can approach the problem of discriminating between categories with two general philosophies. In the first, which is called generative learning, we learn that John Kerry has two eyes, a nose, a mouth, and great head of hair. If we present the computer with a novel image of John, we can identify him by these characteristics. However, given that we have learned a very general model of John, if we presented the computer with an image of George Bush as well, the computer might incorrectly indicate that John was contained in the image, when, in reality it was an image of George. Another philosophy, called discriminative learning aims at learning the differences between the categories. We noted that John has a great hair, so if modeled this fact, namely that John has a better head of hair than George, than, when presented with an image containing either George or John, we should be able to tell them apart. These intuitions about discriminative and generative learning are formalized in Section 6.2 and form the backbone of this thesis. In particular, when should we utilize generative learning approaches versus using discriminative learning approaches? In which situations is one beneficial and why?

Our intuition tells us that discriminative approaches might be most useful when our object categories are quite similar to one another. After all, we would expect Bicycles and Airplanes to have very few over-lapping features, while Bicycles and Motorbikes might have many more features in common (such as wheels). One particularly interesting case of similar categories are Human Faces. Faces are, at a pixel level, very similar to one another. Humans are extremely adept at discriminating between different faces, presumably by recognizing subtle differences in facial appearance. We begin this thesis by studying Face Recognition in Part I. We construct complete systems capable of identifying and discriminating between different faces and demonstrate our techniques on collections of face data which exhibit large variations in appearance.

Our discussion and experiments on faces lead naturally to the issue of discriminative and generative learning discussed above because faces are extremely similar to one another. One inherent requirement of discriminative algorithms is how to obtain the labeled data necessary to train the algorithms. By labeled data we are referring to example images of John Kerry and George Bush which are necessary for learning. We approach the problem of labeling images using algorithms which fall under the general category of "Active Learning" algorithms. Active Learning algorithms present the user with images to label which are the

most informative, thereby minimizing the total number of images which must be labeled. Part II discusses various active learning approaches, and we present a novel method which works effectively in reducing the number of images which are required to be labeled during Google Image searches.

The final part of the thesis, Part III, is an in-depth analysis and comparison of generative and discriminative learning techniques. We study when discriminative methods are more useful for object recognitions, what benefits generative models provide, and suggest hybrid generative-discriminative algorithms which combine the best of both worlds.

Part I

Recognizing Faces

Chapter 2

Googling for Jenifer Aniston

2.1 Abstract

How do we identify images of the same person in photo albums? How can we find images of a particular celebrity via Google Image searches? Both of these tasks require solving numerous challenging issues in computer vision. Among them are (1) finding the location of individuals in images, (2) maintaining robustness to variability in pose, image quality, lighting, occlusion, and scale, and (3) using an appropriate distance metric in order to compare the detected individuals. Many of these issues have been worked on in isolation within the computer vision community, however there exist few systems which combine all components together into a complete system capable of being effective when confronted with the variability of real images*. In this work we aim to create a visual recognition system, that, given a target image, is able to find all other images of that individual within a typical photo collection or web search. Each individual is represented by a feature vector composed of extracted patches around automatically detected facial key-points. We use a training set of over 1000 images to generate a distance metric which combines and weights different components of the feature vector to drastically increase recall performance. We analyze different components of our system to provide insight into such as issues as: Which facial features are most important for recognition? What representation for facial features is most useful? Finally, we demonstrate our system on a large image-set of 99 individuals which we collected from the web and show impressive ability to* automatically *detect images of the same person.*

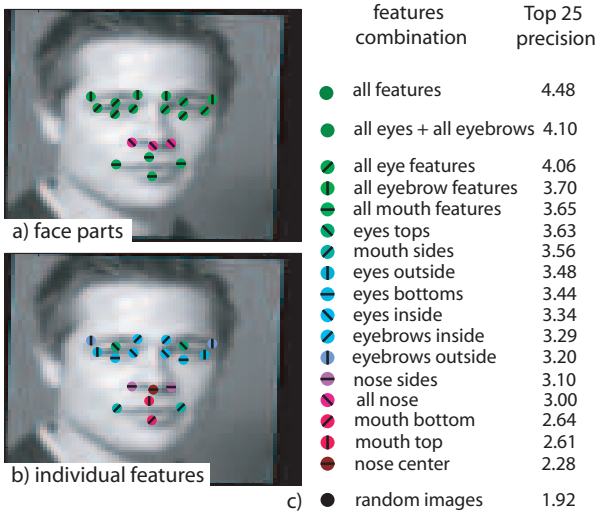

Figure 2.1: Which facial key-points are most useful? We rank the ability of different facial key-points *by themselves* to recall images of the same person within a large image data-set (see Section 2.6.1 for a more thorough description of the performance metric) (a) The performance of different features on recall experiments. Corresponding number and color-code in panel c) indicates the precision within the top 25 images, higher number is better. Recall performed only with: eyebrows structure, eyes structure, nose structure and mouth structure. (b) Performance when a face is characterized by a single individual patch (two in case of symmetry, e.g. both sides of the mouth are included together). The same color scale is used for panel a) and b). (c) Scores of parts and individual features. The set consisting of all parts performs best, followed by the eyes and eyebrows structures. Overall all features related to the eyes and eyebrows perform well. 'Random images' is the baseline method that draws randomly 25 images and indicates the precision. This experiment used 7×7 patches and raw intensity values, the ranking did not change significantly when using 13×13 patches and image gradients. Note that we used an L1 distance to measure similarity and did not apply a our learned distance metric for this comparison.

2.2 Introduction

The most common image searches on the web are celebrity pictures. This is, at the moment, implemented by searching keywords, and it is thus susceptible to errors: many images containing our favorite heroes are missed because they are not properly indexed. If we could search images for known faces, no celebrity picture would be missed any longer. This technology could, more in general, be useful for (1) searching the Web for an image of a particular individual (e.g., searching for Brad Pitt on Google Image search), and (2) finding all images of an individual within a personal photo collection (e.g., asking for all images of Grandpa in your personal photo collection or MySpace pages). Note that this work was done in collaboration with Pierre Moreels.

There has been significant previous work involving various aspects facial recognition

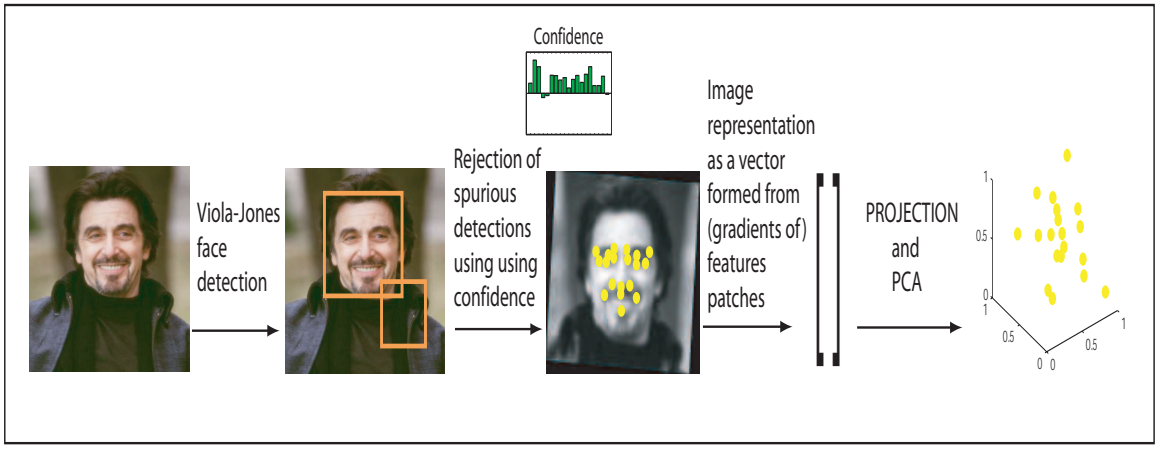

Figure 2.2: Schematic of the facial identity finder algorithm, starting from a photo-album all the way to an optimal representation in a new space. Notice that the whole process is automated: we don't require any hand-labeling of images. We *do* require that the faces be more or less in frontal poses as the Viola and Jones detector as well as the Everingham feature detectors were trained on frontal faces and frontal face features respectively.

and detection, and face recognition is one of the most studied fields in computer vision. Many algorithms exist for detecting the location of a face in images including Rowley et al.[RBK98], Schneiderman et al.[SK00] and the popular Viola and Jones [VJ01] detector which is both fast and reliable. Once the face is detected, a feature representation must be created in order to compare different faces. Both global, in which the entire face is segmented and made into a feature vector, and local, in which only specific key-points are used to create a feature vector, have been suggested. Global approaches, such as the well-known Eigenface [TP91] technique which maps global representations for a face onto a eigenbasis, tend to suffer from slight variations in alignment which can cause large differences in feature representation. Local representations tend to have more success and have been used by numerous authors including [EZ06]. In particular, the Everingham facial feature detector [EZ06], which was trained on a large data-base of facial features seems to work very well in practice.

Although there has been much previous work on facial representations and recognition, most of the work has focused on controlled environment, well-segmented and/or aligned images of faces. This includes the CMU Pie data-set and the Yale face data-set. There has been relatively little work on faces in real scenes, although there are exceptions, most notably [SSZ04, ESZ06] who work on finding images of actors in scenes and [BBE+04] who

use both text and images to automatically associate names to faces from news articles.

Furthermore, there is existing work on learning distance metrics and/or mapping functions to place facial features in appropriate spaces to perform recall on. The most prominent is the work using FisherFaces [BHK97] which uses a combination of Eigenfaces and Linear Discriminant analysis to learn a linear mapping.

Finally, the psychological literature has made numerous interesting contributions regarding the cues which humans use to recognize faces. See [Sin02] for an excellent review, which, among other observations, states that the eye-regions of faces are very useful for facial recognition.

In this work, we utilize and extend some of this existing work, including the Viola and Jones [VJ01] face-finder and the Everingham facial feature finder [ESZ06], and combine it with novel methods for learning distance metrics in face space, to create a complete system capable of accurately retrieving images of the same person in a challenging data-set of 99 individuals obtained from the Web (our individuals happen to be Celebrities). We report performance results on numerous different tasks and compare our performance to other existing methods.

Section 2.3 describes the general algorithm we employ. In Section 2.4 we describe the face detector, the facial feature finder, and facial feature representations used. In Section 2.6.1 we show how to create an effective distance metric. Section 3.6 shows and compares results of our complete system. We conclude in Section 2.10.

2.3 Overview

Figure 2.2 gives a schematic of our facial recognition system. First we find the face using Viola and Jones. Next we remove spurious detections and extract a feature representation. Finally we project the face into a new space using a learned mapping function. The resulting feature representations should reflect facial identity, i.e. images of the same person will be closer to one another than images of different people.

2.3.1 Performance Metrics

Here we introduce 3 performance metrics which mimic typical facial recognition tasks. The three performance tasks which we consider are: (1) Given a target image of a particular

Figure 2.3: Two examples for the 'Top 25 Precision' retrieval task. We queried for Will Smith (panel a) and for Owen Wilson (panel b) - the query images are showed in the top left with a blue outline. In both cases the query returned 7 correct results (green outline) out of the 25 closest matches after projection and PCA. It is important to remember that the category 'Will Smith' and the category 'Owen Wilson' contain only 10 examples each, therefore a perfect answer would return 10 samples from the target category out of 25 results. This is to contrast with the 'Google-images' engine, where the target category typically contains hundreds of images for each celebrity.

identity, how often is the nearest neighbor to this target, actually the same individual? This task would be useful in such applications as finding the most similar-looking celebrity to a person. The rank of the best performing image of the target individual among nearest neighbors, is termed *Best Rank Distance*. (2) Given a target individual, our goal is to find

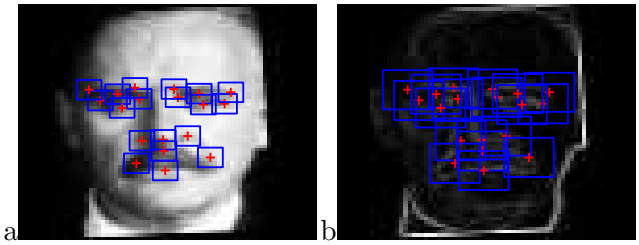

Figure 2.4: Example of the relative size of each patch. (a) Examples of 7×7 patches on a raw intensity figure (this yielded the optimal performance for raw patch representations, see Figure 2.6). (b) Example of gradient image and the size of extracted patches, 13×13 patches performed best when using the gradient as a feature.

K images, among which images of the target individual are as frequent as possible. Think of, for instance, a Google Image Search in which we would desire a high number of the target individual within the top 25 returned images. This is the detection performance with a recall of 25, we term it *Top 25 Precision*. Note that in our experiments we limit the number of images of each single individual to 10, such that the highest value achievable by *Top 25 Precision* would be 9. (3) The final metric concerns the rank distance between a query image and the set of all other images of the same individual. This rank distance is taken in the whole space of available images. In the optimal situation, this rank distance would be one for each query image, as we would like all the images of the same celebrity to be closer to one another than images of other people. We normalize this rank distance by the total number of images and call this metric *Average Rank Distance*. For both (1) and (2) we consider each celebrity individually and search for them among the Data-Set of 200 images. We use and compare the results from these metrics to evaluate the performance of different representations and learned distance functions.

2.4 Feature Extraction and Representation

In order to detect faces we use the OpenCV [Int] implementation of the Viola and Jones [VJ01] face detector. The output from the face detector is a set of bounding boxes which identify the position of the face in the image. Most images from the '99-Celebrities' data-set (see Section 2.5) yield a single detection, however in 68 cases (out of 1266) it mistakenly finds two or more faces.

Next, these bounding boxes are used as input for the impressive Everingham facial-feature detectors [ESZ06]. This detector identifies the position of 19 facial features as well

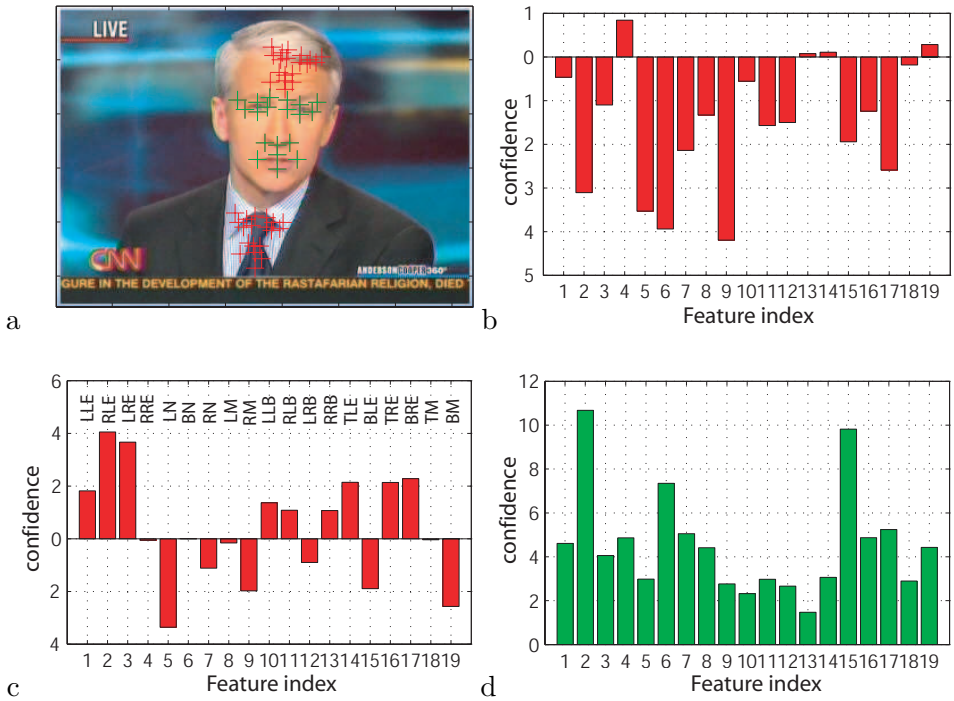

Figure 2.5: Example of Viola and Jones detection and Everingham feature detectors. (a) The Viola-Jones face detector found 3 'faces' in the image. b-c-d) confidences associated to each feature from each of the 3 detections. The following abbreviations are used: L=left, R=right, T=top, B=bottom, B=eyebrow, E=eye, N=nose, M=mouth. Using our heuristic based on parts' confidence, the incorrect detections are rejected (in red - the detection on the forehead corresponds to confidence scores in b., the detection on the tie corresponds to confidence scores in c.). The correct face is accepted (shown in green)

as the confidence it has with these detections. The facial-features identified are shown in Figure 2.4.

We used the following heuristic in order to discard spurious face detections: let CF be the 19-dimension confidence vector for a face, $C^+ = max(CF, 0)$ be its positive component and $C^- = -min(CF, 0)$ its negative component. We accept a face detection if $\sum_k = 1^19C^+ > 4 * \sum_k = 1^19C^-$, and reject it otherwise. Using this method, we successfully rejected all but 4 false face detections, and introduced only 4 new false rejects. Figure 2.5 shows an example of successful rejection of spurious matches on an image that generated 3 detections.

Image Set	Total Number Images	VJ Multiple Detection	False Alarms
99 Celebrities	1066	68	4
BG Celebrities	200	15	0

Table 2.1: Table showing the size of our data-sets as well as the performance of the Viola and Jones detections. (First Column) The two data-sets, both a set of 99 individual celebrities downloaded from the web and a set of 200 other images also downloaded from the web. (Second Column) The total number of images in each data-set. (Third Column) Total number of images for which the Viola and Jones algorithm generated multiple detections. (Last Column) Number of remaining false alarms after heuristic based to remove spurious detections (see Section 2.4).

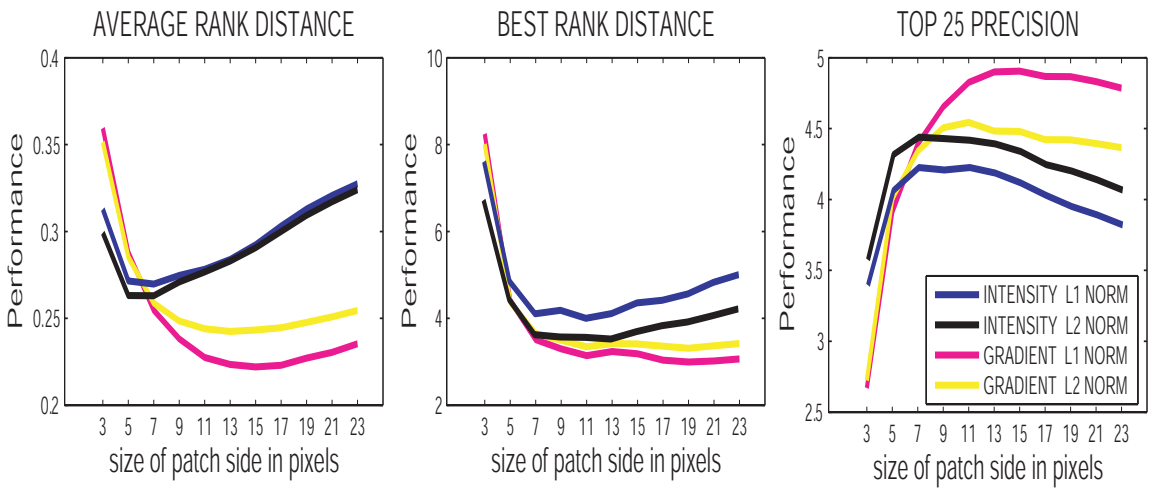

Figure 2.6: Variation of our three performance criteria with size of the patches used for face representation. We plot results for a simplified experiment for which no projection was performed, L_1 and L_2 distances were used on the raw patches. We display results of the simplified experiment both using image intensities and image gradients. X-axis: the size of patches extracted. Y-axis: performance using various metrics. All experiments averaged over the set of 99 Celebrities (see Section 3.6 for more details). (Left) Comparison using *Average Rank Distance* metric. (Center) Comparison using *Best Rank Distance*. (Right) Comparison using *Top 25 Precision*. The best performance is consistently obtained when using 7×7 patches in the case of raw intensity, and 13×13 patches when using image gradients.

2.4.1 Facial Feature Representation and Size

For each detection accepted by the previous steps, we normalize the Viola-Jones bounding box to a fixed size of 80×80 pixels. We rectify variations in orientation by aligning the eye corners to a same position for all images. We characterize each feature in a face by a patch of variable size extracted around the feature location. The set of patches describing the facial features detected in each face are converted to a vector $\vec{x}_i$ corresponding to image i.

The choice of the size of the patches for feature representation is a trade-off. If patches are too small, the representation is sensitive to artifacts in the image, and not discriminative enough as the patch fails to capture enough of the local texture around the feature of interest. Conversely, if the patches are too large, features include too much detail specific to a particular image and have poor generalization properties. In this section, we investigate the influence of the patch size on the recognition performance.

For this section and Section 2.4.2 only, for the sake of speed we did not optimize the system with respect to a distance metric Φ nor reduce dimensionality with PCA. Rather,

we used the raw L_1 and L_2 distances between patches. As a consequence, the performance reported in this experiment is lower than the results in Section 3.6. Figure 2.6 describes the results of our experiments. The performance of patch sizes from 3×3 pixels up to 23×23 was computed for the three score measures defined. We computed the score with L_1 and L_2 distances, both when sampling patches from the raw intensity image and when sampling them from the gradient image. Overall, the best patch sizes when using raw image intensity were 7×7 and 9×9 pixels, while larger patches performed better when using gradients (13×13 and 15×15 performed best). Note that these patches are always extracted after the image has been rectified for orientation and resampled to 80×80 pixels. The overall best performance was obtained with the L_1 norm and gradient values. The superior performance of gradient images is no surprise our images show a huge variability in lighting conditions and gradient images some invariance with respect to lighting.

2.4.2 Evaluation of Face Subparts

One question that arises naturally is: Which features in the face are most important for recognizing a specific person? In an attempt to answer this question, we investigated the performance of various subparts of the face on a simplified experiment (see Figure 2.1).

Features were extracted using 7×7 patches and intensity values. Faces were characterized by various combinations of features. One experiment focused on the sets of features that form face parts: eyebrows, eyes, nose, and mouth. The other experiment focused on individual features: outer corner of eyebrows, inner corner of eyebrows, top of the eyes, outer corner of eyes, inner corner of eyes, bottom of the eyes, sides of the nose, tip of the nose, top of the mouth, sides of the mouth and bottom of the mouth. For features which occur symmetrically on both sides of the face, both left and right feature were included together. For example, the left side of the mouth and the right side of the mouth were included together.

Figure 2.1 shows the performance for the 'Top 25 detection' (see Section 2.3.1) criterion, color-coded by decreasing performance. Panels a) and c) indicate that as expected, face structures perform better than individual features. In particular, the most successful face structures are the eyebrows and the eyes. This was the case both when considering face parts and when considering individual features. Interestingly, this is consistent with the human performance results from [Sea06], where Sinha reports that eyebrows are among the

most important features for the human face recognition task.

2.5 Data Sets

In creating our data-sets we attempted to mimic the actual statistics and variability encountered in real-world images contrary to more controlled data-set such as the CMU PIE face data-set or the Yale Face data-set. In particular our faces contained the typical variability found on the Web: large variations in lighting, not aligned, varying resolution (our resolution varied from about 100×130 to about 500×800). One caveat is that we attempted to limit the amount of pose variability to roughly frontal images as can be seen in Figure 2.3.

We collected a data-set of 99 individuals from the Web which were mostly celebrities as they tended to have a number of images available. Hence we call this data-set the 99-Celebrity data-set. We ensured that each individual had a minimum of 10 images (although some had as many as 16). We also collected a background set of 200 images which were used to assess the performance of our system (see Table 2.1) for a description of the size of the data-sets. Figure 2.3 shows the typical variability of our faces.

2.6 Learning a Distance Metric

In the previous section we have described how to automatically detect a face and extract features of the face. We now take an additional step in suggesting that each component of our feature vector should note be weighted equally, i.e. certain parts of the face may be more important for recognition than other (Figure 2.1 suggests this is a reasonably intuition as different facial features are better and worse for recognition). The natural question arises, how do we weight these features? We proceed by *learning* a metric which increases performance on recognition tasks.

2.6.1 The Relative-Rank Distance Metric

Our goal is to learn a distance metric between any two images such that images of the same person are deemed close to one another, while images of different people are farther from one another. More formally consider each celebrity indexed by c and all feature representations $\vec{x}_i^c$ for an image i in class c. The following cost function follows naturally from our the

criteria just mentioned:

$$\mathcal{C} = \sum_{c} \sum_{i,j \in c} \sum_{k \notin c} \text{Dist}(\vec{x}_i, \vec{x}_j) - \text{Dist}(\vec{x}_i, \vec{x}_k) \tag{2.1}$$

Qualitatively, if $\mathcal{C}$ is less than zero then we are doing well on average.

Let us consider the distance $\text{Dist}(\vec{x}_i, \vec{x}_j)$. Now we assume that we can learn some arbitrary mapping function $\phi(\vec{x}_i)$ which projects each feature vector from $\mathcal{R}^P \rightarrow \mathcal{R}^Q$. The we can rewrite our distance function using the kernel, $\text{Dist}(\phi(\vec{x}_i), \phi(\vec{x}_j))$. The L_1 and L_2 distances between mapped feature vectors are written as follows:

$$L_1 : Dist(\phi(\vec{x}_i), \phi(\vec{x}_j)) = \sum_{r=1}^{Q} \left| \phi(\vec{x}_i) - \phi(\vec{x}_j) \right| \tag{2.2}$$

$$L_2 : Dist(\phi(\vec{x}_i), \phi(\vec{x}_j)) = \sqrt{\sum_{r=1}^{Q} \left(\phi(\vec{x}_i) \phi(\vec{x}_j) \right)^2} \tag{2.3}$$

The above mapping ϕ can be an arbitrary function. However, if we consider ϕ to be a linear function (i.e. a matrix Φ) and we let $M = \Phi\Phi^t$, then we can re-write the squared L_2 distance as:

$$L_2 : Dist(\phi(\vec{x}_i), \phi(\vec{x}_j)) = (\vec{x}_i - \vec{x}_j)^t M (\vec{x}_i - \vec{x}_j) \tag{2.4}$$

Now consider again Equation 2.1. We would like to minimize this function. We proceed by using the conjugate gradient method which requires that the derivatives of the cost function w.r.t. Φ for the L_1 and L_2 distances.

For the L_2 distance, we consider the simpler task of computing derivatives of the squared cost function w.r.t. $M = \Phi\Phi^t$:

$$\frac{\partial}{\partial M} (Dist^2_{L_2}(\phi(\vec{x}_i), \phi(\vec{x}_j))) \tag{2.5}$$

$$= \frac{\partial}{\partial M} \left((\vec{x}_i - \vec{x}_j)^t M (\vec{x}_i - \vec{x}_j) \right) \tag{2.6}$$

$$= (\vec{x}_i - \vec{x}_j)^t M (\vec{x}_i - \vec{x}_j) \tag{2.7}$$

For the L_1 distance, the component (p_0, q_0) of the derivative is

$$\frac{\partial}{\partial \Phi_{p_0 q_0}} \sum_p |\sum_q \Phi_{pq} a_q| = \frac{\partial}{\partial \Phi_{p_0 q_0}} |\sum_q \Phi_{p_0 q} a_q| \tag{2.8}$$

$$= a_{q_0} \cdot sign(\sum_q \Phi_{p_0 q} a_q) = a_{q_0} \cdot sign((Ma)_{p_0}) \tag{2.9}$$

where we used the simplifying notation $a = x_i - x_j$. This can be written as the product of two vectors:

$$\frac{\partial}{\partial M}\big(Dist_{L_1}(Mx_i - Mx_j)\big) = \begin{pmatrix} sign(y_1) \\ . \\ . \\ . \\ sign(y_Q) \end{pmatrix} (x_i - x_j)^t \tag{2.10}$$

where $y = M(x_i - x_j) = Ma$. Note that there is a measure zero set when the derivative is undefined, namely when $\Phi(vecx_i) = \Phi(\vec{x}_j)$. If all feature vectors $\vec{x}$ are unique this can only be satisfied when Φ is not full rank. We never encountered this situation in practice.

Finally consider again Equation 2.1. Depending on the task at hand we may want to change how much we penalize the difference, $\mathcal{U} = \text{Dist}(\vec{x}_i, \vec{x}_j) - \text{Dist}(\vec{x}_i, \vec{x}_k)$, when it is violated. For instance if we are interested in the *Closest Rank Image* we may not want to penalize heavily inequalities which result in large positive values of U, while, on the contrary if we are interested in the *Average Rank Distance* we would want to penalize large positive values of U. We can include a non-linearity into our cost function with this desired behavior.

$$\mathcal{C}(x) = e^{\frac{x}{\alpha}} \tag{2.11}$$

Increasing the value of α reduces the influence of large positive values of U while decreasing α increases the influence of large U values. We ran experiments using various values of α shown in Figure 2.7.

The cost function is optimized using the conjugate gradient algorithm and usually converges after 100 iterations. We restart the algorithm multiple times (3×) to avoid local minima.

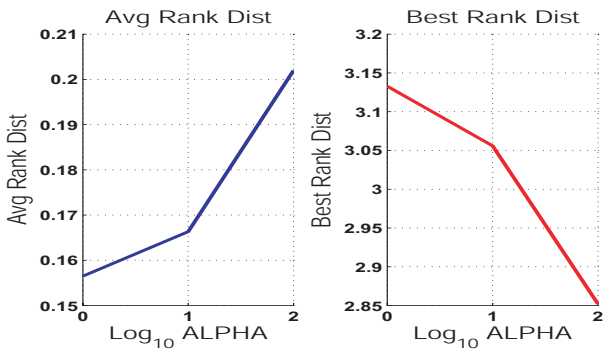

Figure 2.7: The effect of varying the steepness of the slope for our exponential cost function $\mathcal{C}$ on two different performance metrics. (Left) Effects on Average Rank Distance Metric. Note that decrease in performance as we increase α. (Right) Effects on Best Rank Distance. Note that performance *increases* as we increase α. Increasing α results in the optimization giving equal weight to incorrect relative distances which are very far from one another and very close to one another. This intuition is consistent with the results shown in the plot. Results shown from using an L1 metric on 7×7 extracted intensity patches.

2.6.2 Creating Triplet Distances

Now consider again Equation 2.1 and the process of optimizing the function. Which images to we assign as being close and which images are farther from one another? Consider a celebrity c and all images i of this celebrity. We enforce that images of the same celebrity, j are always closer to one another than to images of other celebrities which are indexed by k. By enforcing this criteria we generate a large list of 'triplets' of the form $[ijk]$ which specify that image $\vec{x}_i$ and $\vec{x}_j$ should be closer to one another than image $\vec{x}_i$ and $\vec{x}_k$. This list of triplets is then used to optimize our cost function.

2.7 Results

We evaluate the performance of our complete system on the two data-sets described in Section 2.5. We use the three performance criteria described in Section 2.3.1. We refer the readers to Figure 2.3 to get a sense for the difficulty of the data-set being used. We did not perform any preprocessing on these images.

2.8 Experiments Using a Learned Distance Metric

We conducted numerous experiments using our learned distance metrics. We compared this performance to both the raw feature representations as well as FisherFaces. In all

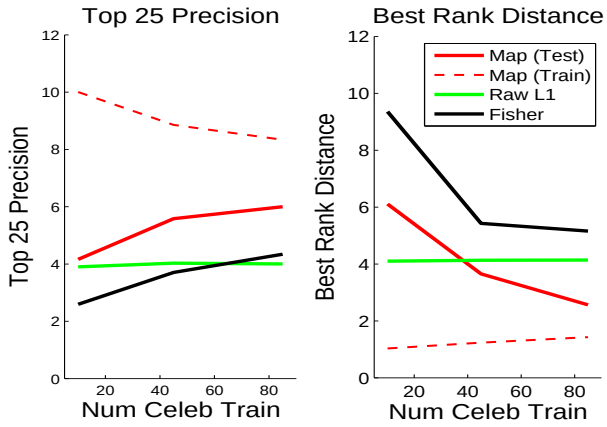

Figure 2.8: Effects of the number of individuals (celebrities) used for training on performance. (Left) Top 25 Precision metric. Results averaged over all celebrities in the test set. An L1 distance metric is optimized using 7×7 intensity patches. The x-axis is the number of individuals used for training ($\times 10$ this number of images are used for training). The y-axis is the precision of the top 25 returned celebrities. Note that the maximum achievable value is 9. Green line is the raw L1 performance of these features *before* mapping. Black line is the performance of FisherFaces [BHK97] when trained using the same number of celebrities. Dotted red line is the performance on the train set of individuals. Solid red line is the performance on the test set of individuals. Note that we are over-fitting: we expect the solid and dotted red lines to converge when we are not over-fitting. Our distance metric is out-performing both the raw L1 distance as well as FisherFaces when we train with 85 individuals. (Right) Same as left plot but using the Best Rank Performance metric. Lower is better. Again we see over-fitting, but our distance metric still far outperforms the baseline methods despite over-fitting. In this case best performance would be 1, which occurs when identical celebrities would always be neighboring one another.

experiments we initially projected our features down to 100 PCA dimensions, and our mapped feature space always contained 50 dimensions, i.e. Φ was a matrix of dimensionality 50×100. We use an $\alpha = 1$ for these experiments.

In Figure 2.8 we varied the number of celebrities we train with in order to understand the asymptotic properties of learning with our distance metric. These plots have two main take-home messages: (1) Our learned distance metric performs well when compared to using either the raw features or FisherFaces (both techniques are widely used in the literature). (2) We are over-fitting as indicated by the distance between the training and test error, indicating that if we trained with more individuals (i.e. collected more celebrities) we may be able to increase performance even further. The over-fitting is the results of the large number of parameters in the projection matrix Φ (5000) and the rather limited set of individuals we train with (we train with up to 85 individuals and a total of about 150 images).

In Figure 2.9 we compare the performance of various feature sets (4 of them) using

Average Rank			Best Rank			Top 25		
Random	Best	Feature	Random	Best	Feature	Random	Best	Feature
0.5	**0.13** (.25)	L1 I 9		**2.7** (4.2)	L2 I 9	1.92	**5.8** (4.8)	L1 I 7

Table 2.2: The best mapping functions. Rand: performance if we chose random images. Best: the performance of our best mapping algorithm. In parantheses the performance without mapping, i.e. on the raw feature vectors. Feat: the feature set used. L1/L2: the distance metric used. I: intensity features. 7/9 the size of the patches used. E.g. 7: 7×7 patches.

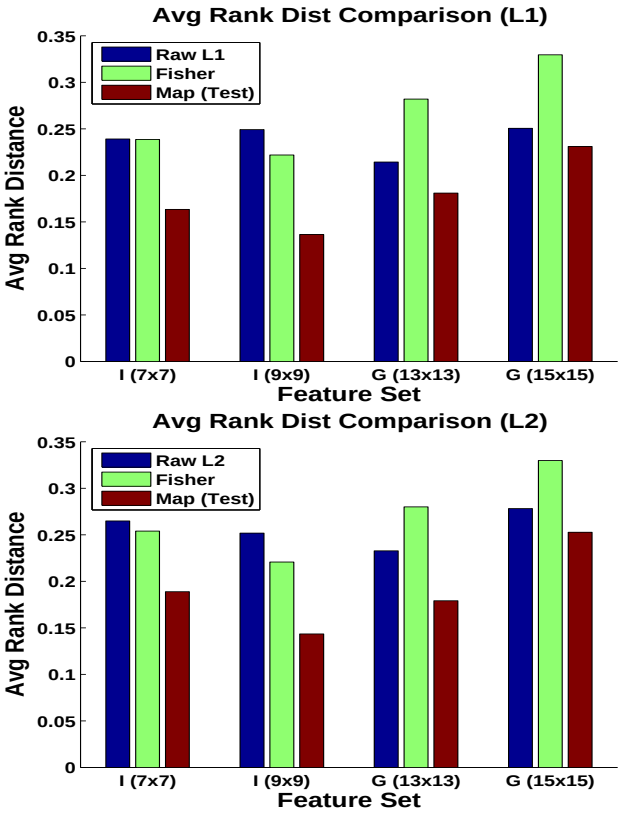

Figure 2.9: Comparison of the Average Rank performance metric. X-axis is 1 of 4 different feature representations: 7×7 Intensity patches, 9×9 Intensity patches, 13×13 Gradient patches, and 15×15 Gradient patches. These were chosen as they performed the best in Figure 2.6. First column is the raw L1 distance performance before learning the mapping. Second column is the performance using FisherFaces. Last columns is the performance using our mapping. We outperform the other metrics in every feature set tried here. Notably gradients perform worse here relative to intensities, see text for a discussion. (Bottom) Same plot as above but using the L2 distance to optimize the mapping and to compute the raw distance. The same trends seem to hold. $\alpha = 1$ for these experiments. Results averaged over 3 iterations.

both L1/L2 distances. We note that using Gradient features yielded the high performance in Figure 2.6, i.e. prior to mapping. While in our experiments using the mapping Φ the intensity yielded the best performance. This is most likely due to the large size of the gradient feature vectors used compared to the size of the feature vectors used with only intensity (the intensity patches extracted were smaller than the gradient patches extracted). Indeed if we analyze the variance of the PCA coefficients obtained when projecting to 100 dimensions, we find that the gradient vectors encompass about 90% of the variance while the intensity vectors encompass about 65% of the variance. The plots in Figure 2.9 indicate that this loss of information has detrimental effects on performance. Note that due to over-fitting, increasing the projected space of PCA dimensions above 100 results in worse

performance as well. If the number of parameters which must be optimized in our map Φ, we will suffer from over-fitting.

In Table 2.2 we show which mappings perform best on all three performance metrics. We also note the large performance gains achieved over not using the mapping, i.e. using only the extracted feature vector $\vec{x}$ in the variance performance metrics.

2.9 Experiments: Google Celebrity Search

Consider if would like to search for a particular celebrity using Goolge image search. Currently Google image-search only utilizes the text around each image to find similar images: *the information in the actual image is completely ignored!*. In this section we show how to extend the tools we developed in the previous sections to increase the efficiency of the Google image searches, in particular when searching for famous people. The reason we concentrate on famous people is that there tend to be many images of them returned for a Google image search.

Our data-set consists of the fist 250 images returned by Google image-search for 20 different celebrities. Many of the returned results do not actually contain the celebrity of interest. In order to evaluate our approach we rate each of the returned faces for each celebrity using the following criteria:

1. If the image contains the celebrity of interest in a roughly frontal view ($+/-20$ degrees from frontal).

2. If the image contains the celebrity but in a strange pose or under heavy occlusion (for instance with a football helmet).

3. If the image does not contain the celebrity of interest.

How can we filter these returned images so that we have a higher precision of our returned results?

We proceed as follows: First we use Viola and Jones (VJ) [VJ01] to automatically detect a face in the image. For those faces for which VJ found a face we then determine whether there are appropriate features using the confidence measure described in Section 2.4. Next we project all faces which pass both VJ and the confidence test into a face space using a map trained on 85 celebrities as described in Section 3.6. At this point we have a subset

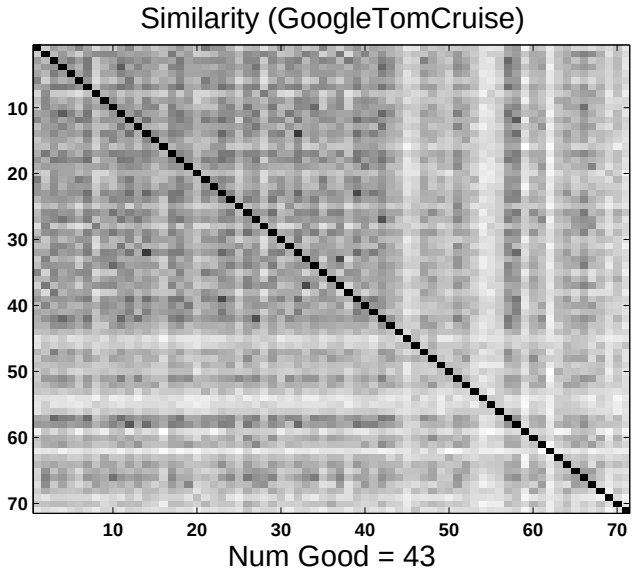

Figure 2.10: How similar are different faces? We performed a Google image-search for 'Tom Cruise'. We filtered out the images which did not contain a face and for which we had low confidence on features (as described in Section 2.9). We extract features automatically around facial key-points. We calculate the L1 distance between all remaining images. We sort the images such that images which have been labelled as having a 'Good' face have indices 1-43 while images which do not have a face are labelled with indices 44-71. Note the faint block structure (with a darker block indicating a stronger similarity). This plot indicates that 'Good' images have a high correlation with one another, such that if we could somehow select images with high correlation we might be able to return the celebrity of interest (presumably the celebrity of interest occupies the largest block).

of the images, each containing a valid face detection which we have projected into a space which provides some invariance.

Figure 2.10 indicates that if we cluster these remaining faces, the largest, most coherent cluster, should contain 'Good' images of our celebrity. How do we cluster? Consider that we have N available images. We seed our clustering algorithm by finding the set of K images which have the smallest average distance between them. We then proceed by greedily adding new images to the cluster that have the minimum average distance between all the images already contained in the cluster. This method, although simple, produces good results in practice. We typically set $K = 3$ for our experiments and N is typically in the range of 50-100. We note some changes in performance when setting $K = 2$ although the qualitative results when viewing the returned images look comparable.

2.9.1 Google Image Search Results

We would like to compare the performance of our algorithm with the raw results returned from Google. As described above we have searched for celebrities on Google and returned the top results. Figures 2.11, 2.15, 2.16, and 2.17 illustrate both the raw returned results from Google as well as the returned results after our algorithm is applied. These figures give us a qualitative feel for how well our algorithm is performing and it seems as though we increase the effectiveness of our search drastically in these cases (other celebrities have similar results). It is also obvious from these returned results that there are many duplicates found which ideally would be removed from the returned results although were not for these experiments to highlight the differences in the quality of returned results between the different techniques.

How can we quantify performance? In particular how much does our perceptual map buy us at the end in terms of an increase in precision? Does it buy us anything at all? In Figure 2.12 we describe the precision criteria we use when evaluating the returned results for 'Nick Lachey'. Notice that overall using our algorithm with the mapping at the end yields the highest precision performance. We show the same plots for all searched celebrities in Figure 2.13. Note that these performances are all based on user ratings: they may not reflect the overall quality of the algorithm accurately. In Figure 2.14 we compare directly the performance before and after the perceptual map averaged over all experiments and we see that the map does indeed perform better than not using the map. We also show how well the map compares to using the raw returned images from Google and here we see a very large increase in performance.

2.10 Discussion

We demonstrate impressive recognition results on a challenging data-set of facial images. In particular we show how to use our system to dramatically increase the efficacy of searching for images of a particular celebrity online. There are numerous avenues for further exploration which include other cues used for facial recognition. For instance the hair is a very useful feature as shown remarkably well in [Sin02] where one confuses Al Gore for Bill Clinton by mapping the hair from one to the other. In addition, the work could be extended to encompass larger variations in pose by learning a more powerful distance metric. Our

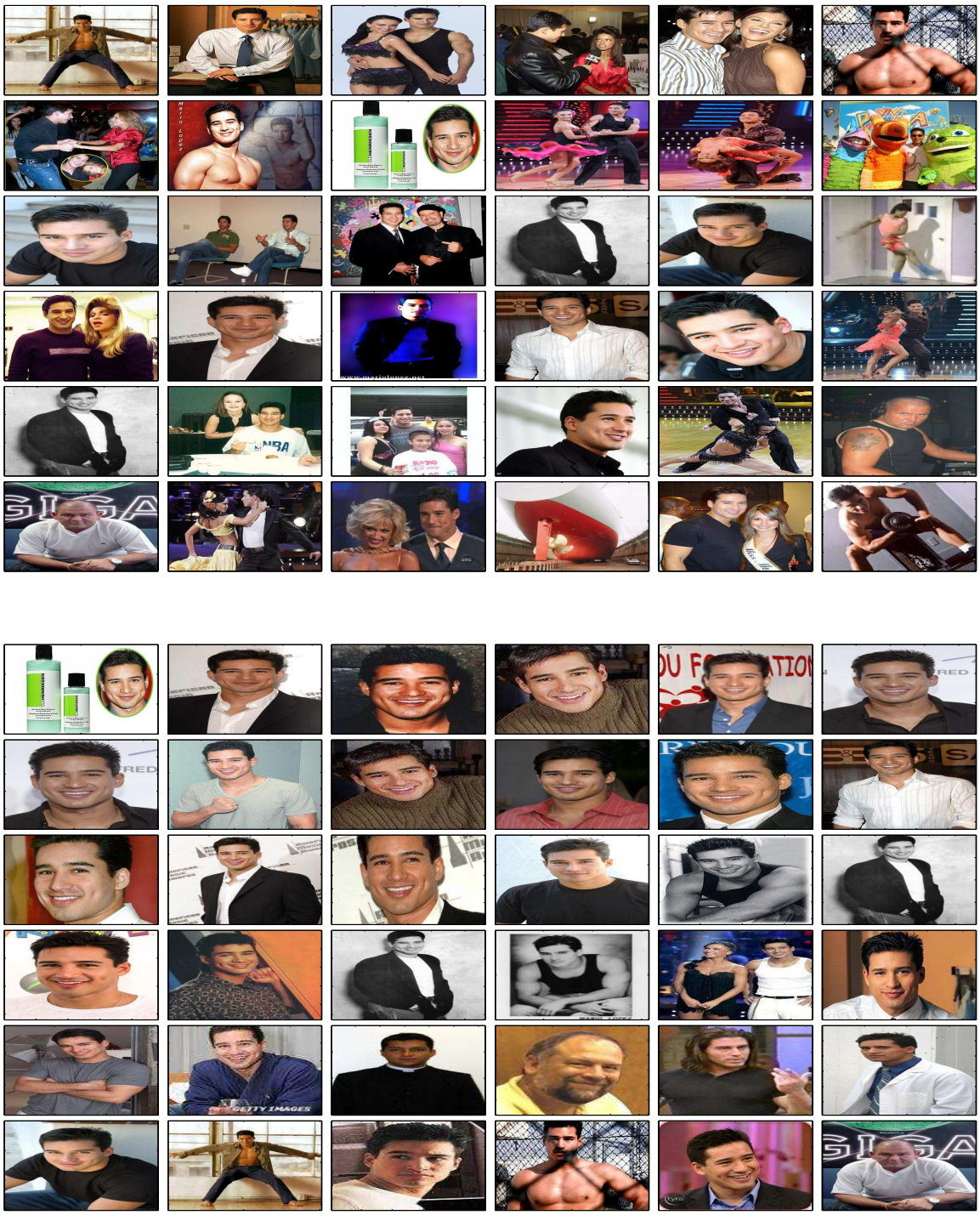

Figure 2.11: Google Image Search: Mario Lopez. (Top) Raw returned results from Google. (Bottom) Returned results after our algorithm. Note that there seems to be more consistency in the returned results on the bottom. Returned results are row-wise.

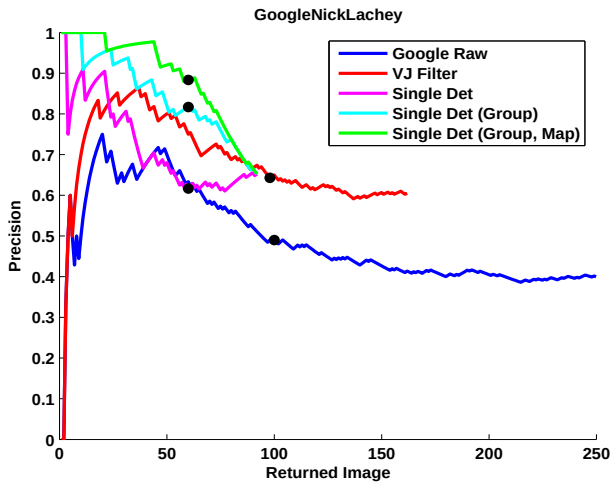

Figure 2.12: Precision using different recall criteria when Googling for 'Nick Lachey' (a famous pop singer). X-axis: the number of images recalled. Y-axis: the precision of those recalled images, how many of the recalled images have been labelled as 'Good'. Blue line: raw Google performance. Notice that this has the worst recall. Red Line: recall after all images that do not contain at leat one face which passed our confidence test. This tends to perform slightly better. Purple: precision when we only consider images where there is a single VJ detection which passed the confidence test. Magenta: precision after we perform grouping on the same set as purple. Green: precision after we apply a mapping trained on a distinct set of 85 celebrities. Note that Green performs the best in terms of precision, which is substantiated by the subjective experience of viewing Figure 2.17. The solid black dots on each line indicate the maximum number of 'Good' examples for each condition.

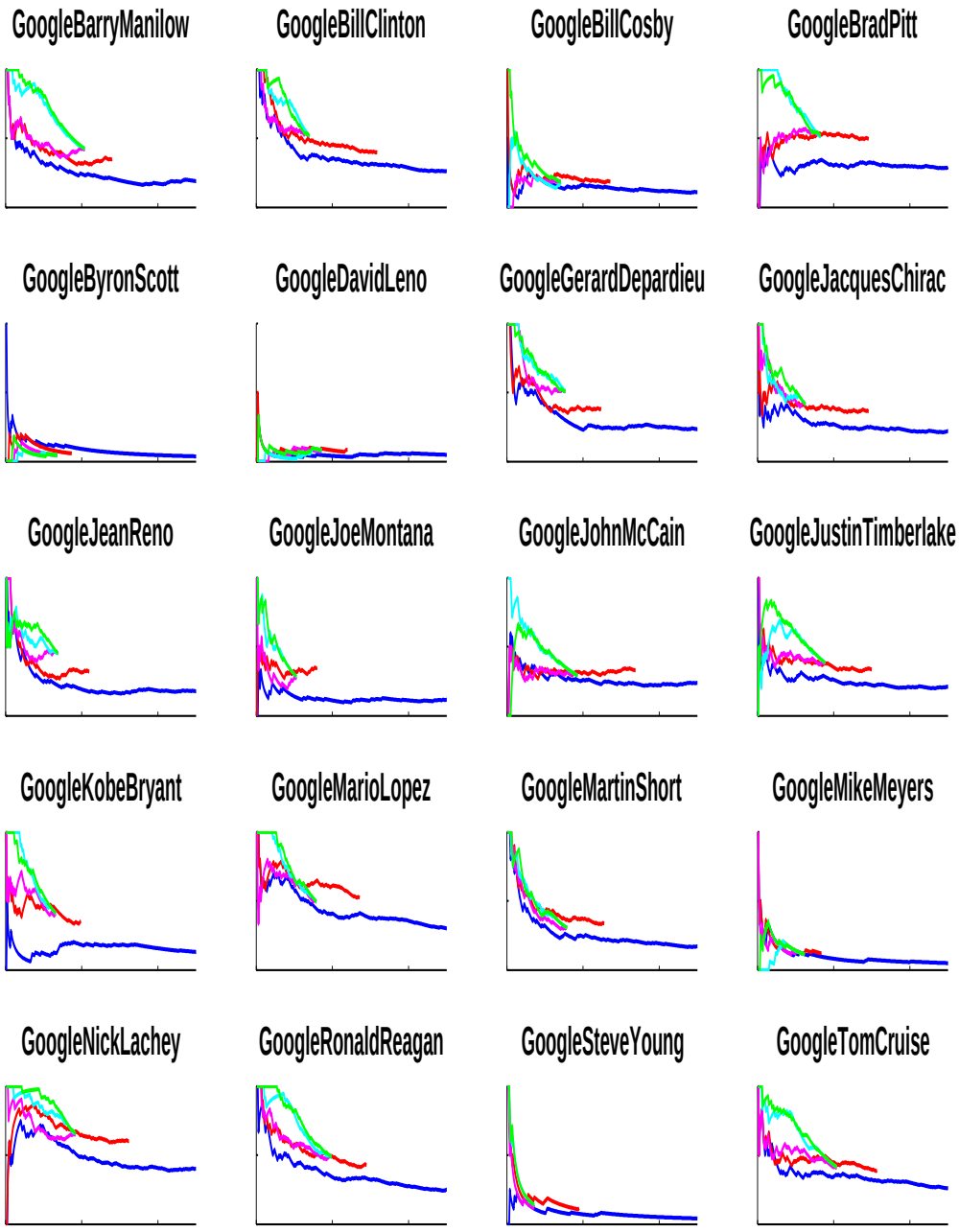

Figure 2.13: Same as Figure 2.12 but for all 20 searched celebrities. Note that in general our algorithm does much better than the baseline.

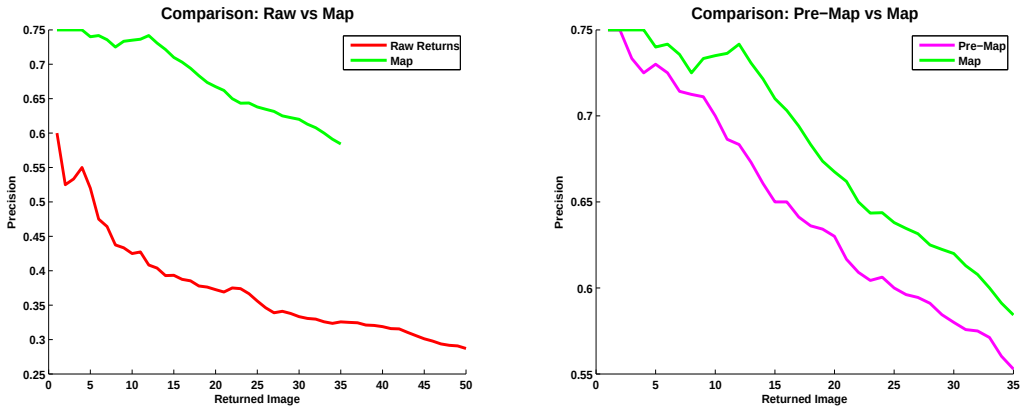

Figure 2.14: Detailed comparison of different precisions averaged over all celebrities. (Left) Comparison of the raw precision by typing in a celebrity name. Here we see a dramatic increase in precision performance. (Right) Comparison of precision before and after the mapping. Note the increase in performance when using the mapping compared to not using any mapping. These plot is averaged over all celebrities shown in Figure 2.13.

current system was only trained on more-or-less frontal poses, presumably we could also train our system on facial images which exhibit more pose variability. Finally, it would be interesting to examine if our system can be trained to learn other ideas such as facial *similarity*. This idea of finding perceptually similar faces will be pursued in the next section.

2.11 Appendix

Here we show more examples of both the raw returned results from Google searches and the results after we have applied our algorithm and the perceptual mapping. Figures 2.15, 2.16, and 2.17 all illustrate these results.

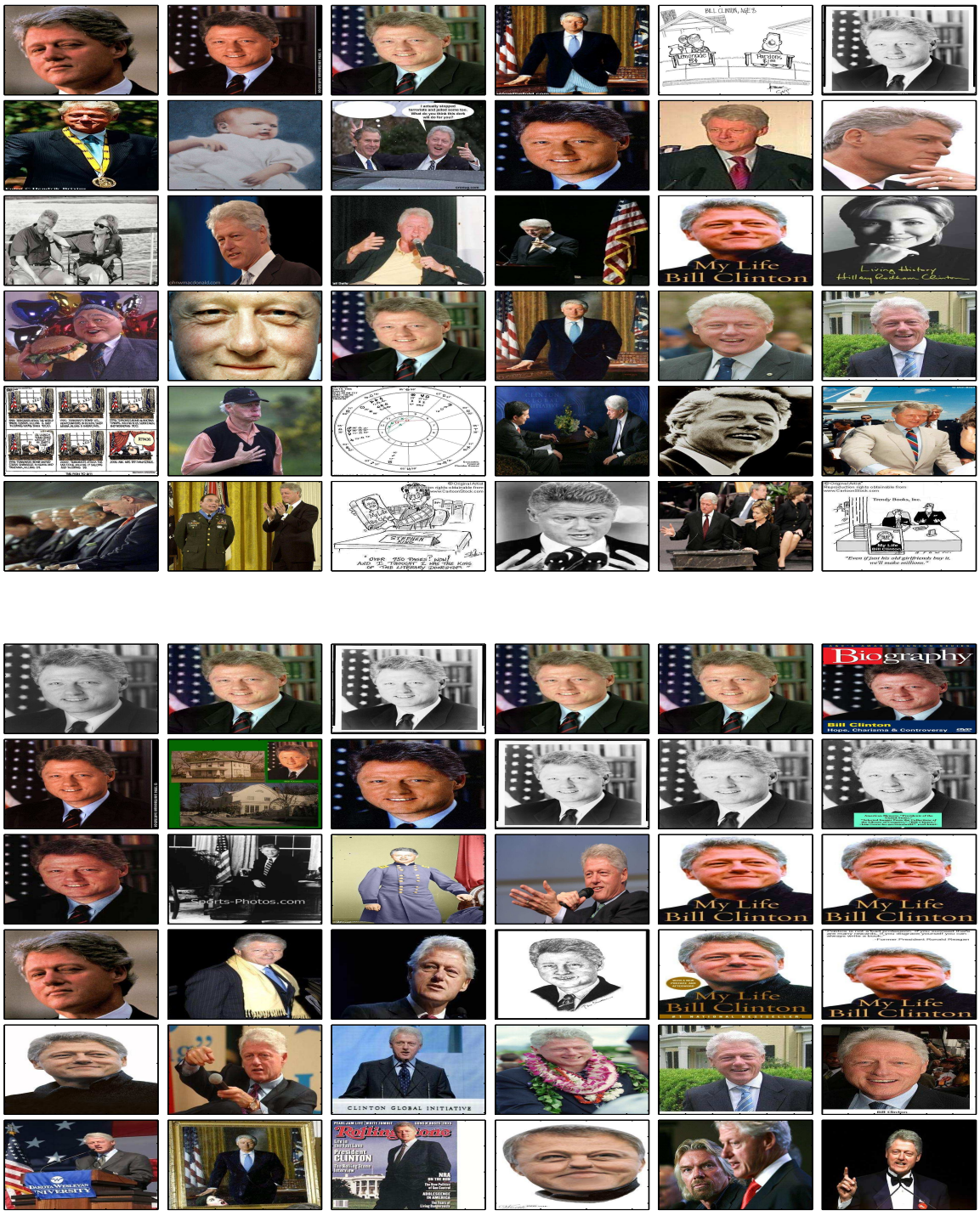

Figure 2.15: Google Image Search: Bill Clinton.

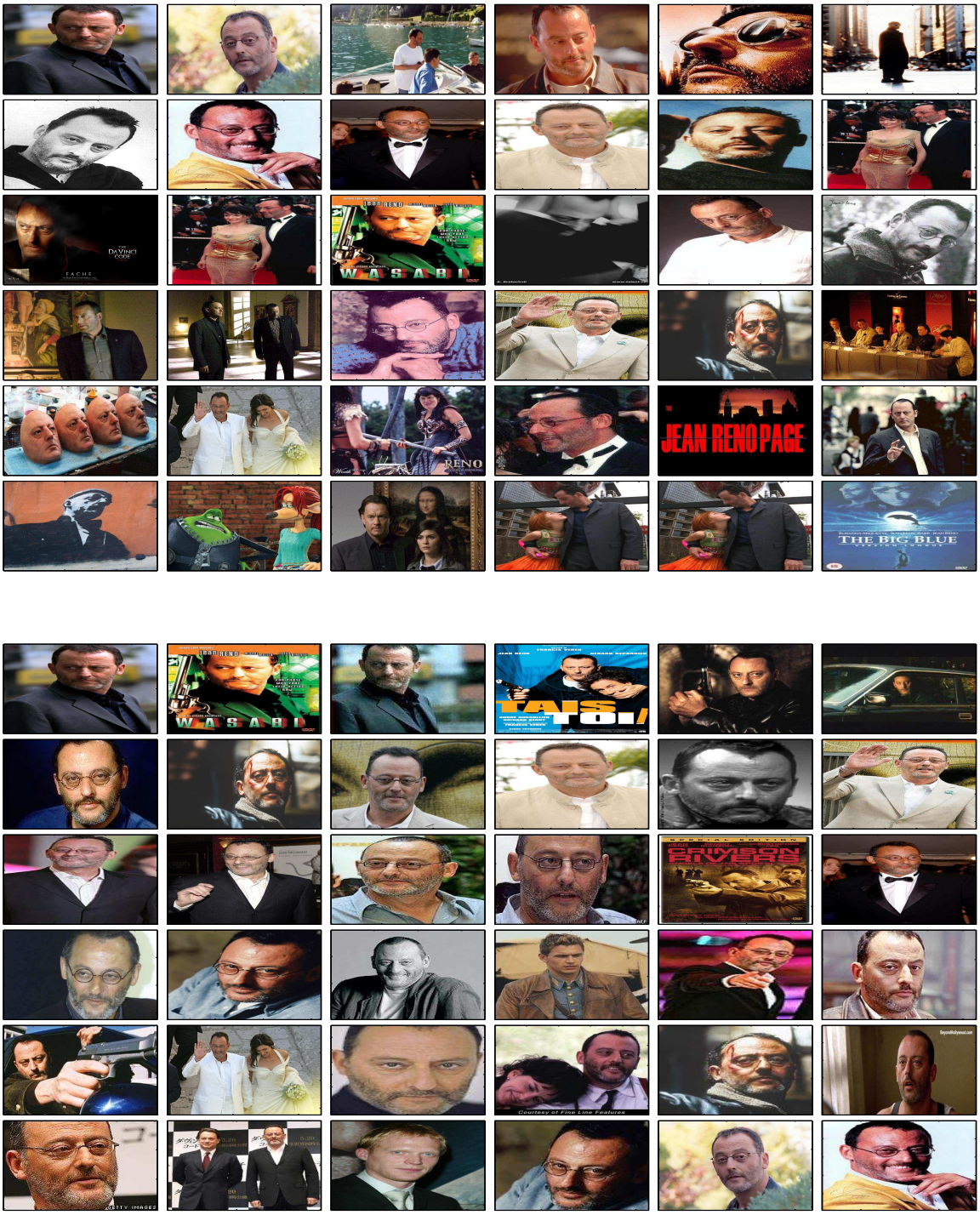

Figure 2.16: Google Image Search: Jean Reno.

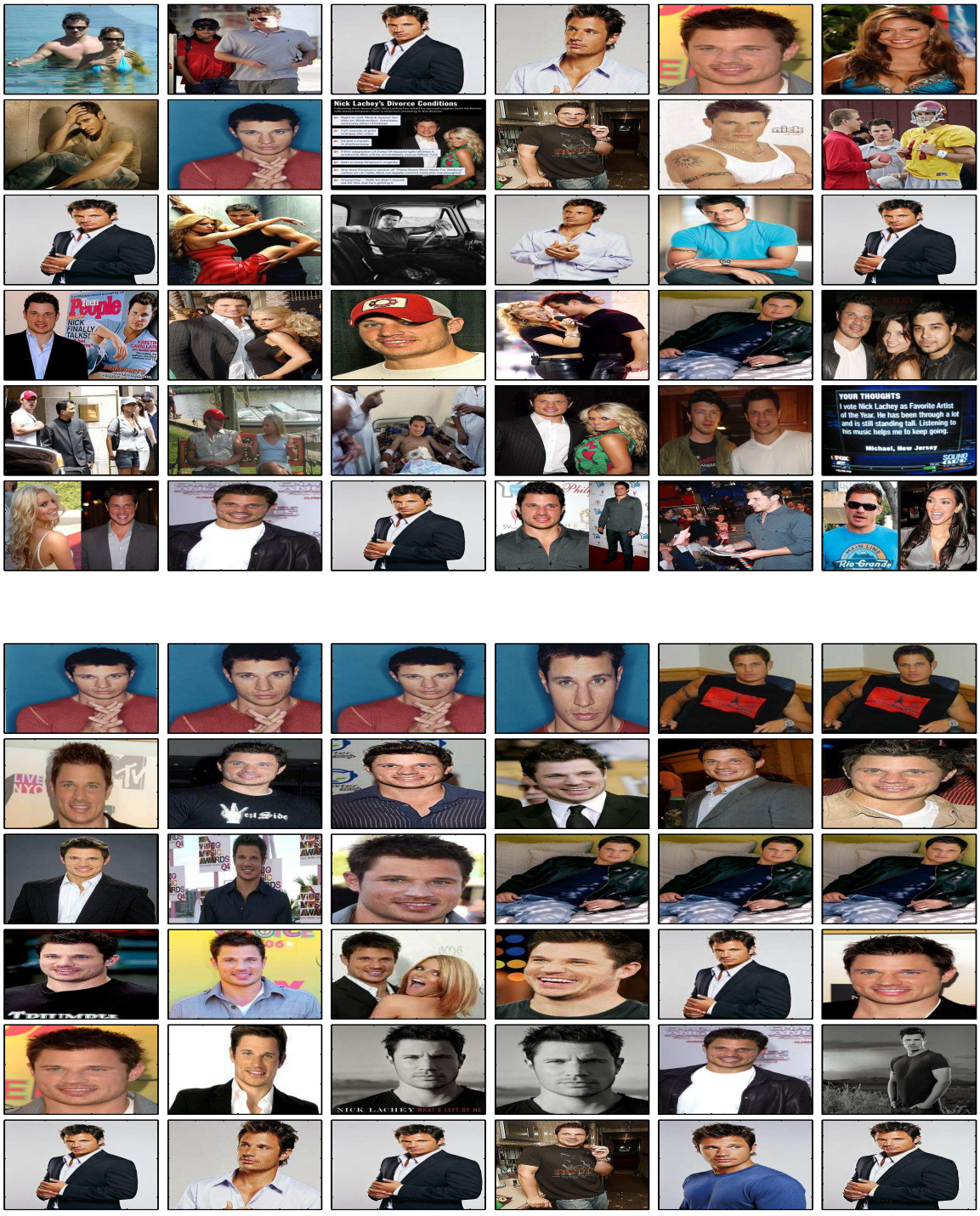

Figure 2.17: Google Image Search: Nick Lachey.

Chapter 3

Finding Similar Faces

3.1 Abstract

Automatically determining facial similarity is a difficult and open question in computer vision. The problem is complicated both because it is unclear what facial features humans use to determine facial similarity and because facial similarity is subjective in nature: similarity judgements change from person to person. In this work we suggest a system which places facial similarity on a solid computational footing. First we describe methods for acquiring facial similarity ratings from humans in an efficient manner. Next we show how to create feature vector representations for each face by extracted patches around facial key-points. Finally we show how to use the acquired similarity ratings to learn functional mapping which project facial-feature vectors into **Face Spaces** *which correspond to our notions of facial similarity. We use different collections of images to both create and validate the Face Spaces including: perceptual similarity data obtained from humans, morphed faces between two different individuals, and the CMU PIE collection which contains images of the same individual under different lighting conditions. We demonstrate that using our methods we can effectively create Face Spaces which correspond to human notions of facial similarity.*

3.2 Introduction

Humans naturally perceive the similarity between different objects. Humans are especially sensitive to facial similarity and it has been suggested that individuals seek partners with similar facial attributes [Hin89]. Facial similarity is particularly useful in social situations such as determining familial relationships or dating preferences. The goal of this work is to place facial similarity on a solid computational footing. We suggest to learn functions

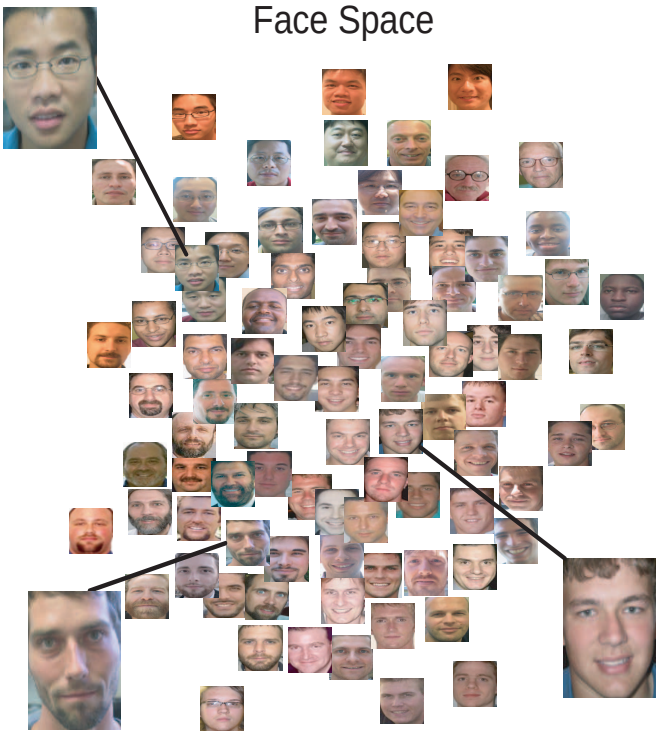

Figure 3.1: Example of typical Face Space. Perceptual ratings from 130 images were used to generate a linear map. $\mathcal{R}^{\mathcal{D}_{\text{MAP}}} = 100$, $D_{\text{PCA}} = 200$. MDS was performed to embed projected faces into a 2D space. Notice that some areas do not conform well to notions of similarity, such as the upper right of the space. Other areas contain groups of similar-looking faces (these areas are highlighted by enlarged images), such as 'Asian', and 'Men with Beards and Sunken Eyes'. Overall the map seems todo a good job of creating a metric space which preserves notions of facial similarity. A version of this work appeared in [HhLP07].

which map measured facial features to metric spaces in which similar looking faces are near one another.

While there is a vast amount literature devoted to facial recognition, judging similarity is a more subtle and difficult topic. Our challenges include: (A) obtaining reliable facial similarity judgments from human observers, (B) developing 'objective' approaches to similarity to supplement measurements from humans, and (C) automatically mapping measured facial features to a space where the natural metric predicts human similarity judgments.

Within the psychological literature there have been numerous studies on how to create metric spaces for faces which reflect perceptual notions of similarity (see for instance [Ash92]). Multi-Dimensional Scaling (MDS) [Kru64] is often used to embed faces into a metric space based on perceptual judgements of facial similarity. Within these embedded metric spaces, faces which appear similar are nearby one another and thus these spaces are often referred to as *Faces Spaces*. These methods have two major drawbacks: (1) MDS methods are not useful when presented with new faces as MDS does not create an explicit mapping function, (2) The faces used were always in standard poses and constant lighting conditions and do not exhibit the same statistical variations found in real-world images.

The computer vision community has a long history of mapping faces into lower dimensional representations for recognition, such as the 'unsupervised' Eigenface and 'supervised' FisherFace [BHK97] methods. More recently LeCun et al. [CHL06] proposed an interesting supervised non-linear mapping using contrastive divergence learning and a convolutional neural network which they apply to numerous data-sets in addition to face data-sets. The goal of our work is to explore in a principled manner the creation of facial similarity spaces which reflect perceptual notions of similarity. We generate explicit maps from extracted facial features and use mostly *real-world* images and demonstrate our techniques using real data obtained both from the web and personal photo collections. Figure 3.2 gives an overview of our proposed algorithm.

The paper is organized as follows: In Section 3.3 we describe and compare methods for effectively obtaining facial similarity data. In Section 3.4 we describe the various data-collections used. In section 3.5 we show how to use this training data to construct explicit functional mappings from feature space to face spaces. Finally, in Sections 3.6 and 3.7 we will present and discuss our experiments.

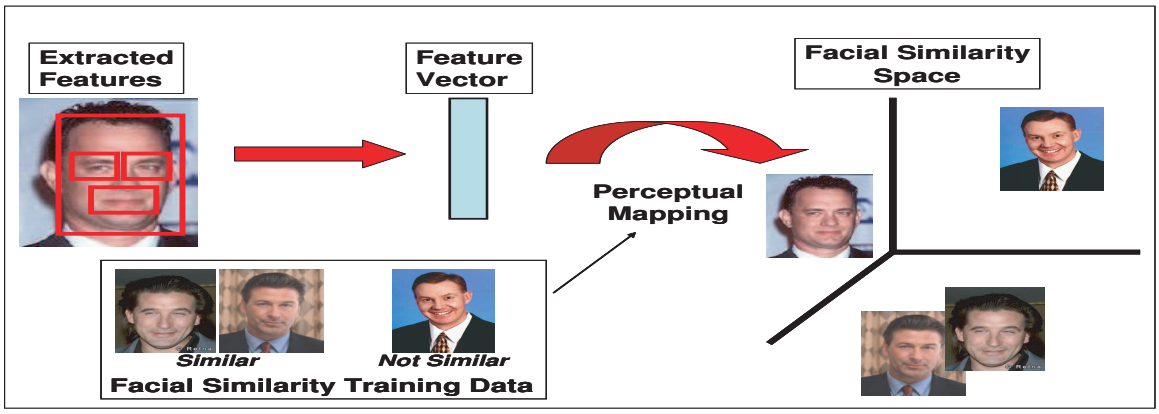

Figure 3.2: Outline of proposed facial similarity algorithm. Patches are extracted from the eyes and mouth and converted to a feature vector. A mapping function is generated using perceptual similarity training data which is used to project the feature vectors to a metric space which reflects human similarity judgements.

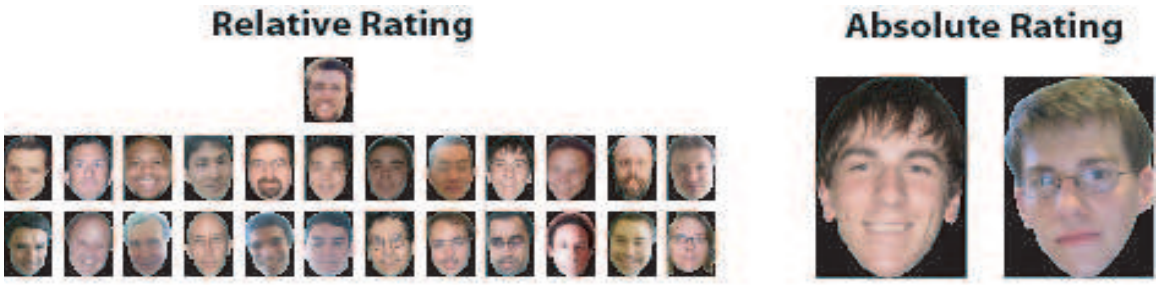

Figure 3.3: (Left) Relative Rating experiment. Subjects are asked to select which of the 24 faces are most similar to the target face located on top. (Right) Subjects are asked to rate, from 1-7, how similar the two faces are. Subjects were given a precise definition of each numerical value, from (7) 'Same Person' to (1) 'Completely Different'.

3.3 Obtaining Facial Similarity Data

A difficult requirement inherent in creating and evaluating a facial similarity space is acquiring large amounts of facial similarity data in an efficient manner. It is desirable that: (1) the similarity measurements be obtained quickly, (2) the measurements be accurate, and (3) the ratings be collected on a large set of faces. Authors have suggested numerous methods for comparing and rating the similarity of faces (see for instance [Rho88] or [HEZP05]) and here we evaluate variants of two common paradigms which we call *Absolute* and *Relative* rating methods. Figure 3.3 describes and compares both the methods.

We compared the two rating methods by asking 5 subjects to rate the facial similarity of 127 face images: 100 'random' face images and 9 sets of 3 images of the same person (the images of the same were photos of minor celebrities). We included images of the same person in order to ensure that each subject was accurately performing the rating tasks (i.e. that when the subjects were presented with two images of the same individual, they would indicate that these images were very similar to one another). It took subjects on average 3s to make an absolute rating and 12s to make a relative rating.

We wish to understand how consistent subjects were in assessing facial similarity. That is, if the subject were asked to make the same similarity judgement 2× would they make the same judgement? For space considerations we have included the analysis of consistency within the Supplementary Materials, but note that consistency within a subject was slightly higher than consistency between subjects. However, consistency between subjects was surprisingly high.

3.3.1 Synthetic Experiments using MDS

Which method, the relative or absolute rating method, is more efficient in creating *Face Spaces*? The relative method yields relative rating information: e.g., Face $\mathcal{A}$ is closer to Face $\mathcal{B}$ than Face $\mathcal{A}$ is to Face $\mathcal{C}$. The absolute method gives the absolute distance between two faces as judged by the subject: e.g. Face $\mathcal{A}$ and $\mathcal{B}$ are distance 2 apart. How can we compare the efficiency of these two rating methods? We proceed by using Multi-Dimensional Scaling (MDS) [Kru64] on both relative and absolute rating data.

First we generate a set of synthetic vectors, where each vector represents a face. From this synthetic data we generate artificial absolute and relative ratings. We then use these artificial ratings to re-create the original vector space using MDS. Next we describe the steps more explicitly.

(Step 1) Generate a random set of N vectors of dimensionality D. Each vector represents a face and the perceptual information available to the subject. (Step 2) Generate the pairwise distances between all vectors which correspond to perceptual distances between faces. (Step 3a) *Absolute Measurements*: distances are discretized into 7 discrete values, [1..7]. An absolute measurement is indicated by the discrete value between two vectors. I.e. if vector (image) $\mathcal{A}$ and $\mathcal{B}$ are 2 apart, then this corresponds to an absolute rating of 2 between these two vectors (images). (Step 3b) *Relative Measurements*: Randomly generate 25 images and set one as the target image as in Figure 3.3. Sort the remaining images by their Euclidean distance to the target image and chose the closest Euclidean image.

From the Supplementary Material we know that subjects are not always consistent in their ratings (i.e when presented with the same set of faces they will not always make the same similarity judgement). We add appropriate noise to the synthetic analysis to ensure that the synthetic responses have the same amount of uncertainty as the subject responses.

Once the sets of absolute and relative ratings have been generated from the synthetic data we perform MDS on both sets of of ratings. For the relative ratings we assign a distance of 1 and 2 to close and far images respectively. We use the Sammon [Sam69] stress criteria for creating the MDS spaces which penalizes most points which are measured as being close to one another (i.e. faces which are perceptually similar) and which are far apart in the

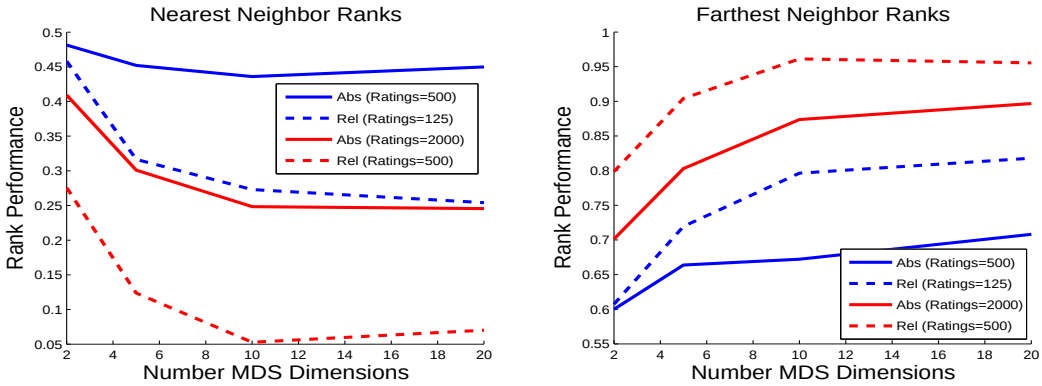

Figure 3.4: Synthetic comparison of Absolute and Relative measurements using MDS. (Top) Nearest Neighbors. For a vector (image) $\mathcal{A}$ we find the nearest neighbor to $\mathcal{A}$ (let this be $\mathcal{B}$). These two vectors (images) are embedded into an MDS space which is created using a set of either absolute or relative ratings. We then calculate the rank distance of these two vectors in the embedded MDS space: $r = \text{rank}(\mathcal{A}, \mathcal{B})$. The y-axis is the average rank distance between all images in the MDS space. The best possible performance would be a value of 1 as this would indicate every nearest neighbor in the original space was a nearest neighbor in the embedded space. A lower rank distance is better. Solid lines: absolute ratings, dashed lines: relative ratings. Different colors indicate different numbers of acquired ratings. The number of absolute/relative ratings were chosen such that they took an equivalent amount of time. The dimensionality of the embedded space is varied on the x-axis. (Bottom) Farthest Neighbor. Same as top but instead of looking at the rank of the *closest* point we look at the rank of the *furthest* point. In this case larger ranks r indicate that the MDS embedding is performing better. 200 random vectors of dimensionality 10 were generated for these experiments. Relative ratings seem to re-create the vector space more effectively over all parameter initializations.

embedded space. Explicitly it minimizes the stress $\mathcal{E}$, where

$$\mathcal{E} = \sum_{k \neq l} \frac{[d(k,l) - d'(k,l)]^2}{d(k,l)} \tag{3.1}$$

and $d(k,l)$ and $d'(k,l)$ is the distance between points k and l in both the original and embedded space respectively. Other stress functions yielded qualitatively similar results. Figure 3.4 compares the two rating techniques and we find relative ratings re-create the original target space more accurately than absolute ratings.

3.4 Data Collections

We wish to create a Face Space by training a mapping function using a set of perceptual training data. After the perceptual space has been generated we would like to evaluate its performance. We use 4 different collections of images, shown in Figure 3.5, to evaluate and create Face Spaces. We describe each below:

Figure 3.5: Different data collections used to train perceptual mappings. (Top Row) Images from two individuals from the PIE collection, note the controlled variations in lighting but little pose variation. (Middle Row) Images of two celebrities, they exhibit some pose, lighting, and facial affect variations. (Bottom Row) Example of three sets of morphed images, center is the morphed image, the sides are the images used to generate the morph. Note that the morphed image appears perceptually similar to the faces used to generate the morph, making this collection useful for training a perceptual similarity map.

Collection	Num Sets	Total Num Images
Perceptual	-	180
Celebrity Morphs	-	52
Celebrity	62	400
PIE	10	270

Table 3.1: Different data collections and the number of images/sets of images in each collection.

Perceptual Measurements We collect perceptual data from subjects by asking two subjects (one Caucasian male and female US native) to rate the similarity of 180 face images using the relative rating technique shown in Figure 3.3.

CMU Pie Collection We selected 10 unique individuals from the CMU PIE Collection [SBB03] and used 27 frontal poses for each individual, which contain variations in lighting and some facial emotions in a controlled environment.

Celebrity Images We wished to obtain multiple images of individuals in more diverse lighting and pose conditions than available in the PIE collection. For this we downloaded 62 sets of images from the web. Each collection had between 3-11 images. These individuals tended to be Celebrities.

Celebrity Morphs[1] We wish to artificially create similar faces using a morphing program. We randomly chose two images of different individuals from our celebrity collection and morphed the images together. We chose the perceptual middle point of the morph when the morph face looked equally similar to the two faces (see Figure 3.5 for examples of the morphs). The generated morphs do indeed look very similar to the two faces which generated them.

3.5 Creating a Perceptual Mapping

Next we discuss how to use to create an explicit mapping from features extracted from face images to a space which conforms to facial similarity.

3.5.1 Facial Feature Representation

First we describe how to obtain a feature (vector) representation for each face. We manually annotated points at the corners and above/below the eyes and mouth. We note that automatic detectors for these particular facial locations have been shown to be successful on similar data [EZ06], but for this study we preferred more controlled data. We extracted patches around both eyes and the mouth at the size shown in Figure 3.6 and resized them to be 17×24 and 25×14 respectively. The patches were combined into one vector and used to represent each face. Each patch was normalized to have zero mean and unit variance. We also conducted experiments which included the area around the nose, but found no

[1] We used the program FaceMorpher Multi by Luxand Development to create facial morphs.

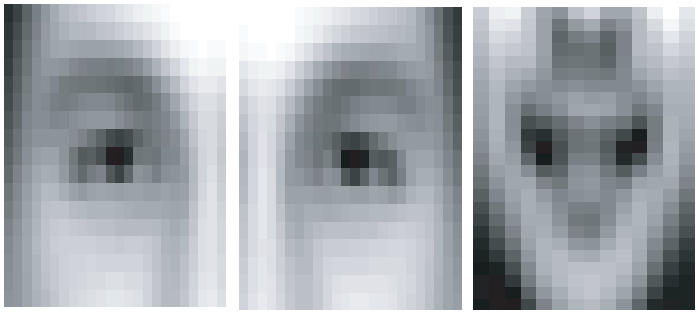

Figure 3.6: Examples of the average visual features extracted from a set of faces. Note the relatively large size of the extracted feature.

noticeable change in performance. The dimensionality of all feature vectors was reduced to $D_{\text{PCA}} = 200$ dimensions using PCA which typically encompassed $> 99\%$ of the variance. We also experimented using kPCA with an RBF kernel to reduce dimensionality and found no noticeable change in performance.

3.5.2 Representing Ratings using "Triplets"

In Section 3.3 and Figure 3.3 we discussed how to obtain relative facial similarity measurements. Here we discuss how to represent relative measurements so they can be used to optimize a perceptual map. Each relative rating tells us that the image which is picked (i.e. Image $\mathcal{B}$) is more similar to the target Image $\mathcal{A}$ than any other image in the set of 24 images, indexed $\mathcal{C}$. We can represent this information using a triplet $[\mathcal{A}, \mathcal{B}, \mathcal{C}]$. For any triplet, the subject has indicated that $\mathcal{D}(\mathcal{A}, \mathcal{B}) < \mathcal{D}(\mathcal{A}, \mathcal{C})$ where $\mathcal{D}$ is some perceptual distance metric used by the subject. It is easy to see that each relative rating generates 23 such triplets.

We can represent data collections containing images of the same person, such as the Celebrities and PIE collections, using triplets as well. We consider that images of each individual are more similar to one another than to any of the other images in any collection. In this case, in the triplet $[\mathcal{A}, \mathcal{B}, \mathcal{C}]$, $\mathcal{A}$ and $\mathcal{B}$ are images of the same person and $\mathcal{C}$ is an image of a different individual. For example if we have 2 images of the same individual, and 100 images of other individuals, we can generate $2 * 100 = 200$ triplets.

We follow similar logic for the morphed images. In this case the morphed image is similar to both of the individuals used to generate the morph, but dissimilar to other individuals. For instance, if we generate a morph from two individuals where we have 2 images of each individual, as well as 100 images of other people, we can generate $4 * 100 = 400$ different triplets.

3.5.3 Perceptual Map

How can we generate a perceptual map which projects the feature vectors extracted in Section 3.5.1 to a space conforming to our notions of similarity? Explicitly, we would like a mapping f which takes feature vectors representing each face of dimensionality D_{PCA} and projects them into a space of dimensionality D_{MAP} such that $f : \mathcal{R}^{\mathcal{D}_{\text{PCA}}} \rightarrow \mathcal{R}^{\mathcal{D}_{\text{MAP}}}$, where D_{MAP} is typically 100. Results are not particular sensitive to small changes in the dimensionality of D_{MAP}: we found qualitatively similar results for $D_{\text{MAP}} = 50 - 200$. Here

we assume a linear map, although the framework is applicable to any differentiable function f. We can represent our mapping function as $\vec{y} = f(\vec{x})$, or, since we are assuming a linear map, $\vec{y} = M\vec{x}$, where M is a matrix of $\mathcal{R}^{\mathcal{D}_{\text{MAP}}} \times \mathcal{R}^{\mathcal{D}_{\text{PCA}}}$, and $\vec{x}$ is a feature representation for a face, and $\vec{y}$ is the projected representation after the linear map.

Consider a particular triplet $t = [\mathcal{A}, \mathcal{B}, \mathcal{C}]$ as described in the previous section. Now consider a feature representation for the images $\mathcal{A}, \mathcal{B}, \mathcal{C}$ which we denote by $\vec{a}, \vec{b}, \vec{c}$. For a particular triplet, we would like that the distance between $\vec{a}$ and $\vec{b}$ be less than the distance between $\vec{a}$ and $\vec{c}$ in the projected space. Thus we would like a cost function which penalizes inequalities when the following is true: $\mathcal{D}(M\vec{a}, M\vec{b}) > \mathcal{D}(M\vec{a}, M\vec{c})$ where M is the projection matrix to be optimized. We use the squared L2 metric to calculate distances in the projected space and penalize inequalities using:

$$S_t = ||M\vec{a}_t - M\vec{b}_t||^2 - ||M\vec{a}_t - M\vec{c}_t||^2 \tag{3.2}$$

$$S_t = (\vec{a}_t - \vec{b}_t)P(\vec{a}_t - \vec{b}_t)^t - (\vec{a}_t - \vec{c}_t)P(\vec{a}_t - \vec{c}_t)^t \tag{3.3}$$

Where $P = M^t M$. The mapping is linear, so the derivative of the penalty term w.r.t. the linear mapping is a matrix and can be written as:

$$\frac{\partial S_t}{\partial P} = (\vec{a}_t - \vec{b}_t)^t(\vec{a}_t - \vec{b}_t) - (\vec{a}_t - \vec{c}_t)^t(\vec{a}_t - \vec{c}_t) \tag{3.4}$$

and if we impose an exponential cost function, C_e and sum over all triplets t:

$$\mathcal{C}_e = \sum_t exp(\frac{-S_t}{\beta}) \tag{3.5}$$

$$\frac{\partial \mathcal{C}}{\partial M} = \frac{1}{\beta}\sum_t exp(\frac{-S_t}{\beta})\frac{\partial S_t}{\partial M} \tag{3.6}$$

where β controls the sensitivity of the penalty (we usually set to $\beta = 1$ for our experiments), i.e. the steeper the exponential distribution (smaller β), the more we penalize triplets in which the inequality is not respected. We considered other cost functions including a sigmoid function, C_s:

$$C_s = \sum_t Sig(-S_t) \tag{3.7}$$

where $Sig(\text{x}) = \frac{1}{1+exp^{-x}}$. We also considered the cost function proposed by the authors

	Exponential	**Sigmoid**	**Rank [BCB98]**
20 Celeb Train	$.25 \pm .04$	$.25 \pm .03$	$.27 \pm .04$
40 Celeb Train	$.2 \pm .03$	$.21 \pm .03$	$.23 \pm .04$
60 Celeb Train	$.17 \pm .04$	$.18 \pm .05$	$.20 \pm .04$

Table 3.2: Comparison of different cost functions when different numbers of celebrities are used for training. Rank performance on celebrity data-sets shown, see Figure 3.9 for details on calculating rank performance. Lower is better. Averaged over 20 iterations. The exponential cost seems to perform the best.

of [BCB98] derived from the statistical estimate of the rank correlation between two variables. We found the exponential cost function to, in general, yield slightly better results for most simulations (see Table 3.2 for a comparison of different cost functions). In addition we experimented using a non-linear one-layer Radial Basis Function network as our mapping function. These mappings exhibited severe problems with over-fitting due to the large number of parameters in our map and the relatively small amount of training data available and hence we used the linear map M for our experiments.

We optimize the map using the conjugate gradient algorithm. We initialize the matrix M at multiple starting points to avoid local minima and chose the experiment resulting in the lowest cost. We find that convergence usually occurs after 100 iterations. We witnessed only minor fluctuations in cost due to local minima during optimizations.

3.6 Experiments

In the previous sections we described how to acquire perceptual judgements efficiently and how to generate a perceptual map. In this section we describe our experiments. First, we explicitly indicate how we divide up our training and test set of data so as not to contaminate training data with test data.

3.6.1 Creating Train and Test Sets

Perceptual Ratings: We collected relative ratings on 180 face images which we call this set $\mathcal{E}$. For training, we selected a subset $\mathcal{E}$ as a train set and found all the triplets indexing these training images. The test set consisted of the remaining images and the triplets associated to those test images.

Celebrity/Pie Images: Of the 62 sets of celebrities we chose a subset to train with and

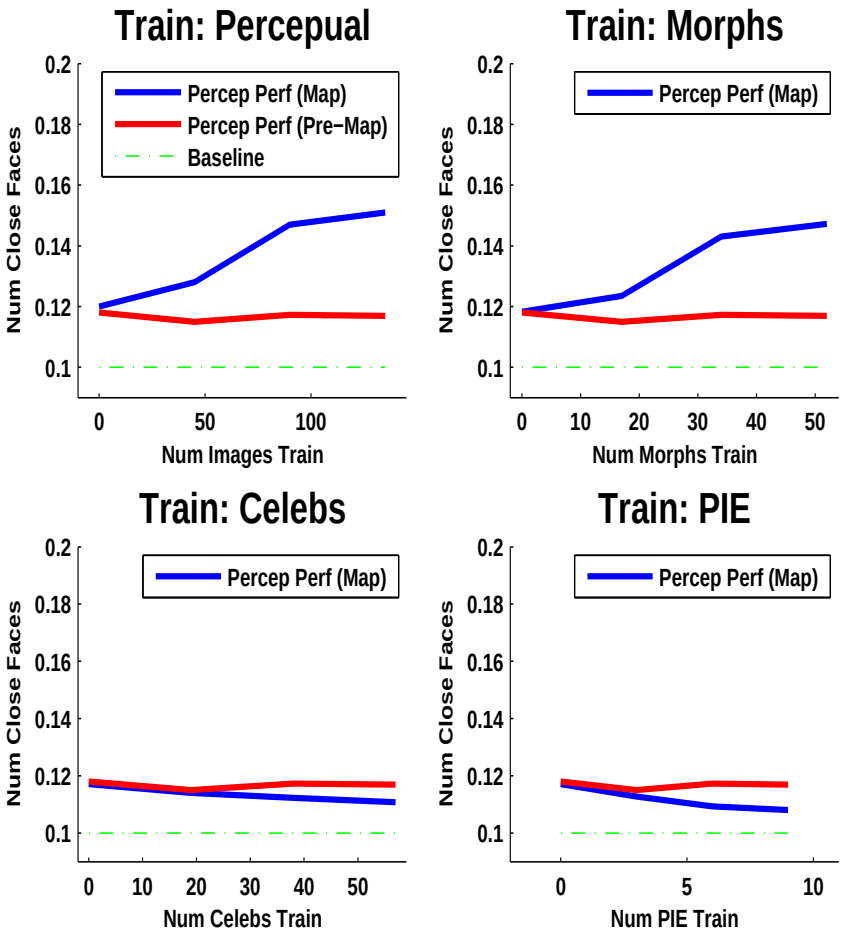

Figure 3.7: How accurately can we reproduce human perceptual judgements? We train mappings using 4 different collections of images (the training collection used to create the mapping is indicated in the title of each sub-plot). x-axis: clockwise from the top left, number of images for which perceptual data was acquired, the number of morph images, the number celebrity individuals, and the number of pie individuals used to train the mapping. y-axis: performance on human perceptual measurements using the metric described in Section 3.6.3. Higher is better. Red line is the performance expected by chance. The best performance is obtained when the map is trained with perceptual data (about 4× better than baseline performance), although morphed images perform well as well. Training a mapping using either the Celebrity and PIE data-sets results in poor performance: these collections do not seem appropriate for training mappings which predict human perceptual judgements.

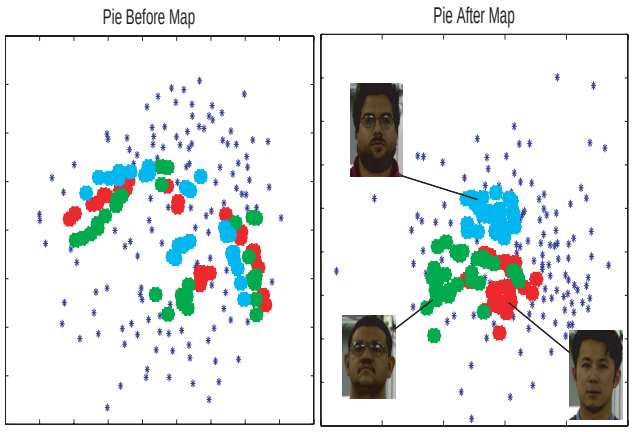

Figure 3.8: Distance between PIE individuals before and after mapping. Note the tighter clusters after the mapping, the right image, than before the mapping, the left image. This mapping was generated by training on 54 sets of *Celebrities*, no PIE images were used for training the map. (Left) The blue stars represent non-PIE images. Each color represents a single PIE individual, and each dot represents 1 of 27 different images of this person under different with different lighting and facial affects. The embedding was generated using MDS on the PCA reduced representations of each feature vector. (Right) The same set of points after being mapped into face space and embedded using MDS. Pictures indicate the identity of each cluster. Note that each PIE individual is now clustered in a particular area of space and that the points representing each individual are now closer to one another. The mapping has learned invariance to lighting and facial affect. This is somewhat remarkable considering that the map was trained on only Celebrities and generalized similarity information across data-sets to the PIE images.

the remainder was used as a test set. 120 additional images from the set $\mathcal{E}$ were used as 'far' images, e.g. in the triplet $[\mathcal{A}, \mathcal{B}, \mathcal{C}]$, $\mathcal{A}$ and $\mathcal{B}$ referenced an image of the same celebrity while $\mathcal{C}$ indexed one of the images from the set $\mathcal{E}$. For PIE experiments, we used 10 sets of PIE people in a paradigm identical to the Celebrity Images.

Celebrity Morphs: Each morph image was generated from two celebrity images and the triplets, $[\mathcal{A}, \mathcal{B}, \mathcal{C}]$ were generated such that $\mathcal{A}$ corresponded to the morphed image, $\mathcal{B}$ to an image of a celebrity from which the morph was generated, and $\mathcal{C}$ an image from $\mathcal{E}$. We used morphs and their associated celebrities for training and the rest of the images for testing.

3.6.2 Assessing Map Performance: Rank

How should we measure performance before and after the perceptual mapping(s)? Since the learned map M can scale the space arbitrarily, a Euclidean distance metric is not appropriate. We chose instead to measure performance using the *rank* between images. The rank r between two images is the number of images in between two images. Consider a set of N face images. Let n_i^c indicate that image i is of celebrity c. Finally let M^c be the number of images of celebrity c in the set N. We can measure the average rank of celebrity c as:

$$r^c = \frac{1}{M^c(M^c - 1) \times N} \sum_{i,j \in c, i \neq j} \text{rank}(\vec{n}_i^c, \vec{n}_j^c) \tag{3.8}$$

3.6.3 Assessing Map Performance: Closest

We would also like to quantify the performance of the perceptual ratings obtained from humans. This is a bit trickier thank using the rank distance. We proceed as follows. For each image $\mathcal{A}$, find the 10% closest L2 distance images. Now consider all triplets of the form $[\mathcal{A}, \mathcal{B}, \mathcal{C}]$. Calculate the percentage of times $\mathcal{B}$ is in the set of closest images. The higher this percentage, the better the metric space is at predicted perceptual judgements. By chance we would expect on average 10% of triplets to have an image $\mathcal{B}$ in the top 10% closest images (a perfect map would yield roughly 19%). See Figure 3.7 for experiments evaluated using this performance metric.

3.6.4 Holistic vs. Feature-Based Recognition

There has been some debate in the psychological community between whether humans recognize faces by processing the whole face at once (holistic processing) or whether they proceed by analyzing individual features (feature-based processing) [NGOL06]. Within the computer vision community both methods have been utilizes, with holistic Eigenface [TP91] and Fisherface [BHK97] methods competing with more local feature-based methods [EZ06].

In this work we initially performed experiments using a global representation based on performing a 3D warp of the face and extracting the entire warped face into a feature vector (this work was done in conjunction with Mark Everingham and Andrew Zisserman of Oxford University, not published). We found that the global representations performed inferior, in general, to feature-based representations. However, it is not entirely clear whether the inferior performance was due to the artifacts generated by the 3D warp or due to the holistic nature of the face representation. (Note that we only performed affine 2D warps on extracted features for the feature-based approach).

One interesting note is that although we chose to use a local feature based approach, the optimal patch-sizes for these features overlapped one another significantly (see Figure 3.6). Smaller features, only centered on a small region of the face and not overlapping one another were inferior, in general, to larger features which covered large swaths of the face and overlapped significantly with one another (see Figure 2.6). Small patch sizes might not accurately represent such notions as facial adiposity or the relative positions of features to one another. In a sense, these are global characteristics of the face. These experiments suggest that utilizing a mixture of both local and global cues seems to result in the strongest overall performance.

3.7 Results

The different collections of data exhibit different statistical variations as shown in Figure 3.5. Not surprisingly, we found that a mapping had the highest success when tested on the same collection as it was trained with (see Figures 3.7 and 3.9): the map learned robustness to the statistical variations within the *training* data-set which it generalized to the *testing* set. Figure 3.1 shows a nice example of a Face Space generated using perceptual data.

There are several interesting observations from our experiments: (1) Training using the

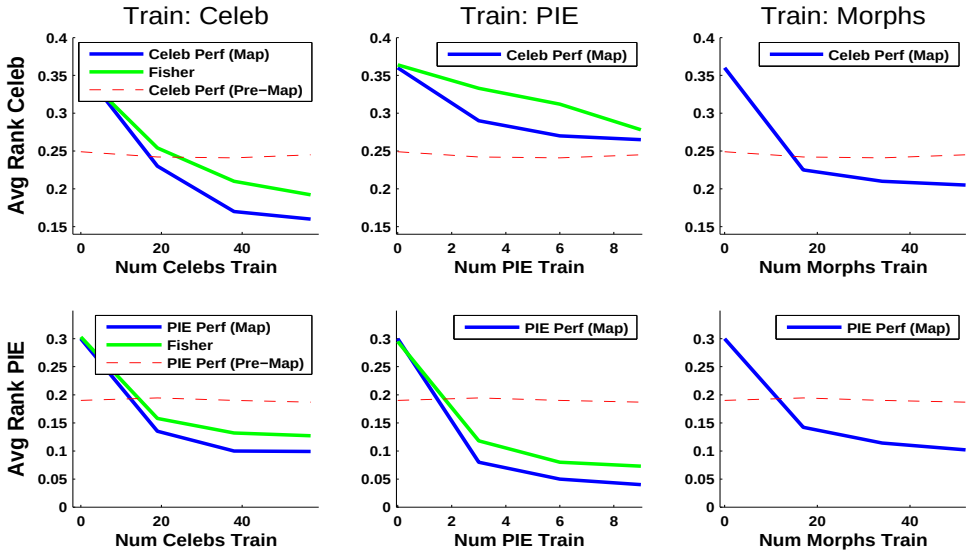

Figure 3.9: Rank performance of both the Celebrity and PIE collections after training mappings using different data. (Top Row) Rank performance of the celebrity collection. X-axis: the number of celebrities/PIE individuals/morphs used for training. Y-axis: the rank performance on the celebrity data-set (lower=bettter). Training with celebrities results in the best performance. Dotted red line is the rank performance before the mappings are applied. Green line is the performance when we train a mapping using Fisher Linear Discriminants (see [BHK97]). Fisher seems to yield slightly worse performance when compared to our mapping. Note that increasing the number of individuals used for training yields better rank performance as the mapping over-fits less and generalizes more. (Bottom Row) Same but measuring rank performance on the PIE data-set. Training on PIE individuals results in the best performance. For both the top and bottom row, training on the same data-set as one tests on yields the best performance. Training using morph images seems to create good mappings for both PIE and Celebrity collections. Note that we create distinct train/test sets for all experiments (see Section 3.6.1). All results averaged 25 times.

morphed images generated Face Spaces which predicted user similarity judgements well (although not as well as training on facial similarity judgements from subjects). However training using either the Celebrity or PIE collections did not generalize to good performance on the human data. The latter two collections exhibit mostly variations in lighting and facial affect and the variability inherent there did not generalize to facial similarity. (2) Our performance curves, although leveling out, do not seem to have saturated, indicating that by acquiring more training data we might be able to improve performance further (see Figures 3.7 and 3.9. (3) The Celebrity collection generalized well to the PIE collection although the PIE did not generalize well to the Celebrity collection (Figures 3.8 and 3.9). Presumably the PIE collection, which was obtained in a highly controlled environment, did not exhibit enough variability to generalize to the other collections.

3.8 Automatic Similarity Detection

In this section we combine the automatic detection techniques described in the previous chapter with the idea of learning a similarity metric described in this chapter. In particular we automatically detect the face, validate that it is a face using the Everingham face model (reject false detection faces), extract features, and finally map the features into a space using a learned similarity metric. Figure 3.11 shows results for a particular query. The results are, in general, strong, with the first closest returned faces containing faces which look similar to the target face. Note that this is a completely automated procedure for finding similar faces such that the user must only take a picture of an individual and run the program to find similar faces. No marking of facial features is necessary.

When did our algorithm yield poor performance? We found experimentally that when the data-base contained relatively few examples of a specific type of face, such as "old-Indian men", the results poor. There were simply not enough faces near the target face and thus the algorithm returned relative nonsense. On the other hands, in dense areas of the space such as "young-white-males" the algorithm performed quite well, often returned many very accurate and similar examples.

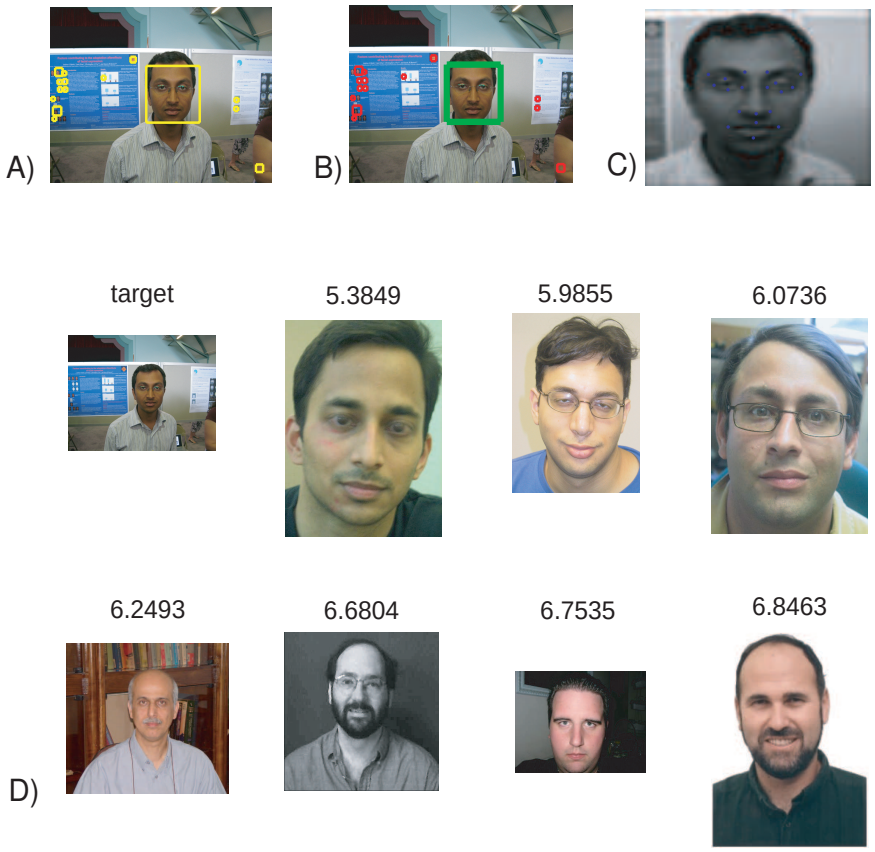

Figure 3.10: Overview and example of automatic facial feature detection and mapping of facial similarity. A) Initial VJ detections of faces. B) Filtered faces using feature validation procedure, Green is the accepted face, red are rejected faces. C) The detected inner facial features. D) The most similar faces to the target faces. Titles are the L2 distances of each face from the target face. Note that the first results are all young Indian men as is the target image. As we move farther away the results become less consistent.

Figure 3.11: Two more examples of target faces and the closest identified faces in the data-base. (Top) The target face is a young Caucasian male and the returned results are nearly all young Caucasian males. (Bottom) The query image is a young asian male and the returned results are nearly all young asian males. The results are impressive in that there was not labeling of the images in any way. The algorithm automatically detected and extracted facial features and mapped these features to a space which obeys our notions of facial similarity.

3.9 Discussion

We have shown how to construct and evaluate facial similarity spaces which mimic human perceptual judgements on real data. In addition we show the flexibility of the approach: training the mapping with different data-collections results in different *Face Spaces*. This work takes the first steps towards creating metrics and mappings for faces which correspond to human perceptual judgements.

3.10 Appendix: Consistency of Raters

We wish to measure the consistency of our subjects during both the relative and absolute rating methods. We had 5 subjects rate images using the relative method and a different set of 5 subjects rate images using the absolute method.

3.10.1 Absolute Ratings

Recall that during an absolute rating subjects are required to choose a number from 1 to 7 indicating how similar two faces are to one another. Figure 3.12 shows the number of times each rating was chosen by each subject. The right-skew in the distribution indicates that subjects are more likely to say faces are dissimilar than similar. In order to assess how consistent subjects were we interleaved trials in which subjects were shown the same two images and required to assess their similarity. If subjects are perfectly consistent they will always indicate the same similarity number (from 1 to 7) between the images. If they are not consistent they will indicate a different number. Figure 3.13 shows results across 5 subjects for both the condition when they see the hair of the faces and when they do not. We note that subjects seem to be reasonably consistent in rating the similarity of images, and they are even reasonably consistent across subjects.

3.10.2 Relative Ratings

We would like to analyze consistency for relative ratings as well. How should we proceed? Again we have inter-leaved trials where the subject sees the same set of faces and must choose which, from a set of 24 faces, is most similar to a target face. We have a total of 10 sets of images which we repeat 4 times as in the absolute experiments. For one set of images consider each of the 4 trials. The subject may pick 4 different images as being

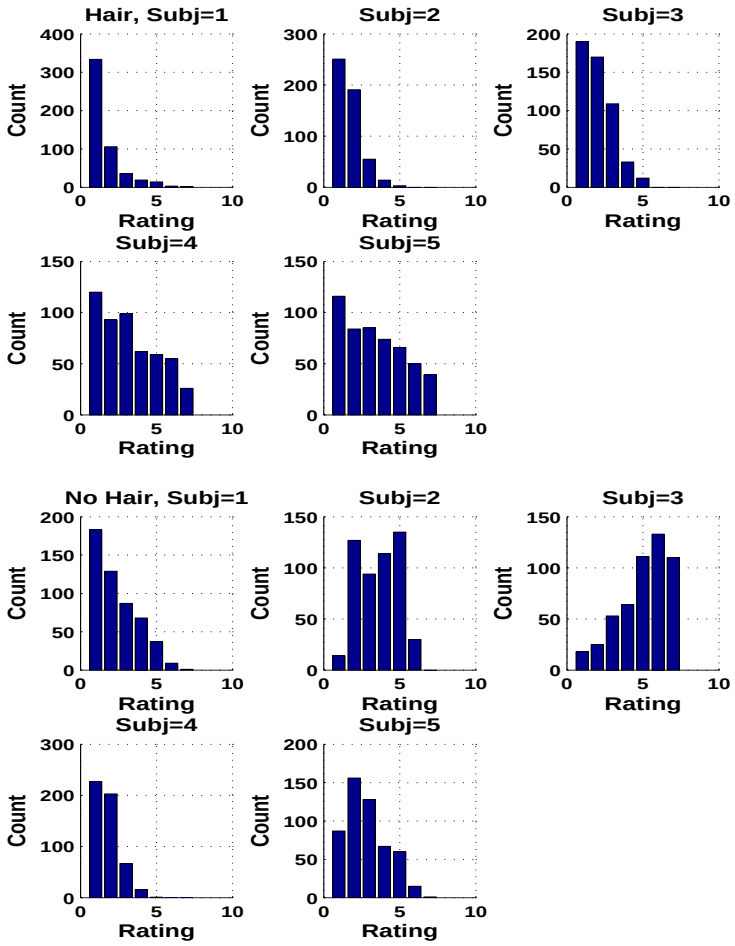

Figure 3.12: What was the distribution of ratings for the absolute rating task? (Top Set) Faces shown with hair. Each plot is a different subject. The x-axis is the particular rating chosen. The y-axis is the number of times this rating was chosen. Note the right-skew distribution. Subjects were not equally likely to chose any particular rating. Subjects were most willing to say faces were very dissimilar (rating 1). (Bottom Set) Same when images were shown without hair.

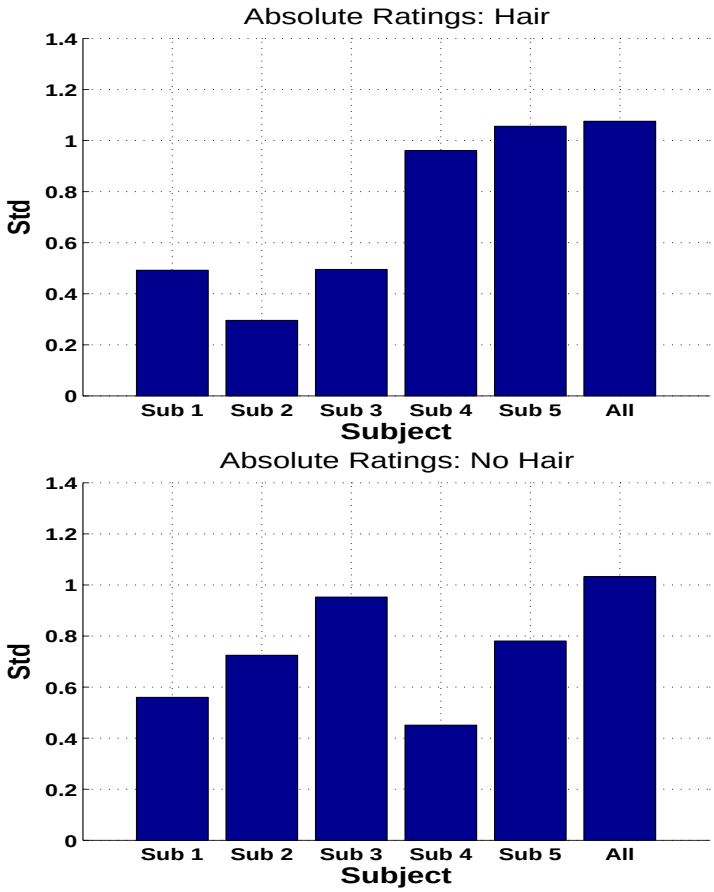

Figure 3.13: (Top) Results when subjects are shown images with hair. There were a total of 10 unique pairs of faces which were repeated 4 times. The order in which faces were shown to subjects was the same. The repeat pairs were interleaved among a total set of 514 trials. We measured the standard deviation (std) in the responses of each subject to the repeated pairs. A low std indicates that the subjects are consistent, they always chose the same number. We averaged the std across all pairs and plot the results as the first 5 bars. The last bar indicates the std across all subjects and gives an idea to inter-subject consistency. Here the mean score of each subject is subtracted from each score and the std is taken across all subjects. Although there does seem to be some inter-subject variability it does not appear to be extremely drastic. (Bottom) Same as top but when the subjects are shown images without hair. Note that there does not seem to be an appreciable decline in performance.

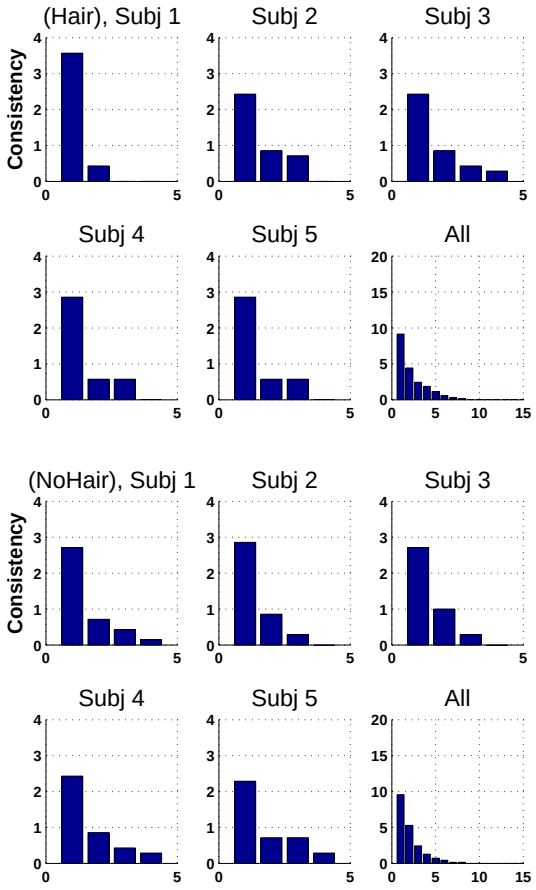

Figure 3.14: (Top Set) Results when subjects are shown images with hair for relative rating rating experiments. The consistency of the 5 subjects in the relative rating scheme. Perfect performance would be indicated by a single bar of height 4: the subject always picks the same image during the relative experiment. The final bar plot indicates how consistent subjects are between each other. In this case the maximum would be a 16 (4 subjects $\times$4 interleaved experiments. Again we see that subjects are reasonably consistent between themselves: different subjects tend to pick the same faces as being most similar to the target. (Bottom Set) Same as top but when the hair is not shown. The results between hair and no hair conditions seem consistent.

similar, or may consistently pick the same image as being similar. The latter case is, of course, preferable. To graphically illustrate performance, we consider the number of unique groups which each subject selects. In the first case there would be 4 groups, while in the latter there would be only a single group. We then look at the cardinality of these groups and sort them in decreasing order. Figure 3.14 shows results averaged over the 10 sets of repeated trials. Note that subjects seem to be reasonably consistent in their choice of the most similar face.

Part II

Active Learning

Chapter 4

Entropy-Based Active Learning

4.1 Abstract

Most methods for learning object categories require large sets of labeled training data. However, obtaining sufficient labeled training data can be a difficult and time-consuming endeavor. We have developed a novel, entropy-based "active learning" approach which makes significant progress against this problem. The main idea is to sequentially acquire labeled data by presenting an oracle (the user) with unlabeled images that will be particularly informative when labeled. Active learning chooses the order *in which the training examples are acquired, which, as shown by our experiments, can significantly reduce the overall* number *of training examples required to reach near-optimal performance. At first glance this may seem counter-intuitive: how can the algorithm know whether a group of unlabeled images will be informative, when, by definition, there is no label directly associated with any of the images? Our approach is based on choosing an image to label that maximizes the expected amount of information we gain about the set of unlabeled images. The technique is demonstrated in several contexts, including improving the efficiency of web image-search queries and open-world visual learning by an autonomous agent. Experiments on a large set of 140 visual object categories taken directly from text-based web image searches show that our technique can provide large improvements (up to 10x reduction in the number of training examples needed) over baseline techniques.*

4.2 Introduction

There are many situations in computer vision where the cost of obtaining labeled data is extremely high. Consider the problem of obtaining sufficient training data to build recog-

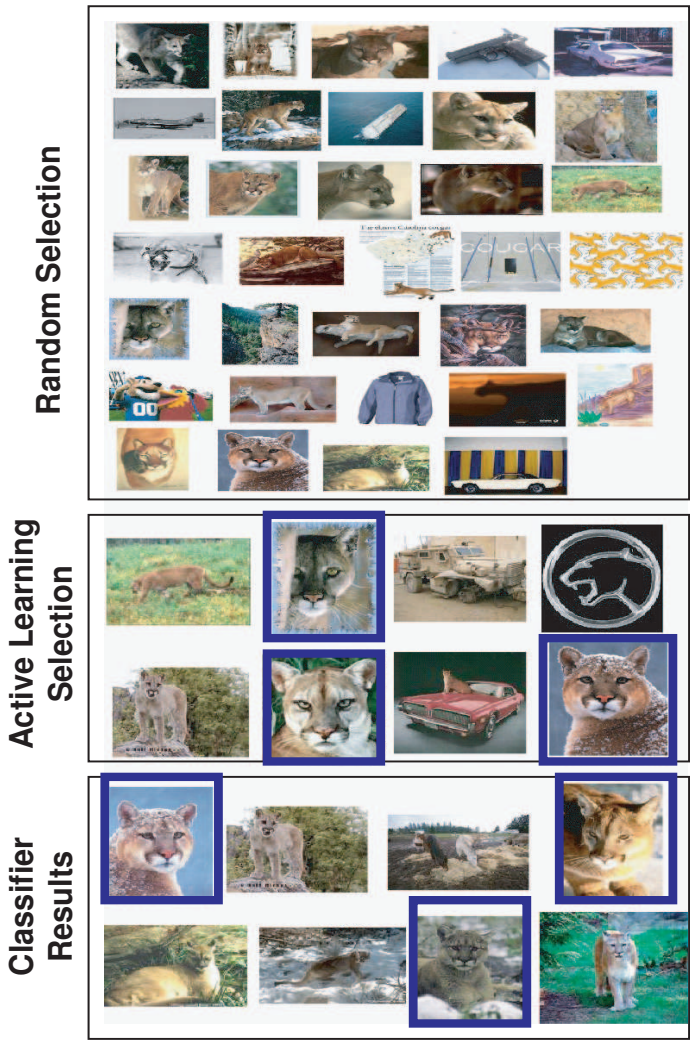

Figure 4.1: Web image search for category 'Cougar'. The user is allowed to label images to refine the image search query. (Top) Images the user needs to label in passive learning (randomly choosing images for the user to label) in order to achieve 82% of maximum performance. (Middle) Images sequentially selected by active learning for the user to label. The blue boxes indicate images the user has marked as 'Good'. Note that the user is required to rate over 4× fewer images when active learning is used compared to passive learning. In this example, the user prefers images that show the head of a cougar. (Bottom) The top 8 returns of the resulting classifier trained using the active learning images. Blue boxes indicate images which are 'Good' according to the user. Figure 4.2 shows similar performance gains for 137 image search categories.

nizers for thousands of image categories; clearly, one needs to be as efficient as possible when confronted with such a large number of categories and images. By intelligently choosing a subset of the images to label, we may be able to dramatically reduce the cost of obtaining an adequate labeled training set. As shown in Figure 4.2, similar issues arise when performing text-based searches for a particular object class. A basic search may return a high percentage of images that do not match the target concept. If we could refine these searches by acquiring user input, we could drastically increase the precision of the returned results. However, since user time is precious, it is critical that we attempt to squeeze as much information as possible from a minimum amount of feedback. Finally, consider an autonomous agent traversing a world and encountering new object classes. The agent is allowed to query an oracle (e.g. a human) regarding information found in the world, but, as the oracle's time is valuable, we would like to keep the number of queries to a minimum (e.g., consider a Mars rover, which must consume precious resources and time to query human 'oracles' back on Earth). Note that this portion of the thesis was done in collaboration with Michael C. Burl.

In this chapter we employ active learning to more quickly and efficiently learn visual object categories. In general active learning paradigms have 4 key components: (1) a set of labeled training examples, (2) a set of unlabeled examples for which labels can be obtained at some cost, (3) an oracle (e.g., a human) that can provide correct labels, and (4) a methodology for deciding which unlabled examples to request labels for given the current state of knowledge. Typically, this process occurs iteratively so that unlabled examples are selected and then labeled by the oracle. Given the new information from the oracle, additional unlabled examples are selected for labeling. Colloquially, we refer to the image selected for labeling at each interation as the "Most Informative Unlabeled Point" (the MIUP). This formulation of active learning is similar to that described in [MN98].

The main question in active learning is determining how to select the next image to label given what is currently known. Many heuristics have been developed; one of the most common is to choose examples which are the "most confused" with respect to the current classifier being used. For instance a confused point might be the point which lies closest to the decision surface separating two classes. Tong and Koller [TK00] develop this idea for Support Vector Machines (SVMs) by looking at the closest point to the current separating hyperplane. This idea has been further developed for image retrieval experiments [TC01].

Song et al. [SOS] take a different approach to selecting the most-confused point by generating numerous viable classifiers based on the known labels and choosing the point to label as the one that is most-confused by these classifiers. The authors of [FJ04] compare yet another "most confused" point approach and apply it to image retrieval experiments.

Relevance feedback, a related method, has been studied for content-based image retrieval (CBIR) systems since the mid-1990's. Many of these techniques focus on learning similarity measures between images or on weighting the importance of low-level features such as shape, color, and texture in defining the user's target concept. See [ZH03] for a review.

We take a different approach to active learning in the hopes of improving performance and improving the flexibility of the active learning approach. In fact, given the multitude of image classifiers available, we would like our active learning approach to be agnostic w.r.t. the underlying image classifier being used. We suggest to choose MIUP which results in acquiring the most information about the unlabeled images, or, expressed another way, which minimizes the expected uncertainty of the unlabeled set of images. Our algorithm, in the same spirit as [SOS], generates numerous viable classifiers in order to identify the MIUP and is well-defined for any underlying classifier being used (in this paper we demonstrate the technique on SVMs, Nearest Neighbor, and Kernel Nearest Neighbor classifiers).

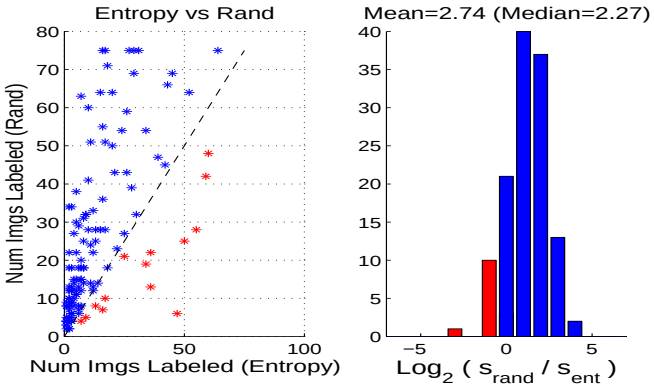

Figure 4.2: Comparison of Minimum Expected Entropy (MEE) active learning against passive learning (random sampling) over 137 categories of image search data (see an example category in Figure 4.2). (Left) Scatter plot showing the number of labeling rounds to reach 85% of asymptotic maximum performance for MEE (x-axis) versus random sampling (y-axis). Points above the diagonal indicate that MEE reaches near-optimal performance with fewer labeling rounds than passive learning. Each point represents a different object category and is the result of averaging over 50 experiments for the category. All experiments used an unlabeled pool of 250 images. (Right) Histogram of the $\log_2$ speedup (multiplicative improvement) of MEE versus passive learning. We indicate the mean and median increase in performance in title, i.e., a mean of 4, indicates that on average active learning reached target performance 4$\times$ faster than random learning.

In Section 4.3 we describe our algorithm. In Section 4.7 and the types of features we extract. In Section 4.4, we describe the data-sets used and show that our technique provides substantial speedup over competing methods on a large set of 137 image categories. We also consider the important question of deciding automatically when enough labeled data has been acquired. Finally in Section 4.5, we conclude and discuss implications of this work. Two technical appendices are included, which provide a brief overview of the Lazebnik Spatial Pyramid Match Kernel [LSP06] (used as the underlying classification method in our experiments) and several alternative active learning approaches that were tested.

4.3 Active Learning

We formalize our discussion of active learning as follows. Suppose we have a set of N images with each image belonging to one of L possible classes. Initially we assume that the class labels for all images are unknown. Active learning begins by choosing one or

more of the N images; these images are presented to an oracle that provides the correct class label(s). In subsequent rounds, the active learning algorithm considers both the the currently labeled images and the remaining unlabeled images and chooses additional images from the unlabeled set that would be particularly informative if their labels were known.

Let $\mathcal{U}^{(t)}$ be the pool of unlabeled images at the start of round t and let $\mathcal{L}^{(t)}$ be the corresponding pool of labeled images. Initially, we have $\mathcal{U}^{(0)}$ containing all N images and $\mathcal{L}^{(0)} = \emptyset$. For simplicity of notation, we will assume that one unknown image is to be chosen in each round and assigned a label, although see Section 4.3.4 for a discussion of the multi-return case.

To admit both deterministic and random algorithms, we suppose that an active learning algorithm outputs an $(M \times 1)$ vector $\mathbf{w}$ that specifies a probability distribution over the images in the unlabeled pool, where M is the number of unlabeled images available in the current round. A deterministic algorithm simply sets all the elements of $\mathbf{w}$ to zero, except for one element which is set to 1. (This element is then guaranteed to be picked.) *Random sampling* (equivalent to passive learning) sets $\mathbf{w}$ to $1/M \cdot \underline{\mathbf{1}}$, where $\underline{\mathbf{1}}$ is an $(M \times 1)$ vector of ones. Given $\mathbf{w}$, the oracle chooses an image according to this distribution and returns its label. This process leads to new labeled and unlabeled sets for the next round.

$$\mathcal{L}^{(t+1)} = \mathcal{L}^{(t)} \cup \{\mathbf{x}^{(t)}, y^{(t)}\} \tag{4.1}$$

$$\mathcal{U}^{(t+1)} = \mathcal{U}^{(t)} \setminus \mathbf{x}^{(t)} \tag{4.2}$$

where $\mathbf{x}^{(t)} \in \mathcal{U}^{(t)}$ is the example chosen in round t and $y^{(t)}$ is its label assigned by the oracle.

4.3.1 Minimum Expected Entropy

The usual goal with active learning is to learn, as quickly as possible, a decision function $g(\cdot)$ that accurately assigns class labels y to test images $\mathbf{x}$. However, given the uncertainties involved (even the form of the underlying class-conditional probability distributions is unknown), it is difficult to directly optimize this criterion. Instead, we have developed a novel active learning approach that attempts to sequentially minimize the expected entropy (uncertainty) of the labels for the unlabeled images given the current labeled set.

Let $\mathcal{H}(\cdot)$ represent the entropy of a set of images. In round t, we want to choose the image that produces the maximum reduction in entropy (equivalently, maximum gain in

information) once its label is known.

$$\mathbf{x}^{(t)} \;=\; \arg\max_{\mathbf{x}} \; \mathcal{H}\left(\mathcal{U}^{(t)}|\mathcal{L}^{(t)}\right) - \mathcal{H}\left(\mathcal{U}^{(t+1)}|\mathcal{L}^{(t+1)}\right) \tag{4.3}$$

Since only the second term depends[1] on $\mathbf{x}$), we can instead solve the following *minimization*:

$$\mathbf{x}^{(t)} \;=\; \arg\min_{\mathbf{x}} \; \mathcal{H}\left(\mathcal{U}^{(t+1)}|\mathcal{L}^{(t+1)}\right) \tag{4.4}$$

There is a problem with our formulation so far. In both Equations 4.3 and 4.4, we have an entropy conditional on $\mathcal{L}^{(t+1)}$, which presumes we know the label that the oracle will assign to $\mathbf{x}$. Since this label information is unknown before we consult the oracle, $\mathcal{H}\left(\mathcal{U}^{(t+1)}|\mathcal{L}^{(t+1)}\right)$ cannot be calculated. To resolve this issue, we instead compute an entropy conditional on each possible result the oracle might give for the label of $\mathbf{x}$. We then average these conditional entropies weighted by the probability that $\mathbf{x}$ takes on a particular label to generate an *expected entropy*:

$$\overline{\mathcal{H}}_{\mathbf{x}} \;=\; \sum_{j=1}^{L} P(Y=j|\mathcal{L}^{(t)}) \cdot \mathcal{H}\left(\mathcal{U}^{(t+1)}|\mathcal{L}^{(t)} \cup \{\mathbf{x}, j\}\right) \tag{4.5}$$

where Y is a random variable representing the label of $\mathbf{x}$. The Minimum Expected Entropy (MEE) algorithm chooses the image that results in the minimum value for $\overline{\mathcal{H}}_{\mathbf{x}}$.

$$\mathbf{x}^{(t)} \;=\; \arg\min_{\mathbf{x}} \; \overline{\mathcal{H}}_{\mathbf{x}} \qquad \text{(MEE)} \tag{4.6}$$

The main difficulty in implementing MEE is to estimate $\mathcal{H}\left(\mathcal{U}|\mathcal{L}\right)$. (The superscripts that indicate the epoch number have been dropped to simplify the notation.) This quantity is the *joint entropy* over the random variables Y_k representing the labels of the unlabeled images conditional on $\mathcal{L}$. The joint entropy, of course, depends on the full joint probability distribution over the vector of Y variables, which is difficult to estimate. Therefore, we make use of the subadditivity property of entropy:

$$\mathcal{H}\left(Y_1, Y_2, \ldots, Y_M|\mathcal{L}\right) \;\leq\; \sum_{k=1}^{M} \mathcal{H}\left(Y_k|\mathcal{L}\right) \tag{4.7}$$

to replace the joint entropy by a sum over the individual (marginal) entropies; this new

[1] The dependence is implicit; $\mathbf{x}$ is the new image from $\mathcal{U}^{(t)}$ to be labeled.

quantity serves as an upper bound for the joint entropy. (The bound is tight if the Y_k's are independent.)

To estimate $\mathcal{H}(Y_k|\mathcal{L})$, we simply need to know the probability distribution over the possible label values that Y_k can take, then the entropy is given by:

$$\mathcal{H}(Y_k|\mathcal{L}) = -\sum_{l=1}^{L} P(Y_k = l|\mathcal{L}) \cdot \log_2 P(Y_k = l|\mathcal{L}) \tag{4.8}$$

Estimation of the label probabilities is discussed in the next subsection. Algorithm 4.3.1 provides a pseudocode summarization of the Minimum Expected Entropy approach.

---

for each round t **do**
  for each unlabeled image $\mathbf{x}_i \in \mathcal{U}^{(t)}$ **do**
    for each possible class label $j \in \{1, \ldots, L\}$ **do**
      Estimate $P\left(Y_i = j|\mathcal{L}^{(t)}\right)$
      for each unlabeled image $\mathbf{x}_k \in \left(\mathcal{U}^{(t)} \backslash \mathbf{x}_i\right)$ **do**
        for each possible class label $l \in \{1, \ldots, L\}$ **do**
          Estimate $P\left(Y_k = l|\mathcal{L}^{(t)} \cup \{\mathbf{x}_i, j\}\right)$
        end for
      end for
      Calculate conditional entropy $\mathcal{H}_j$
    end for
    Combine conditional entropies $\mathcal{H}_j$ for $j = 1, \ldots, L$ into an expected entropy $\overline{\mathcal{H}}_{\mathbf{x}_i}$
  end for
  Set $\mathbf{w}$ to δ_{i_*} where $\mathbf{x}_{i_*}$ yields lowest expected entropy.
end for

4.3.2 Look-Ahead Estimate of Class Probabilities

Here we consider how to estimate class probabilities for the unlabeled images given a set of labeled images $\mathcal{L}$ when we have classifiers, such as kernel nearest neighbor or SVM, that only return hard class decisions[2]. The key idea is to use a one step look-ahead scheme to construct a committee of classifiers. The predictions of the committee are then used to derive the desired label probabilities. The look-ahead step considers each of the M currently unlabeled images in $\mathcal{U}$ and each possible value for its class label. Let $\{\mathbf{x}_m, n\}$ be a look-ahead image and its hypthesized label. Next we construct a classifier from $\mathcal{L} \cup \{\mathbf{x}_m, n\}$. Repeating this process for each unlabeled image and each possible value for its label yields $M \cdot L$ classifiers, which we apply to each of the images in $\mathcal{U}$.

[2]Although the SVM hyperplane distance can be used to construct pseudo-probabilities as in [Pla00], this approach cannot be applied to other types of classifiers.

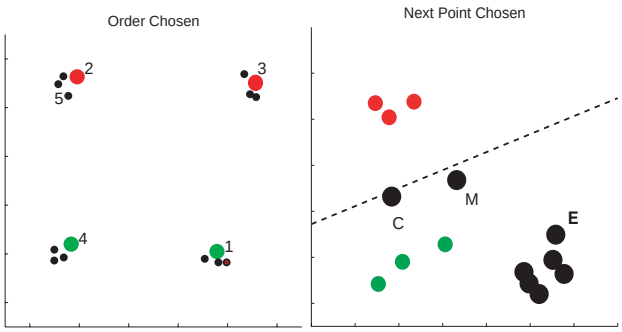

Figure 4.3: (Left) Illustration of minimum expected entropy (MEE) active learning for a set of $N = 20$ points from $L = 2$ classes. The numbers show the order in which MEE presents points to the oracle with the colored circles showing the resulting label. With its first four queries, MEE visits each of the "clusters" providing for the quickest reduction in the uncertainty of the labels of the other points. This experiment was run using a Nearest Neighbor classifier. (Right) Comparison of different active learning approaches. The red and green points are currently labeled, while the black points are unlabeled. The dotted line shows the SVM hyperplane found with the current set of labeled points. The next query to the oracle is shown for three different active learning approaches: (E) MEE, (C) closest to the current hyperplane as in [TK00], and (M) most confused point based on lookahead estimates for the label probabilities. Observe that MEE chooses a point that sits next to many other unlabeled points *and* is in a relatively unexplored region of space. Note that our MEE framework allows us to use *any* underlying classifier: Nearest Neighbor classifiers are used in the left figure while SVMs are used in the right.

The classifier results can be collected into a block-structured matrix $\mathbf{B}$ consisting of L blocks by L blocks with each block being an $(M \times M)$ matrix. (As usual, L is the number of class labels and M is the number of currently unlabeled images.) The (l, n) block contains an indicator matrix of 0's and 1's. The (k, m) position within a block is a 1 if the classifier trained with $\mathcal{L} \cup \{\mathbf{x}_m, n\}$ says that example $\mathbf{x}_k$ belongs in class l. We can then use B to write:

$$\underline{\mathbf{P}}(Y_k = l) \quad = \quad \frac{1}{M} \cdot \mathbf{B} \cdot \underline{\mathbf{P}}(Y_m = n) \tag{4.9}$$

(The notation is such that the probability vectors $\underline{\mathbf{P}}(\cdot)$ on the LHS and RHS are stacked up in "label-major" order.) The RHS probability vector acts like a prior probability on the labels of points. The LHS is analogous to a posterior, re-estimated after we see the predictions of the committee of look-ahead classifiers. This equation can be understood from either a histogram viewpoint or an expectation viewpoint. From the histogram viewpoint, we are accumulating the probability that a classifier selected randomly from the committee says "1" to the event $(Y_k = l)$. From the expectation viewpoint, we are computing over the entire committee an expectation for the binary-valued classifier confidence in the event $(Y_k = l)$. (This duality exists because the expectation of a binary-valued random variable $E[X]$ is the same as $P(X = 1)$.)

From Equation 4.9, we can obtain an estimate for the class probabilities by taking $P(Y_m = n) = 1/L \cdot \underline{\mathbf{1}}$ on the RHS and multiplying by $1/M \cdot \mathbf{B}$. In principle, this process can be iterated to refine the probabilities. In the limit, the probabilites satisfy the solution of a fixed point problem $\mathbf{p} = 1/M \cdot \mathbf{B}\mathbf{p}$. Solving this equation amounts to finding the eigenvector of $\mathbf{B}/M$ corresponding to eigenvalue 1. (Since the $(1 \times M \cdot L)$ vector $\underline{\mathbf{1}}^T$ is a left eigenvector of $\mathbf{B}/M$ with (left) eigenvalue 1, we know that there exists a right eigenvector of $\mathbf{B}/M$ with eigenvalue 1.) Although the iterative and fixed point approaches are elegant, in practice we have found from a limited set of experiments that a single iteration of Equation 4.9 with uniform probabilities on the RHS yields better results. We are still studying this issue, but believe the main cause is that Equation 4.9 does not adequately reflect the possibility that look-ahead classifiers trained from a finite amount of data are simply wrong. Iterating causes the estimation procedure to develop unwarranted certainty about the class labels.

4.3.3 Computational Cost

The minimum expected entropy algorithm described above, although intuitively appealing, is somewhat expensive computationally. In particular, the algorithm is $O(L^3N^3)$ where N is the number of unlabeled images and L is the number of classes. A typical run on 2Ghz Pentium using $N = 250$ and $L = 2$ and a combination of Matlab and C code takes about 30 minutes. (Due to the cubic dependence on N, using fewer images takes far less time.)

It is interesting to consider how the benfits of active learning change with the size of the pool of unlabeled images. Figure 4.4 addresses this point and shows that increasing the pool tends to constantly increase the performance of Entropy-based active learning over random sampling.

Given the benefit of increasing pool size, it is useful to consider methods of reducing the computational cost. One possibility is to consider only a fixed number of images M when calculating the expected entropy for a particular images, such that we reduce the $O(L^3N^3)$ to $O(L^3N\tilde{N}^2)$ for some constant $\tilde{N} < N$. In particular, we randomly sample a set M unlabeled images from the entire pool of unlabeled images to compute the entropy with. Figure 4.5 illustrates the effects of this sub-sampling of the unlabeled pool and shows that performance tends to drop off fast when subsets are used. There are many other possible methods for increasing speed and we leave these open as topics for further research. However we note that we were able to easily run experiments using 250 unlabeled images and a non-optimized code.

4.3.4 Multiple Return Active Learning

So far, we have viewed active learning as presenting a single image (or a probability distribution $\mathbf{w}$ for selecting a single image) to the oracle for each round of labeling. In practice, however, it will often be preferrable to return a *set* of informative images rather than the single most informative image. Consider the web image search application. Here, the interaction with the user would be cumbersome if they were asked to label only a single image at a time; instead, it would be better to ask the user to label a set of images at once. There are also technical reasons for returning multiple images at once as illustrated in Figure 4.6. Analogous to greedy forward feature selection algorithms, single-return active learning picks three images that do not cover the space as well as if the three images were picked at once

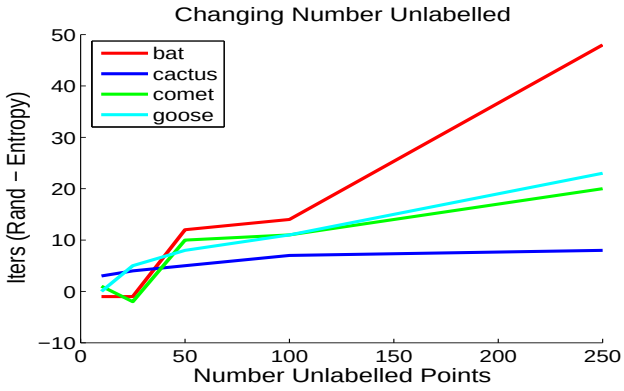

Figure 4.4: Results from four categories showing how the difference in time (iterations) required for random and MEE to reach 85% of maximum performance varies as the size of the unlabeled pool is increased. The relative advantage of active learning is clearly more pronounced when the unlabeled pool is larger.

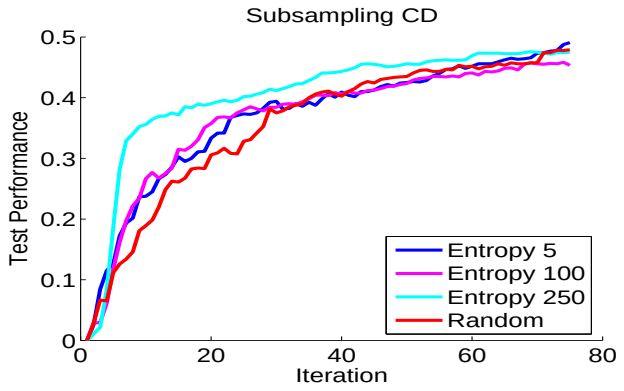

Figure 4.5: Here we consider the effects of only using a subset of the available images to compute the entropy. X-axis: the number of images labeled. Y-axis: performance on a separate test set of data. There are a total of 250 unlabeled images available, and each line represents using a subset of those points. Red represents randomly choosing images. Note that performance falls off quickly as less images are used to compute entropy, i.e. best performance results from using all 250 images in the unlabeled pool. Results shown are using the category 'CD', but are typical of other categories as well.

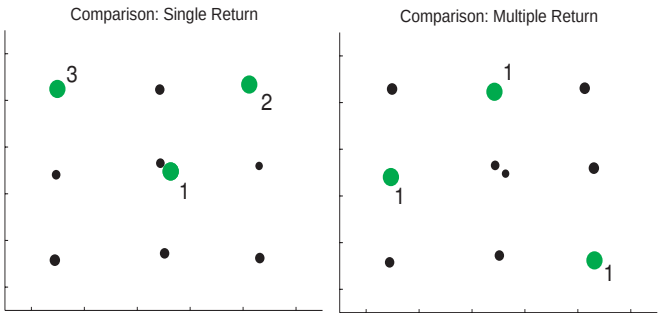

Figure 4.6: (Left) Single-return active learning applied three times. (Right) Multi-return active learning applied once with subsets of size 3. Clearly, the multi-return approach is able to generate a more optimal covering of the space.

as a unit.

Multi-return active learning using the minimum expected entropy principle requires only a minor modification to Algorithm 4.3.1. In particular, the loop over unlabeled images ($\mathbf{x}_i$) is replaced by a loop over subsets of unlabeled images of size s. For each subset, we consider the L^s possible assignments of labels to the elements in the subset and compute an expected entropy as before. The subset that results in the lowest expected entropy is then presented to the oracle for labeling. Exhaustively considering all subsets of size s in each active learning round is clearly only feasible for small s; howver, given that the information value of a proposed subset can be easily evaluated (using the expected entropy), other heuristics can be incorporated to focus consideration onto a smaller number of promising subsets.

4.4 Experiments

There are three key sets of experiments we performed. The first is the UCI Machine Learning data-sets. The second is inspired by Figure 4.2 and involves increasing the precision of web image searches. This is essentially a two-category task, discriminating images that match the target concept from those that do not. The last set of experiments considers many (up to 10) categories and is inspired by an autonomous agent exploring a world.

4.4.1 UCI Machine Learning Data Sets

We chose to experiment first on standard machine learning data-sets, namely the UCI Machine Learning Data-sets. We directly compare entropy-based learning to learning using random sampling.

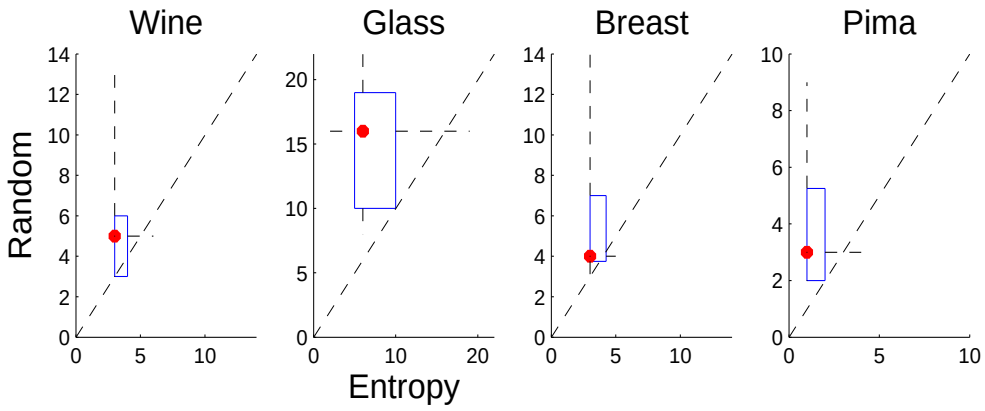

Figure 4.7: Experiments on UCI data-sets. Each column is a box-plot for a different UCI set. Performance is measured by the number of iterations required to reach 85% of maximal performance. Maximal performance was computed by training a classifier on all training samples. NN classifier used. We randomly sampled 120, 75, 75, 50 training examples for the Wine, Glass, Pima, and Breast classes respectively for each experiment. An unseen set of 50 test samples was used. Box-plots over 20 experiments. Performance was measured by taking the average performance across the main diagonal of the confusion table for the testing data. X-axis, number iterations for entropy-based sampling, Y-axis number iterations for random sampling. Values above the main diagonal indicate that entropy-based active-learning reached 85% performance before random learning. Box-plots show 25 and 75 quartiles and indicate the total spread of the data. The red dot indicates the median number of iterations to reach 85% performance. Active learning tends to reach higher performance more quickly than random sampling for these data-sets. Active learning also tends to have a tighter variance indicating more consistency across experiments.

# Categories	# Images	Good	Bad
137	30277	93 (82/134)	178 (56/402)

Table 4.1: Table detailing the large collection of images obtained from web image searches. The total number of categories, the total number of images, the mean and min / max number of images in each of the categories labeled as 'Good' and the same for 'Bad'.

In the experiments, we randomly divide a set of UCI data into a training and test set. The algorithm selects points for labelling from the training data. At each iteration the algorithm selects a single point for labelling from the set U of unlabelled training data. This point is added to the set of training points used to train the classifier. The random algorithm selects a random point from U for labelling. The test set is used to evaluate the performance of the classifier at each iteration. Figure 4.7 contains box-plots for 4 different UCI sets. Note that our active learning algorithm performs very favorably when compared to random sampling.

4.4.2 Web Image Searching

Consider again Figure 4.2 in which the user typed 'Cougar' into an image search engine. The idea is that the user must label a set of images in order to refine the search as most of the returned images do not contain the category of interest. In this case active learning can provide a drastic increases in speed by choosing the MIUP. In this section we explore experiments designed to mimic just such situations.

4.4.2.1 Image Search Data Set

Our goal in this set of experiments was to mimic as closely as possible a real image search on the web. We collected images returned from actual text-based web image searches with Google and PicSearch. In order to obtain comprehensive statistics we collected images using 137 keywords. We next asked 3 sorters to label the images as one of three classes: 'Good', 'Ok', 'Bad'. The 'Good' images contain images of the class of interest while the 'Bad' images do not contain the object of interest [3]. We removed all duplicate images using software which first extracts features using the Lowe Difference of Gaussian detector and SIFT descriptors [Low04] and then compares these sets of features across all images in the category. If there are more than 100 good matches between two images, the images are considered to be identical and one is removed. For our experiments we only used images from the 'Good' and 'Bad' categories. This is the largest data-set of its type to our knowledge. Table 4.1 gives some statistics on the data-set we collected. The full set of category names are too numerous to list here.

4.4.2.2 Results

Our experiments were conducted as follows. For each category we combined the 'Good' and 'Bad' images into a single large pool. From this pool we randomly selected a set of 75 testing images. The rest of the images were used as the pool of unlabeled data for active learning. We then followed Algorithm 4.3.1 and iteratively chose images to label using MEE active learning. We also considered alternative approaches including: (1) random sampling (passive learning) choosing a image, (2) choosing the most confused image, and (2) choosing the unlabeled image with highest kernel density (see Appendix 1 for an overview of these alternative-methods. In all cases, kernel nearest neighbor using the Spatial Pyramid Match

[3] We will make both the positive images and negative images publicly available pending acceptance.

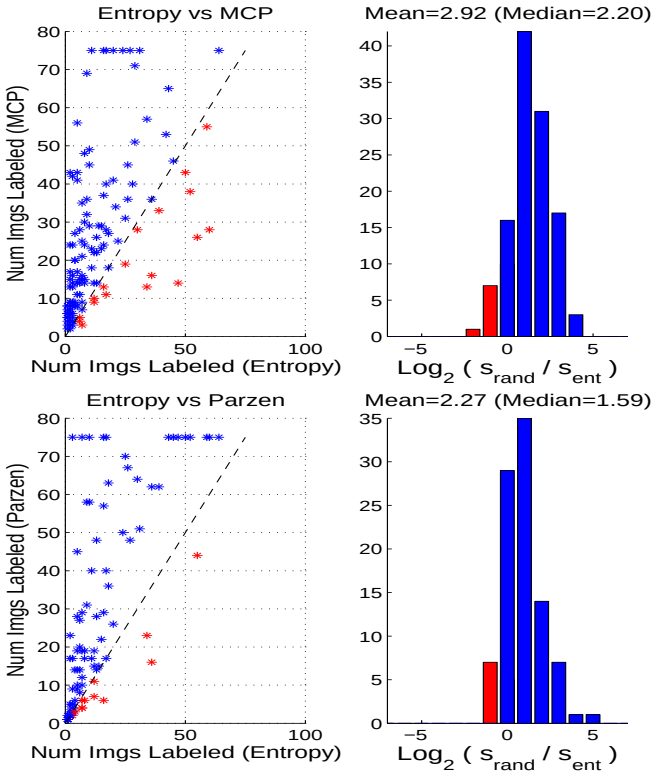

Figure 4.8: Comparison of MEE active learning with: (top) choosing the Most Confused Point and (bottom) the Maximum Unlabeled Density (sampling where there is the highest density of unlabeled points.

Kernel of Lazebnik [LSP06] was used as the classifer in our experiments (see Appendix 2 for an overview of [LSP06] for full details).

How do we quantify performance? In these experiments we are interested in the precision for the top 25 closest images. In other words what percentage of the 25 closest returned images are in the 'Good' class? Let p_{max} be the maximum possible performance on the test set (this occurs when all images in the initial unlabeled pool get labeled). Now consider the number of images, $s_i, i \in (0, 1, 2, 3)$ which need to be labeled to achieve 85% of p_{max} where i indexes over the various active learning methods. Results showing the performance of MEE active learning versus the alternative methods are presented in Figures 4.2 and 4.8. Note that MEE significantly outperforms all of these competing methods. In fact we reach 85% of p_{max} up to 10× faster using MEE to pick the MIUP when compared to random and perform on average close to 3× better than random on these data-sets.

4.4.3 Exploration Agent

The next set of experiments looks at multiple classes. We motivate this experiment by considering an agent travelling through a real or virtual world (for instance a mobile robot exploring the environment or a web-crawler). This agent will be confronted with a slew of visual information. With minimal supervision can the agent discover and learn to recognize multiple categories of objects? Given that there is a considerable cost associated with obtaining a label for any particular image (e.g., the agent must ask a human observer whose time is precious), for which images should the agent request labels?

4.4.3.1 Open-World Learning Experiments

In these experiments, the unlabeled pool contains examples from numerous object categories. Initially our agent has no knowledge of the world and assumes there is only a single object class. The agent chooses informative images via active learning and asks an oracle to label these images; the oracle returns the true label of the unknown image. As new classes are encountered the agent updates its knowledge of the number of classes which exist in its world (i.e., the number of classes L) increases.

Here, we use a slightly different criteria from the Image Search experiments to assess performance. Consider that in this scenario the agent is seeking to build the best classifiers for visual categorization, and thus we consider the classification performance on a separate set of test data.

Our experiments were conducted as follows. First we selected the Good examples from L different categories. From these we randomly choose a set of N images to form $\mathcal{U}$, the pool of unlabeled examples. The rest of the images are used as a test set with which to evaluate the performance of our algorithm.

4.4.3.2 When Have Enough Images Been Labeled?

A natural question which arises is: when has the agent learned enough about the environment? Or, phrased another way, when should the agent stop querying the oracle? Our MEE framework allows us to estimate the entropy $\mathcal{H}^{(t)}$ after each active learning iteration and hence the amount of information gained after each active learning iteration can be approximated by: $\mathcal{I}^{(t)} = \mathcal{H}^{(t)} - \mathcal{H}^{(t-1)}$. In Figure 4.4.3.2 we consider the relationship between $\mathcal{I}^{(t)}$ and the performance gains on the test set. In particular we note a strong relationship

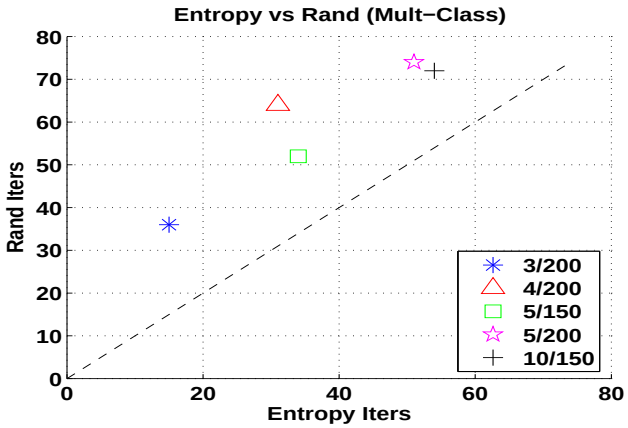

Figure 4.9: Similar to Figure 4.2 but showing multi-class experiments. Each point represents an experiment indicated by the legend. The legend indicates two numbers: the first is the number of classes used and the second is the size of the pool of unlabeled data. Each point is the average over 25 iterations, where each iteration involves choosing a random set of categories and training data for each category.

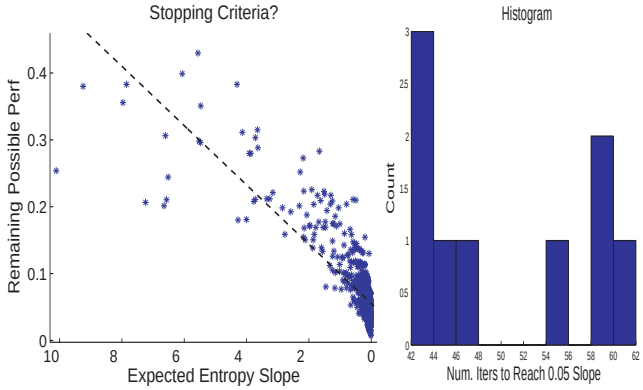

Figure 4.10: Can we use MEE to determine when to stop learning? (Left) Scatter plot. X-axis: the slope of the MEE at a particular iteration t. Y-axis: the remaining performance gain possible at the same iteration t (the difference between the current performance and the maximum possible performance). Dotted black line is a regression over all the points. A steep entropy slope correlates with large potential increases in performance, indicating we should keep learning. A shallow entropy slope (near zero) correlates with little potential for performance increase indicating we should stop sampling images. (Right) Histogram for different 5 class 200 unlabeled image experiments. The x-axis is the time taken to reach a particular slope value less than .05. Since it takes different experiments longer to reach a shallow slope, and from the left figure we see that a shallow slope indicates very little potential for performance increase, we can label substantially fewer images for some experiments using MEE as a stopping criteria.

between the change in information and the change in performance of the system. This allows us to utilize MEE to estimate when we have acquired sufficient information about the unlabelled images.

4.4.4 Performance Analysis

In this section we examine the statistics of the categories when entropy-based learning does well or poorly relative to random sampling. We analyze the data post-hoc, i.e. can we understand why the algorithm performed better or worse than random sampling based on the statistics of the data? Figure 4.4.4 analyzes some properties which seem to correlate with how well active learning will do on a particular set of images. We notice that active learning tends to outperform random sampling when the size of the proportion of good examples decreases. This is due, in part, to the use of our performance metric which measures the precision of the top recalled results. If more 'good' images are labelled the precision should increase. The other observation is that when the 'Good' images are more clustered relative to the 'Bad' images, we notice that active learning outperforms random sampling more

significantly. It seems that active learning shines when there are tight clusters within the good data-set. These results follow some of the results shown in Figure 4.3 which show active learning sampling from clusters sequentially.

4.5 Discussion

We have developed a novel "active learning" algorithm that enables hundreds of complex object categories to be recognized with a minimal amount of labeled training data. Our approach uses a principled, information-theoretic criteria to select the most informative images to be labeled. The technique is well-defined for any underlying classifier (kernel nearest neighbor, SVM, etc.), extends naturally to multi-class and multi-return settings, and can automatically determine when enough labeled training data has been acquired to insure near-maximal recognition performance. Against passive learning and a variety of alternative active learning approaches, our method consistently achieves near-maximal performance with one-half to one-third the number of training and in some cases the improvement is 10x or more.

There are other potential methods for active learning which we have not explored in this thesis. For instance, in this work we take the point of view of minimizing the expected entropy of the unlabelled points. We could instead use alternate metrics for picking which point to label next including the point which results in the largest change in labels if it were labelled one way or another. This is, for a Nearest Neighbor classifier, very similar to the Parzen window approach suggested above, which, in comparison to active learning, does very poorly. Another point of view would be to chose the point to label which, if labelled one way or another, results in the largest change in the parameters of the model. Since we did not analyze parametric methods much in this thesis (i.e. SVMs, Neural Networks, etc), but instead concentrated on non-parametric methods (i.e. Nearest Neighbor), we did not explore active learning methods of this sort. Related to this, using an SVM, it might be possible to look at the change in the number of Support Vectors (SVs) as an indication of how stable a classifier is. This information could be used in addition to the classification performance to calculated the probability of a classifier existing. For instance a classifier which results in a large number of new SVs might be less probably than one which results in no change or a decrease in the number of SVs.

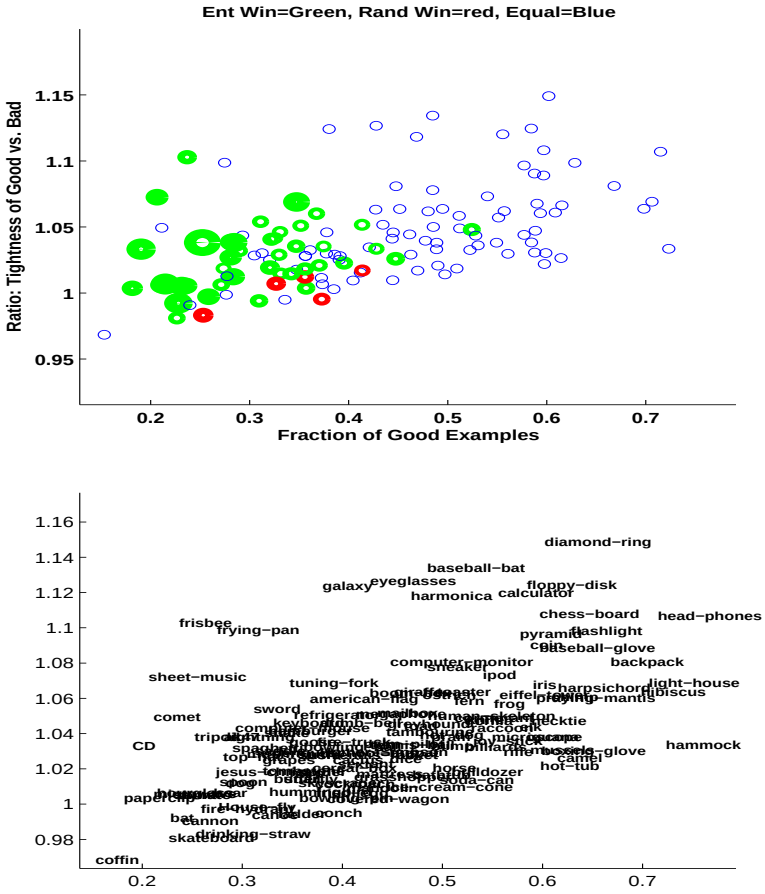

Figure 4.11: Analysis of the statistics of the categories active learning performed well on and which it performed poorly on relative to random sampling. (Top) The color of the dot indicates whether active learning was beneficial or not. Green dots indicate active learning was useful, red dots indicate that random sampling did better and blue dots indicate that the performance was about the same for the two methods. The size of the dot indicates how much better active learning (green dots) or random (red dots) did compared to the other method. I.e. a large green dot indicates that active learning is doing much better than random learning. The x-axis shows the proportion of good examples, i.e. farther to the left indicates that there are fewer good examples: a value of .2 indicates that 20% of the images were labelled as good. The y-axis measures how tight the cluster of good images is relative to the cluster of bad images. For each image we look at the average distance between the closest 3 images in the good cluster to the closest 3 images in the bad cluster. We take the average ratio of these two numbers over all the images. A higher number indicates that the good images are moor clustered than the bad images. What do we observe in this plot? We notice that as the fraction of good images increases, the relative increase of active learning over random sampling also increases (more green dots are on the left). We also notice that as the tightness of the good clusters increases relative to the bad cluster, the performance of active learning also increases. (Bottom) The category labels for the points shown above.

4.6 Appendix 1: Alternative Active Learning Methods

It is useful to compare the MEE approach to other active learning approaches. We chose two methods to compare against: (1) the Most-Confused-Point (MCP) and (2) Maximum Unlabeled Density (MUD). MCP follows the spirit of [TK00] by choosing the image which is most confused between the different classes. To calculate the image that is most confused, we follow the paradigm of Section 4.3.2 to estimate class probabilities for each image. The image which is most confused based on these probability estimates is then selected.

The MUD technique estimates a probabilty density $p(\mathbf{x}|\mathcal{U})$ over the unlabeled points using the Parzen Window kernel density estimation technique. The unlabeled point with the maximum associated probability density is selected for labeling, i.e.,

$$\mathbf{x}^{(t)} \quad = \quad \arg\max_j \sum_i \frac{1}{N} K(\mathbf{x}_j, \mathbf{x}_i) \tag{4.10}$$

where i and j are indices for unlabeled images. The MUD technique is comparable to clustering the unlabeled data and choosing a point near the center of the most prominent cluster. The drawback of this idea is that it does not distinguish between a high density of unlabeled points and a high density of unlabeled points whose labels are uncertain. Both of these alternative approaches are compared to MEE in Figure 4.8.

4.7 Appendix 2: Pyramid Match Kernel

The Spatial Pyramid Matching algorithm of Lazebnik et al. [LSP06] as it currently yields strong performance on similar object categories to those used in this paper (see [GHP07]) and is fast. For each image, we extract a set of SIFT features [Low04]. 10,000 features are chosen at random from a training set of images in order to form a vocabulary of $M = 200$ words (clusters), and the vocabulary is used to map each subsequent feature to one of the 200 words. Next the image is split into a 4×4 grid and the number of times each of the 200 features is found in each of the 16 bins is counted. The matching kernel is computed using the above set of $4 \times 4 \times M$ histograms. In this context *matching* means finding the number of common elements in any two bins. If the counts in two bins are n_1 and n_2 the match is $\min(n_1, n_2)$. Matching is computed using both spatial information and the appearance.

Part III

Comparing Generative and Discriminative Learning

Chapter 5

Conditional Likelihood

5.1 Abstract

Here we explore the relative merits of generative and discriminative learning techniques for object recognition. Visual recognition algorithms learn models from a set of training examples. Generative learning, where each model is trained to represent the data of the corresponding category, is popular both because it creates explicit models of the object classes and because the algorithms are relatively easier to optimize. However, generative models are prone to confusion when confronted with images from similar classes. We study a discriminative approach which maintains the power of generative learning by creating explicit class models, while simultaneously focusing on features unique to each class. We conclude by proposing a multi-class object recognition system which initially trains object classes in a generative manner, then identifies subsets of similar classes with high confusion, and finally trains models for these subsets in a discriminative manner to realize classification performance gains.

5.2 Introduction

Humans can easily recognize and distinguish thousands of visual categories. The best computer algorithms achieve only a fraction of human performance in terms of both the number of classes recognized and the accuracy in distinguishing between those classes. The impressive performance of the human system can be ascribed in part to our ability to recognize the overall appearance of objects, as well as detect subtle differences between very similar object class categories, such as the difference between male faces and female faces or bicycles and motorcycles.

Figure 5.1: (Left) Examples from two very different categories, motorcycle and airplane, and two very similar categories Bush and Kerry. Right, discrimination performance using a generative model for Motorcycle vs. Airplane (left) and Bush vs. Kerry (right). Airplanes and motorcycles are much easier to discriminate using a generative approach. Chance performance indicated by the dotted line.

Object recognition algorithms can be roughly grouped into two separate learning paradigms: generative [Web00, FPZ03, Low04, SK00] and discriminative [VJ01, KH03, OFPA, TMF04]. Generative object recognition algorithms create object class models using only the data of the class to be modelled. These are well suited for modelling large numbers of object categories [LFP06] as they easily allow for the introduction of new object classes. However, by not taking into account the statistics of similar classes, these models can perform poorly when presented with either a large number of object categories, or several similar object categories. Discriminative object recognition techniques generally utilize the data from all object classes to create an explicit decision boundary separating the classes of interest. Such techniques tend to outperform their generative counterparts but suffer from both increased computational complexity and the lack of explicit object class models. Our goal is to take advantage of both generative and discriminative learning methods in order to create explicit models of object class categories while maintaining discriminative power.

In this paper we use a principled probabilistic approach to extend the generative 'Constellation Model' [Web00, FPZ03] framework for creating object class models in a discriminative setting. Utilizing a generative framework in conjunctive with a discriminative cost function has been previously proposed by other authors [Jeb01, Bou03, RH97] and is generally referred to as maximizing the *Conditional Likelihood* (CL). These studies do not observe substantial gains in using CL over generative approaches on traditional learning systems data-sets[1]. One of the objectives of this study was to assess the performance of discriminative methods in visual object recognition.

We conclude by proposing a general object recognition system which utilizes the relative merits of both generative and discriminative learning. In this system, models are initially trained in a generative fashion. Then, subsets of classes with high confusion are identified, indicating that these classes are similar. The subsets with high confusion are then trained in a discriminative manner.

This paper is organized as follows, first we will review the generative approach to the Constellation Model. We will then outline our discriminative learning approach. Finally, we will show examples of both generative and discriminative models and illustrate the system which utilizes both techniques.

[1] Among others, the UCI data-sets.

5.3 Review of the Constellation Model

Our approach to object class modelling builds on earlier work by Weber et al. [Web00] and more recently that of Fergus et. al. [FPZ03]. In this framework, an object is modelled by a 'constellation' of several parts, with each model containing information on both the appearance and relative position of each part. In these experiments we use a simplified version of these ideas which utilizes three independent parts. We use gaussian probability densities to represent the variations in appearance and shape of these components. The parameters for our models are learned by extracted interesting features from a set of training images and using these features to maximize a model representation.

5.3.1 Feature Detection and Representation

Interesting locations, known as *features* or *interest points*, must be identified within all images. We accomplish feature detection using either supervised or semi-supervised learning. For supervised learning, we manually select registered regions of images for learning. In semi-supervised learning, the feature detection process is accomplished using the Kadir and Brady [KB01] detector[2].

For supervised learning, we manually register 9 features for use by our learning algorithm: left eye, right eye, left hairline, right hairline, center hairline, nose tip, left mouth, right mouth, chin (9 features). When features were missing or occluded we chose the closest position in the image as our feature. A constant scale was assumed for all supervised features across all images.

We automate the feature selection process by using a feature detector in semi-supervised learning. The Kadir and Brady feature detector returns the positions, scales, and relative saliency of interest points within an image. The detector is well tuned for detecting circular regions within images, including eyes and wheels. The interest points with the highest saliency are used as features for learning. Figure 5.2 shows examples of detected features.

In order to construct an appearance representation for the salient points, we extract 11×11 pixel patches centered on the Kadir and Brady features. For semi-supervised learning we first scale the size of the patch extracted to correspond to the scale of detection

[2]Note the departure from the terminology of [FPZ03] and [LFP06] who consider their algorithms 'unsupervised'. This distinction becomes particularly important with the advent of purely unsupervised object learning algorithms, i.e. [FPZ04]

Figure 5.2: (Left) Examples of features selected manually for Schwarzenegger. Nine total features are selected. (Center) The same image but with features found using the Kadir and Brady detector. The 15 most salient features are shown.

and then sub-sample the patches to 11×11 matrices. We reduce the number of appearance parameters to be optimized by performing PCA on all patches from every image. We select the first K principal components for our appearance models, where K is typically 10. We use a matrix, A_i^c, of size $F \times K$ to represent the appearances of all features within image i of class c, where F is the number of features used. F typically ranges from 25-30 for semi-supervised learning.

5.3.2 Shape Representation

We construct a shape model to represent the variations in position for each model part. We record the positions of all interest points within an image i for class c in the variable X_i^c. We attempt to find the optimal mean and variance of gaussian densities for the shape model. The positions of our all model parts are relative to the first part. By conditioning on the first model component, the shape model becomes invariant to translations.

5.3.3 Generative Model

Here we present a generative framework which obtains maximum likelihood estimates for parameter values of each object class. Our goal is to find a set of model parameters θ_c where c is a particular class of objects, which optimizes the appearance and relative positions of the patches extracted from images in that class. θ_c represents both the means and diagonal variance components. Consider a set of object classes ranging from $1..C$ and indexed by c along with the images belonging to each class, ranging from $1..N_c$ and indexed by i. We have extracted both appearance, A_i^c, and shape, X_i^c, information from each image I_i^c. We assume that the shape and appearance models are independent of one another and that the images are I.I.D. The log likelihood of the training images given a particular parameter set

θ is:

$$\sum_{i \in c} log(p(I_i^c)) \quad = \quad \sum_{i \in c} log(p(A_i^c|\theta_c) \cdot (X_i^c|\theta_c)) \tag{5.1}$$

The maximum likelihood estimate will find the value of θ_c which optimizes the expression above. Given a particular image I_c^i, we have obtained a set of interest points as described above. We must assign an interest point to a particular model component. Since we do not a priori know which interest point belongs to which model component, we introduce a hypothesis variable h which maps interest points to model parts. We order the interest points in ascending order of x-position. This results in a total of $\binom{F}{M}$ unique combinations of interest points to parts and each hypothesis h will assign a unique interest point to each model part[3] We marginalize over the hypothesis variable to obtain the following expression for the log likelihood for a particular class:

$$= \sum_{i \in C} log(\sum_{h} p(I_i^c, h|\theta_c)) \tag{5.2}$$

$$= \sum_{i \in C} log(\sum_{h} p(A_i^c, h|\theta_c) p(X_i^c, h|\theta_c)) \tag{5.3}$$

The generative approach can lead to difficulties when attempting to distinguish between similar object categories. Fig. 5.3 shows several images from two similar classes, Bicycles and Motorbikes, and the corresponding locations of the best hypothesis. The relative positions and appearance of the parts seem similar between the two classes, possibly leading to confusion during classification tasks.

5.4 Discriminative Model

Here we consider a discriminative formulation for learning object categories. We consider the log conditional likelihood of all the classes given all the data. As for the generative model, we assume that our models can be described by a parameter vector θ_c. By maximizing the Conditional Likelihood expression (CL), we are maximizing the probability that each image

[3]One of the major limitations of the constellation model is the computational cost induced by the combinatorial explosion relating the number of interest points used and the number of model components.

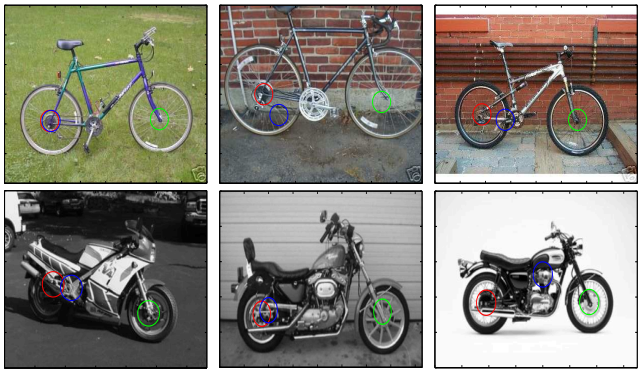

Figure 5.3: Examples of generative Bicycle and Motorcycle models. The circles indicate the positions of the best hypothesis. The generative models tend to model the wheels of the classes.

(represented by I_i^c) belongs to its own label (c):

$$log(\prod_c \prod_{i \in C} p(c|I_i^c)) = \tag{5.4}$$
$$\sum_c \sum_{i \in C} \Big\{ log(p(I_i^c|c)) + log(p(c)) - log(p(I_i^c)) \Big\}$$

Next we expand the partition function $p(I_i^c)$ which corresponds to the probability of a data-point from the current class belonging to any of the other classes. We index the 'competing' classes by g. Furthermore, we remove the prior probability of a class, $p(c)$, as it is independent of the parameter we are optimizing, θ_c. We obtain:

$$\sum_c \sum_{i \in C} \Big\{ \underbrace{log(p(I_i^c|\theta_c))}_{ML} - \underbrace{log\Big(\sum_g p(I_i^c)|\theta_g)p(g) \Big)}_{PartitionFunction} \Big\} \tag{5.5}$$

The equation for maximizing CL consists of two terms, the first is the maximum likelihood term used in the generative approach, from which we subtract the second term, the partition function. Intuitively, the expression maximizes the probability of data belonging to its own class label, while simultaneously minimizing the probability that the data was generated by any of the class labels. Finally, we note that the relative strength of the ML and CL terms can be weighted using a term α, thereby allowing for a continuum between purely ML and purely CL models:

$$\sum_c \sum_{i \in C} \Big\{ \underbrace{log(p(I_i^c|\theta_c))}_{ML} - \underbrace{\alpha \cdot log\Big(\sum_g p(I_i^c)|\theta_g)p(g) \Big)}_{PartitionFunction} \Big\} \tag{5.6}$$

Where $\alpha = 0$ is a an entirely ML approach and $\alpha = 1$ a pure CL approach. Varying the value of α is useful for clearly illustrating the relative merits of generative and discriminative techniques. All our object models used a complete discriminative model with $\alpha = 1$ unless otherwise specified.

Returning to the Motorcycle and Bicycle classes (Fig. 5.4), we notice that the model parts tend to represent the body of the classes rather than the wheels. The features along

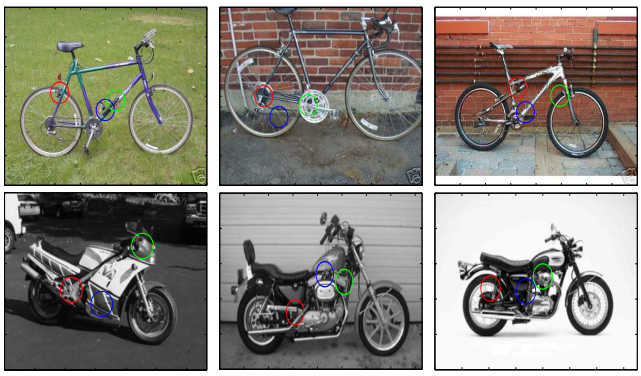

Figure 5.4: Examples of discriminative Bicycle and Motorcycle models. The best hypotheses tend to model the body of both the bicycle and motorcycle classes.

the body seem intuitively to provide more discriminative power.

5.4.1 Model Testing

The performance of both the generative and discriminative models can be assessed by presenting a novel test image, I_i^c and calculating the probability of this test image being generated by any given class: $p(I_i^c|\theta_c^*)$, where θ_c^* is an optimized ML or CL model. If the highest probability model corresponds to the class label for that image, the image is correctly classified. Note that finding the highest probability class using a discriminative decision, namely the class which has the maximum $p(I_i^c|\theta_c^*)p(c)/p(I_i^c)$ or $log(p(I_i^c|\theta_c^*)) - log(p(I_i^c))$, is equivalent to the approach above as the partition function, $p(I_i^c)$, is the same for all classes.

5.4.2 Model Optimization

We wish to maximize the expressions for both the generative and discriminative approaches derived above. We use the Expectation Maximization [DLR76] (EM) algorithm to optimize the generative models. Learning was terminated after 100 iterations had been reached. The conjugate gradient algorithm was used to optimize the CL models. Conjugate gradient require the derivative of the CL expression with respect to each parameter of the model. Learning was terminated when either the gradient at a particular iteration was below a threshold or the maximum number of iterations was reached. We chose random initial starting conditions to initialize both optimization routines.

By varying the value of α we can explore how the computational cost changes as the model becomes more discriminative (see Figure 5.5). We notice that the purely discrimina-

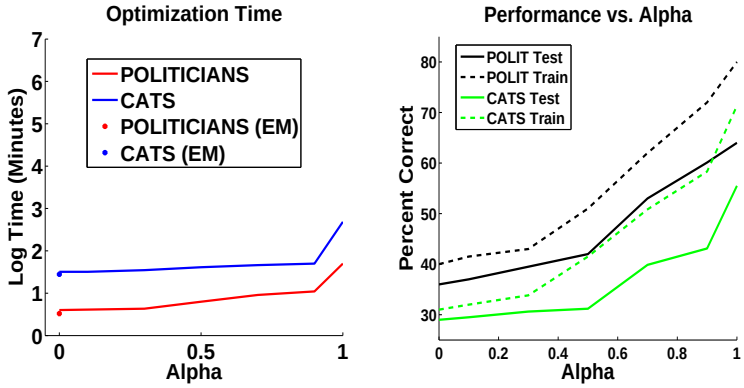

Figure 5.5: (Left) Optimization time as a function of α. At $\alpha = 0$ we have a pure generative approach and $\alpha = 1$ is a pure discriminative approach. Time scale in log base 10 minutes. The CL models take longer to optimize, $6\times$ longer for the Politicians data-set and $10\times$ longer for the Cats data-set. Results shown for both a 9 interest points, 100 train supervised 'Politicians' data-set and 20 interest points, 150 train Cat Species data-set. The red and blue stars indicate the time taken to optimize both sets of classes using the EM algorithm. (Right) Train and Test performance as a function of α for the Politicians data-set (black curves) and the Cat Species data-set (green curves). Increasing the discriminative power increases the performance as well as the amount of over-fitting as measured in the difference between the train and test performance.

tive function seems to be optimized over a more complicated energy landscape as the time for optimization increases significantly. There seems to be a steep increase in computational time when moving from $\alpha = .9$ to $\alpha = 1$, indicating that it might be more computationally favorable to not use the fully discriminative learning model. We compared both the performance and optimization time between the conjugate gradient algorithm with $\alpha = 0$ and the EM algorithm. We noted that the EM algorithm was slightly faster (Figure 5.5) but did not yield noticeably higher performance.

5.5 Experiments

We performed numerous experiments comparing the Conditional Likelihood approach with the Maximum Likelihood approach. First we show examples on a toy-data-set of "PacMen" examples in order to give the reader some intuition for the differences between the generative and discriminative paradigm compared here.

5.5.1 Toy Example: PacMen

To clearly illustrate the benefits of CL over ML we have constructed data-sets consisting of various rotated PacMen. Each class contains 3 PacMen, with two of three PacMen

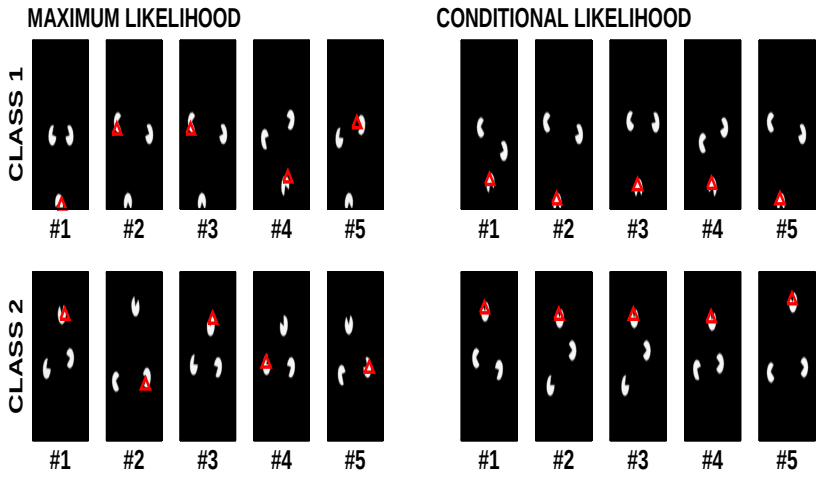

Figure 5.6: Features selected from PacMen classes. (Left) Features selected for a 1 component model using ML. The red triangles indicate the most likely feature selected. Top row are the selected features for Class 1, bottom row are the selected features for Class 2. We show the features found for 5 different runs of the algorithm. There seems to be no bias towards selecting any particular PacMan. (Right) Same as above but using CL optimization. The algorithm consistently picks out the unique PacMan in each class.

being the same in both classes and each class containing one uniquely rotated PacMan (the middle PacMan is unique to each class). We would expect CL to extract features which are consistent within the class but unique between the classes and thus chose to represent the middle PacMan. ML, lacking the knowledge of the PacMen in other classes, shows no bias towards the unique PacMan and thus performs poorly on models containing only 1 and 2 components. Both models have perfect classification performance for 3-part models. Figure 5.6 illustrates the results for a 1 component model.

5.5.2 Supervised Experiments

We initially compared generative and discriminative learning using a supervised data-set of Politicians in order to obtain a better understanding of their relative differences independent of the inconsistencies due to feature detectors. The Politicians data-set consists of images of John Kerry, George Bush, Arnold Schwarzenegger, and Bill Clinton. Each data-set contained about 150 images of which typically 100-120 were used for training. The resolution was around 200×200. See the appendix for information on data collection. Examples of the models found for a typical parameter setting are shown in Fig. 5.7. The highest likelihood hypotheses, as indicated by the circles in part (C) of Fig. 5.7, show that the ML learning

results in parts which model features within the face, while CL learning mostly results in models of the hair-line. The discriminative models appear to have found a more powerful set of features for discriminating between the classes as indicated by the higher classification performance of these models. In Fig. 5.10 we show the effects on performance of varying the number of training images on ML and CL models.

5.5.3 Semi-Supervised Discrimination

We performed experiments on 2 semantically related sets of classes using semi-supervised learning, Bikes (Motorbikes, Bicycles), and Cat Species (House Cat, Tiger, Lion, Cougar). The Cat classes contained about 230 images each with 200 being used for training, while the Bikes sets have about 450 images each with 370 being used for training. We used a maximum of 30 interest points per image. The Cats data-set often contains very similar looking images which make the discrimination task particularly difficult. Figures 5.8 and 5.9 compare ML and CL generated models for these sets of classes. In general, the CL models are more variable between classes than the ML models. Furthermore, we notice substantial improvements for both groups during discriminative training, with the Cats groups showing roughly a 20% increase on average across all 4 classes and the Bike set showing a 15% increase across 2 classes to reach about 90% performance. However, discriminative learning resulted in significant over-fitting and thus required many training examples to reach this performance level. The increased performance comes at a cost, namely many training examples. We also compared discriminative and generative performance on two very different classes, namely Airplanes and Human Faces. Here we do not notice significant changes in performance, indicating that discriminative learning may not be useful when the object classes of interest come from very different categories. Fig.5.10 compares the performance as a function of the number of training examples for all semi-supervised models.

5.5.4 Generative/Discriminative System

The previous sections indicate that discriminative learning can result in substantial performance gains over generative learning, but that these gains come at the price of both increased computational resources and large numbers of training examples. This suggests a natural system for training large numbers of object categories, namely initially train models in a generative fashion, identify areas of high confusion between classes, and train these

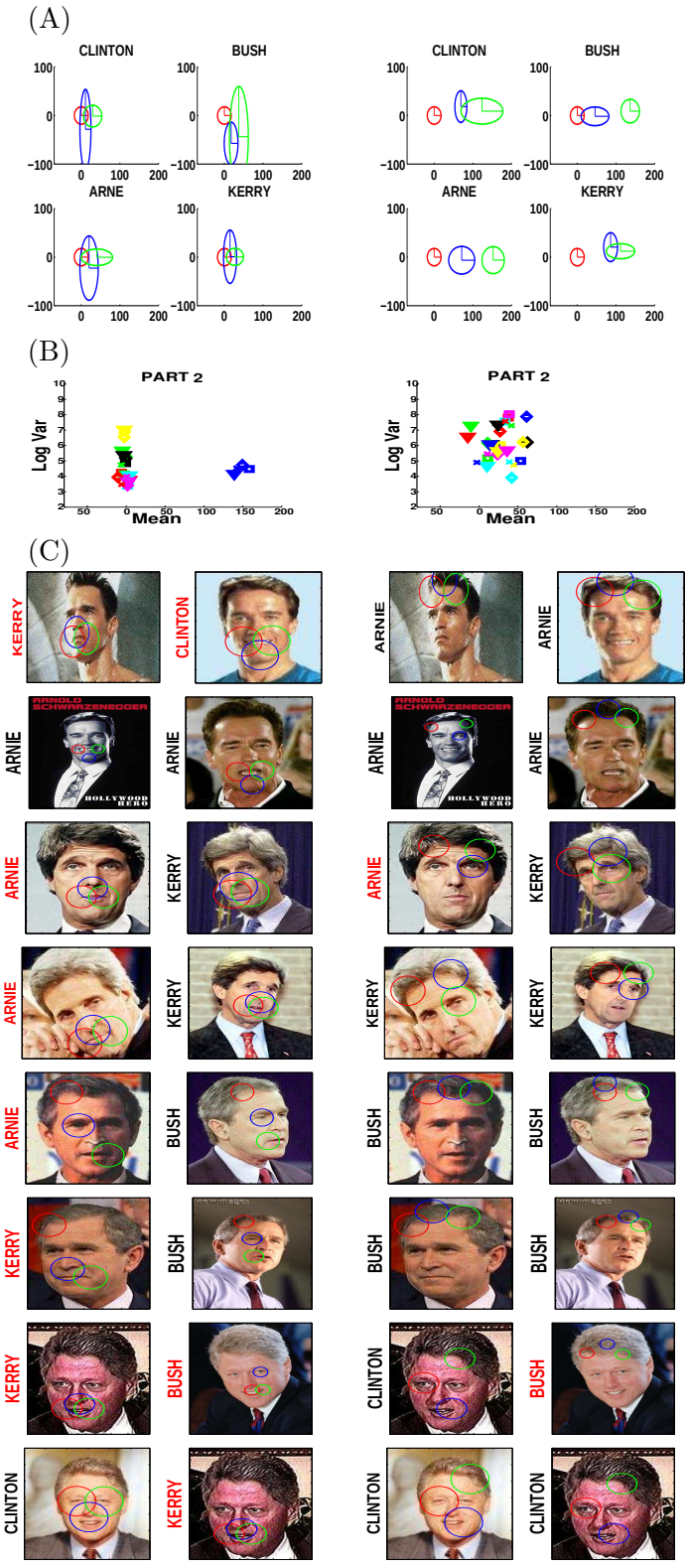

Figure 5.7: Generative (ML) and Discriminative (CL) Politician Models. Left column ML models, right column CL models. (A) The shape models for each class. Each different color circle represents a unique part. (B) Plots of the mean vs the log (base e) of the variance for the first 7 PCA components. Only Part 2 shown although it is representative of the other parts. Each unique shape represents a different class and each color a different PCA coefficient. The ML models are more tightly bunched between classes than the CL models as indicated by the clustering of colors. (C) The locations of the best matching hypothesis in the same images for both ML and CL models. Incorrect hypothesis are marked in red. The CL models are focusing more on the hair-line for the Arnie, Kerry, and Bush classes.

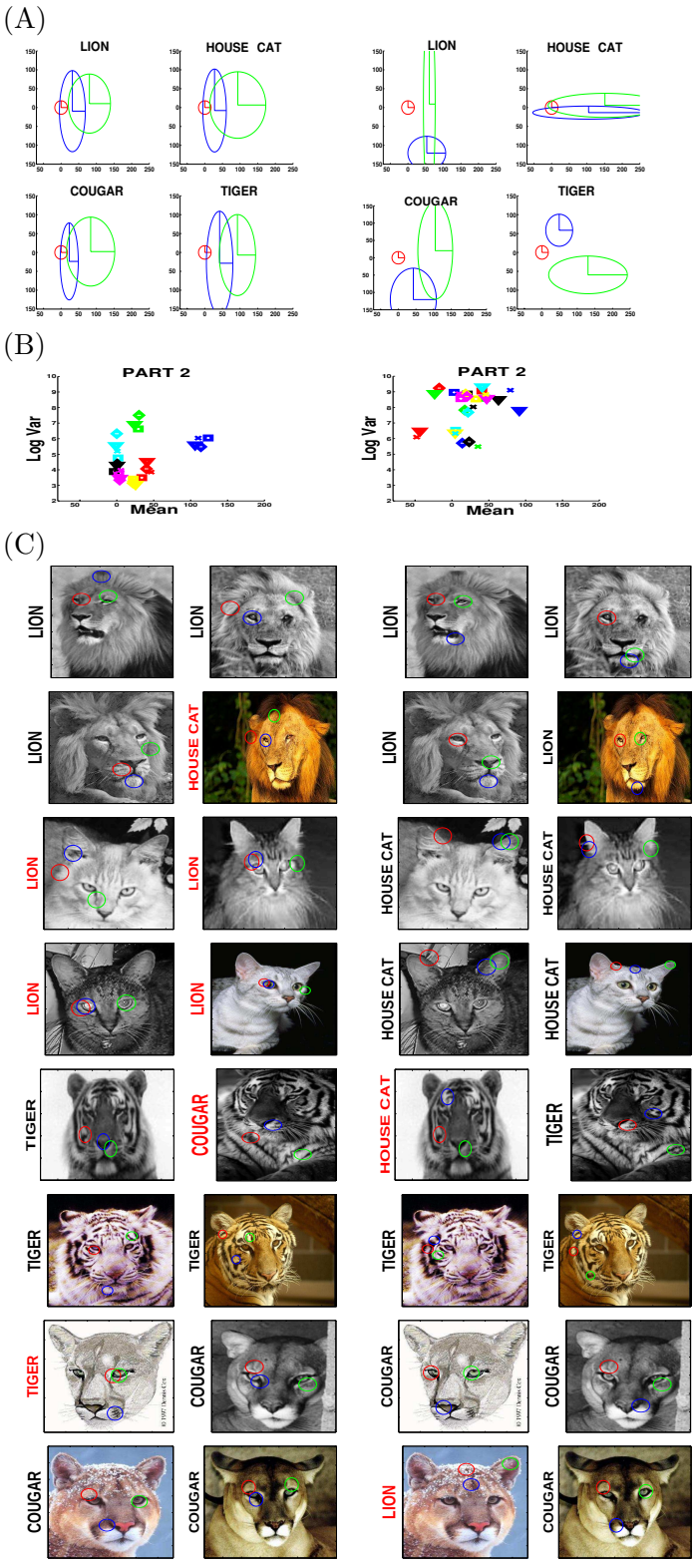

Figure 5.8: Generative (ML) and Discriminative (CL) Models of Cats Species. (A) The CL shape classes are more distinct. (B) The CL appearance models also seem more distinct. (C) ML shape models are broad indicating that the models are focusing on the appearance of patches rather than their relative positions for modelling the object classes. The CL shape models are tighter indicating that there is useful discriminative power in the mutual positions of the parts.

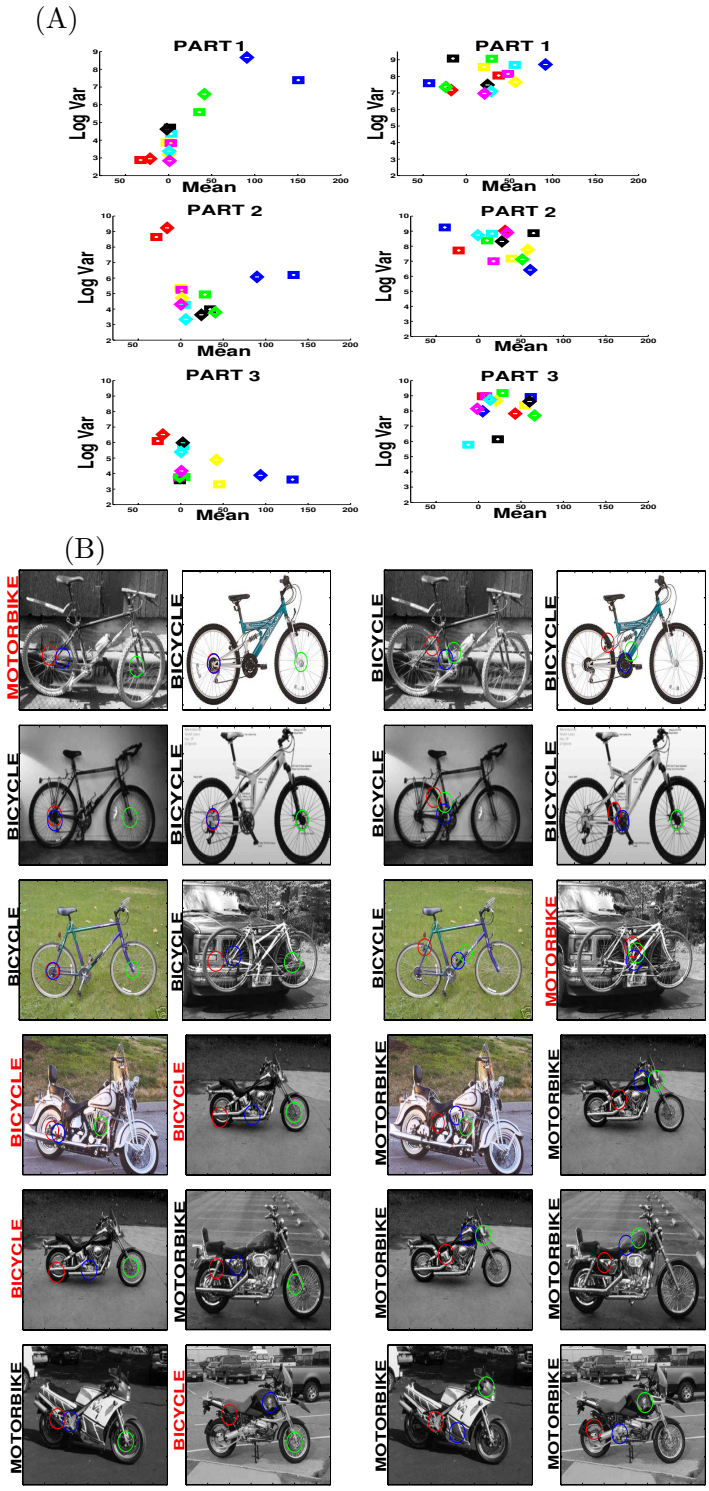

Figure 5.9: Generative (ML) and Discriminative (CL) Models of Bicycles and Motorcycles. ML left column, CL right column. (A) Appearance models for all 3 parts, first 7 coefficients. The ML appearance models tend to be more clustered across classes indicated by the close proximity of same colored points. (B) The highest probability hypothesis noted by the colored circles. The ML models tend to represent the wheels of both bicycles and motorbikes while the CL models tend to focus on the body indicating that features on the body have higher discriminative power than the wheels.

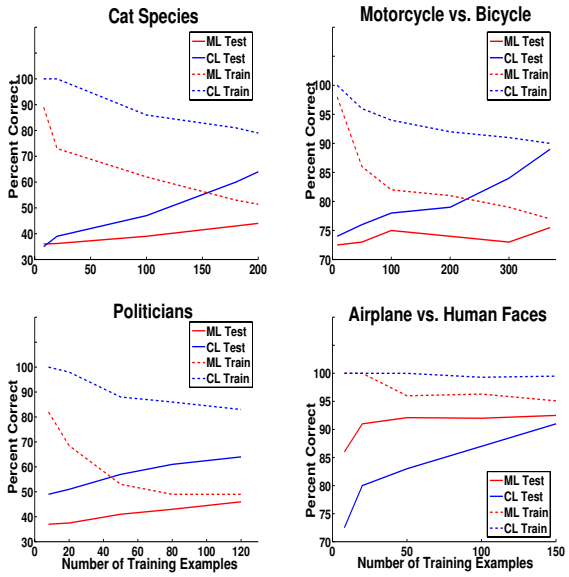

Figure 5.10: Performance plots as a function of the number of training examples. Data-sets shown are Cat Species (4 classes), Bikes (2 classes), Politicians (4 classes), and Airplanes vs. Human Faces (2 classes). CL tends to outperform the ML models when the classes are similar, but does not show significant performance improvements when the classes are distinct (e.g. Airplanes vs. Human Faces) as the generative models exhibit high performance. We also notice over-fitting for all discriminatively trained models.

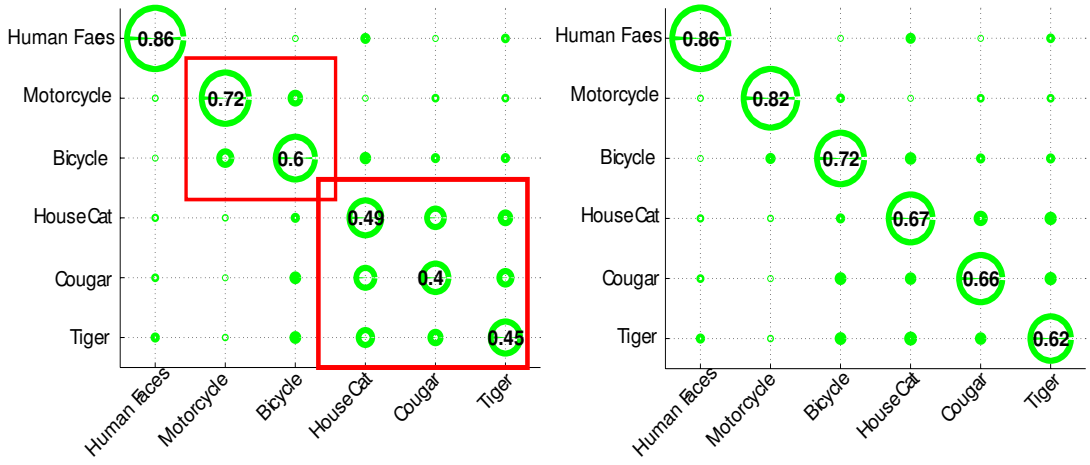

Figure 5.11: (Top) Initial confusion table for 6 classes. The x-axis indicates the true category of the image and the y-axis the predicted category for the image. An (x, y) entry indicates the fraction of times class x was classified as y. The values along the main diagonal are the percent of the time an image was correctly classified. The size of the green dots indicates the magnitude of confusion. Notice that there are subsets of classes which have higher confusion among themselves, namely Cat Species and Bikes. (Bottom) Confusion table after discriminative learning of the Cat Species cluster and the Bike cluster. The performance on the Human Face class does not change. Both discriminative and generative models were trained using the same subset of data. Confused classes were identified using the Training set. Confusion tables shown here indicate Test set performance.

subsets of confused models using discriminative methods, using high numbers of training examples if necessary. We implemented such a system using a set of 6 classes: Cats Species (House Cat, Tiger, Cougar), Bikes (Motorcycles, Bicycles), and Human Faces. The confusion table created from the generative models appeals to our semantic notion of similarity, with the subsets Cat Species and Bikes exhibiting higher confusion among themselves than between other classes (Fig. 5.11). These 2 subsets are ideal candidates for discriminative learning. Using the same training data, we create two independent sets of discriminative models for the Cat Species and Bikes. Test examples are first grouped into a super-set of Cat Species, Bikes, or Human Faces according to the initial generative model, followed by discriminative classification for examples which fall into super-set category containing several classes. Fig. 5.11 shows that this hierarchical method of initial generative, than discriminative classification, results in good performance gains for previously confused classes (%20 improvement for Cat Species and %11 for Bikes).

5.6 Conclusion

We have demonstrated a discriminative paradigm to learn object class categories. We show how a discriminative setting can be used to improve object categorization performance when generative settings cause substantial amounts of confusion between classes. We illustrate the intuitive appeal of discriminative learning, namely the selection of discriminative features. We also note the tradeoff between discriminative and generative techniques, with discriminative techniques outperforming their generative counterparts but requiring both a larger number of training images and a larger computational resources. This provides us with intuition for when discriminative techniques are most suitable, namely when object classes contain high numbers of overlapping features, thereby allowing discriminative learning to select the most informative features for learning each object class. We concluded by proposing an object recognition system which initially trains models in a generative way and then separates the representations of similar classes via discriminative learning.

5.7 Appendix: Data Collection

The Motorcycle, Airplane, and Human Face data-sets are from the Caltech Image DataBase. We collected images of all other object from the web using the Google, Yahoo, and

Lycos search engines. Images were sometimes cropped to emphasize the category of interest and reduce the number of non-object features detected for semi-supervised learning.

Chapter 6

Fisher Kernels and Extensions

6.1 Abstract

Learning models for detecting and classifying object categories is a challenging problem in machine vision. While discriminative approaches to learning and classification have, in principle, superior performance, generative approaches provide many useful features, one of which is the ability to naturally establish explicit correspondence between model components and scene features – this, in turn, allows for the handling of missing data and unsupervised learning in clutter. We explore a hybrid generative/discriminative approach, using 'Fisher Kernels' [JDH99], which retains most of the desirable properties of generative methods, while increasing the classification performance through a discriminative setting. Our experiments, conducted on a number of popular benchmarks, show strong performance improvements over the corresponding generative approach. In addition, we demonstrate how this hybrid learning paradigm can be extended to address several outstanding challenges within computer vision including how to combine multiple object models and learning with unlabelled data.

6.2 Introduction

Detecting and classifying objects and object categories in images is currently one of the most interesting, useful, and difficult challenges for machine vision. Much progress has been made during the past decade in formulating models that capture the visual and geometrical statistics of natural objects, in designing algorithms that can quickly match these models to images, and in developing learning techniques that can estimate these models from training images with limited supervision [BP96, UVNS02, WWP00, FPZ03, Low04, DS04, TMF04, LS04, HP05, GHW06] However, our best algorithms are not close to matching human

abilities. Machine vision systems are at least two orders of magnitude worse than humans in several aspects, including the number of categories that can be learned and recognized, the classification error rates, the classification speed, and the ease and flexibility with which new categories can be learned. Some of this work was presented in [HWP05a, HWP07] and was done in collaboration with Max Welling.

This work is motivated by the challenge of learning to recognize categories that look similar to one another. A number of methods have shown good performance on dissimilar categories (for example airplanes, automobiles, spotted cats, faces and motorcycles as in [FPZ03, DS04]). None of these methods has been shown to perform well on visual categories which look similar to one another such as bicycles and motorcycles or male and female faces. For example, while the 'constellation model' [FPZ03] has error rates of a few percent on dissimilar categories such as faces vs. airplanes and cars vs. cats, it has error rates around 30% if it is asked to recognize faces of different people (see the x-axis of the plots in Fig. 6.1). Why does this discrepancy exist? As we shall see, one potential confound is the underlying generative learning algorithm.

Learning and classification methods fall into two broad categories (see Figure 6.2). Let y be the label of the class and x the measured data associated with that class. A **generative** approach will estimate the joint probability density function $p(x, y)$ (or, equivalently, $p(x|y)$ and $p(y)$) and will classify using $p(y|x)$ which is obtained using Bayes' rule. Conversely, **discriminative** approaches will estimate $p(y|x)$ (or, alternatively, a classification function $y = f(x)$) directly from the data. It has been argued that the discriminative approach results in superior performance, i.e. why bother learning the details for models of different classes if we can directly learn a criteria for discriminating between the classes [Vap98]? Indeed, it was shown that the asymptotic (in the number of training examples) error of discriminative methods is lower than for generative ones when using simple learning models [NJ02].

Yet, among machine vision researchers, generative models remain popular [UVNS02, WWP00, FPZ03, DS04, LS04, Sch04]. There are at least five good reasons why generative approaches are an attractive choice for visual recognition. First, generative models naturally incorporate information about occlusion and missing features. This is because generative methods allow one to establish explicit 'correspondence' between parts of the model and features in the image. For every such mapping, the parts in the model corresponding to the missing features can simply be marginalized out of the probabilistic model, leaving us with

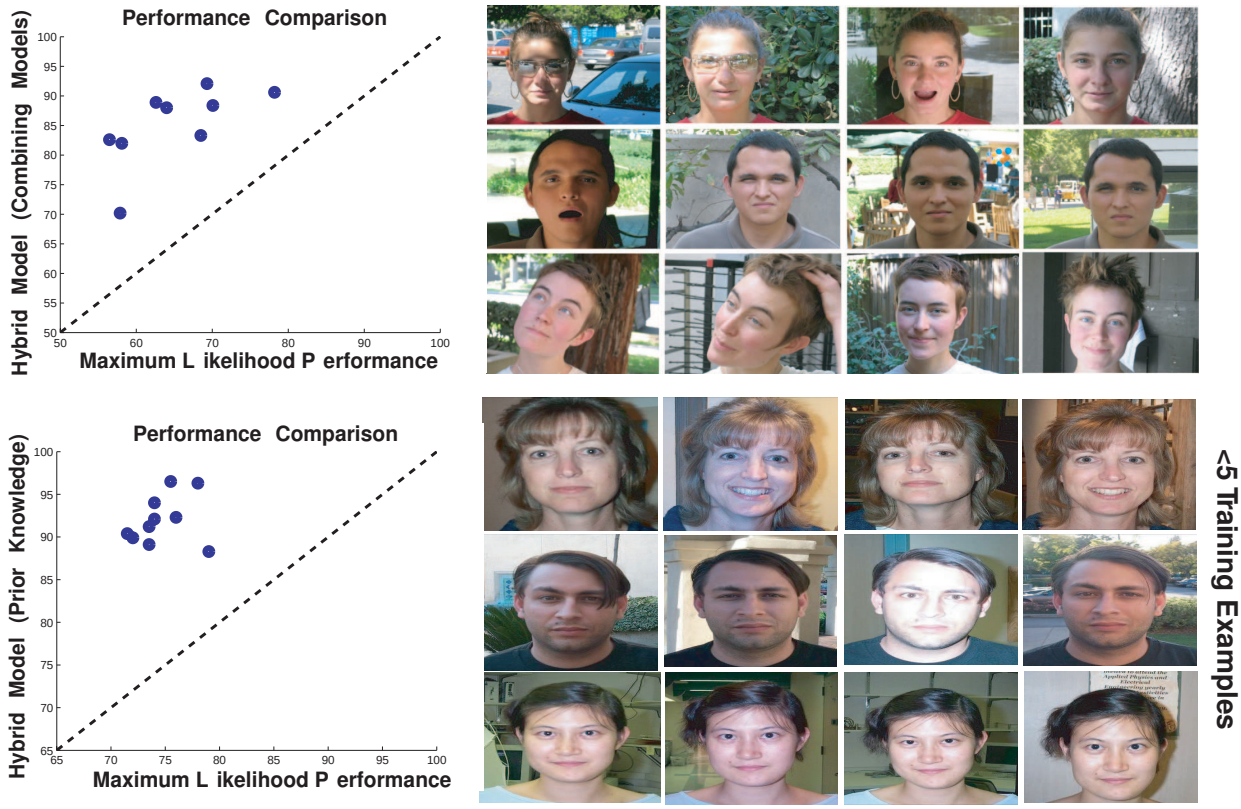

Figure 6.1: A pure generative Maximum Likelihood (ML) approach will not work well when categories are similar in appearance (right column images of faces, each row shows a different person), especially when few training examples are available (scatterplots on the left, x axis). We apply discriminative techniques from Jaakkola et al. [JDH99] to transform generative approaches for visual recognition into discriminative classifiers which retain some of the desirable properties of generative models and perform much better (scatterplots on the left, y axis). (Top) ML in comparison to using combinations of hybrid models. See section 6.7 below for details. Category labels for these faces, from top to bottom: P1, P2, and P3. These new face categories will be posted on the web. (Bottom) ML in comparison to hybrid models in a semi-supervised learning paradigm in which few examples (in this case three training examples) are present. See section 6.6 below for details.

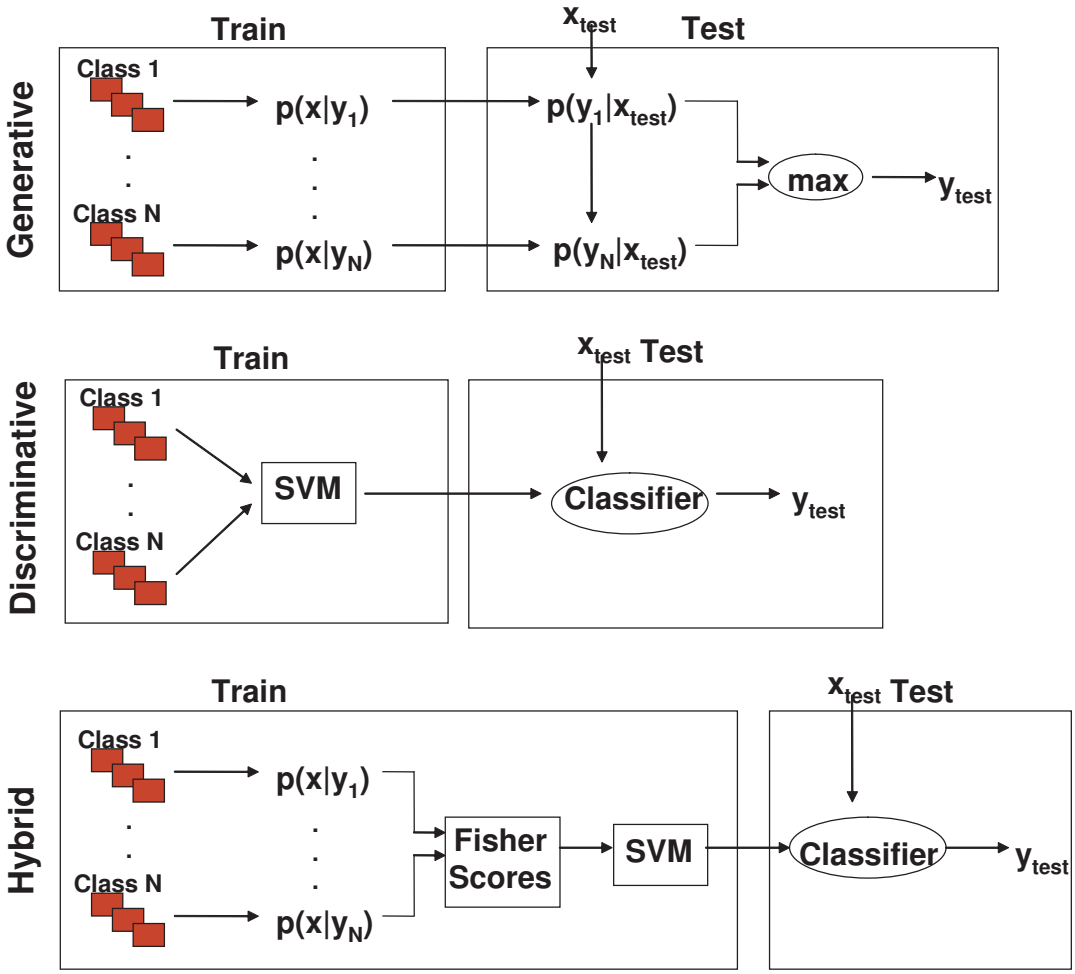

Figure 6.2: Schematic comparison of the generative (top), discriminative (middle), and hybrid (bottom) approaches to learning discussed in this paper. While generative models are a natural choice for visual recognition, discriminative models have been shown to give better performance in different domains. The hybrid model captures many desirable properties of both.

a lower dimensional model over the observed parts [WWP00]. Second, collecting training examples is expensive in vision and training sets come at a premium. Ng and Jordan [NJ02] demonstrated both analytically and experimentally that in a 2-class setting the generative approach often has better performance for small numbers of training examples, despite the asymptotic performance being worse. Third, it has been shown that prior knowledge can be useful when few training examples are available, and that prior information may be easily incorporated in a generative model [LFP06]. Fourth, we ultimately envision systems which can learn thousands of categories; in this regime it is unlikely that we will be able to learn discriminative classifiers by considering simultaneously all the training data. It is therefore highly desirable to design classifiers that can learn one category at a time: this is easy in the generative setting and difficult in the discriminative setting where training data for all categories must be available at once for a decision boundary to be calculated. Fifth, it is unclear, in general, what features to use when training a discriminative classifier on object categories. Consider that many popular algorithms for object recognition rely on feature detectors to find 'interesting' regions within an image. Each image thus is represented as an unordered set of feature detections of variable length. How can these unordered lists be used by a discriminative classifier?

Is it possible to get the best of both worlds and develop approaches with the flexibility of generative learning and the performance of discriminative methods? Jaakkola and Haussler have shown that a generative model can be used in a discriminative context by extracting Fisher Scores from the generative model and converting them into a 'Fisher Kernel' [JDH99] (see Figure 6.2). A kernel represents the data as a matrix of pairwise similarities which may be used for classification by a kernel method, such as the support vector machine (SVM). The field of kernel methods is well developed [Vap98, STC04, SS02, GHS05] and represents the state of the art in discriminative learning. Here, we explore how to apply these ideas to visual recognition.

We calculate Fisher Kernels that are applicable to visual recognition of object categories and explore experimentally the properties of such 'hybrid models' on a number of popular and challenging data-sets. Other kernel-based approaches have been suggested for object recognition, including Vasconcelos et al. [VHM04] who exploit a similar paradigm, using a Kullback-Leibler based kernel and test on the COIL data-set. Wallraven et al. [WCG03] utilize a clever kernel which implicitly compares detected features in different images, but

apply their method to different sets of images than those used in this paper.

In Section 6.3 we briefly review one class of generative models, commonly called the 'Constellation Model', which will be used in the rest of the paper. In section 6.4 we show how transform a generative Constellation Model into a discriminative setting by utilizing the idea of Fisher Kernels. In Section 6.5 we compare the performance of hybrid and generative constellation models. In Section 6.6 we explore how these hybrid models can be extended and effectively used in circumstances where we have a mixture of labelled and unlabelled data, i.e. 'semi-supervised' learning. Finally, in Section 6.7 and 6.8 we show how the hybrid framework can be used to optimally combine several generative models (for example generative models based on different feature detectors and different numbers of parts) into a single classifier. Section 6.9 discusses the main results and observations of this work.

6.3 Generative Models

In this section we briefly review a class of generative models which will be used in conjunction with the discriminative methods described in the next section. In principle any generative model that is differentiable with respect to its parameters can be used. We chose to experiment with the 'Constellation Model' which was first proposed by Burl et al. [BP96]. Weber et al. [WWP00] showed that this model may be learned from cluttered images in a weakly supervised setting in which only a class label is associated with each image using maximum likelihood. Fergus et al. [FPZ03] extended the model by making it scale-invariant and incorporating general purpose feature detectors. We use a simplified version of Fergus' constellation model in which we do not explicitly model occlusion or relative scale.

6.3.1 The Constellation Model

The constellation model is a generative framework which constructs probabilistic models of object classes by representing the appearance and relative position of several object parts [BP96, WWP00, FPZ03]. Given a suitable training set composed of images containing examples of objects belonging to a given category, such models are trained by finding a set of model parameters θ^{MLE} which maximizes the log-likelihood of the model [WWP00, FPZ03]. Both appearance and shape are modelled as jointly Gaussian and $\theta = \{\theta_a, \theta_s\}$ represent

the mean and diagonal variance parameters of the shape (θ_s) and appearance models (θ_a). To remove dependence on location, the x-y coordinates of the parts are measured relative to a reference part, e.g. the left-most part. In our implementation, as suggested by Fergus et al. [FPZ03], appearance is represented by the first 20 PCA components of normalized 11×11 pixel patches which were cut out around feature detections in training images at the scale indicated by the detectors (see next subsection). The number of interest point detections considered in an image is a design parameter.

For each training image I_i we obtain a set of F interest points and their appearance descriptors. We would like to establish correspondence, i.e. assign a unique interest point to every model part, or component, M_j. Burl et al [BP96] showed that since we do not a priori know which interest point belongs to which model component, we need to introduce a 'hidden' hypothesis variable h which maps interest points to model parts. We order the interest points in ascending order of x-position. Note that although we model only the diagonal components of the Gaussian, the model parts are not independent as we enforce that each part is mapped to a unique feature, implicitly introducing dependencies. The result is a total of $\binom{F}{M}$ hypotheses, where each h assigns a unique interest point to each model part. We marginalize over the hypothesis variable to obtain the following expression for the log likelihood for a particular class:

$$\sum_i \log\left(p(I_i)\right) = \sum_i \log\left(\sum_h p(A_i, h|\theta_a) p(X_i, h|\theta_s)\right) \tag{6.1}$$

where $\{X_i\}$ are the relative coordinates of the object and represent the shape information while $\{A_i\}$ are the PCA components described above and represent the appearance information. We assume that the shape and appearance models are independent of one another given a hypothesis h and that the images are I.I.D. This step is key to maximum-likelihood model learning, and to classification, once a model is available (see details in [BP96, WWP00, FPZ03]). We note that exploring all possible hypotheses carries a combinatorial computational cost which severely limits the number of parts and interest points which can be used in the model.

For clarity we consider the makeup of a typical set of parameters θ. Consider one part of a 3-part model. A single part consists of parameters specifying its shape and appearance, θ_s and θ_a respectively. The shape of the part is specified by a two dimensional mean and two

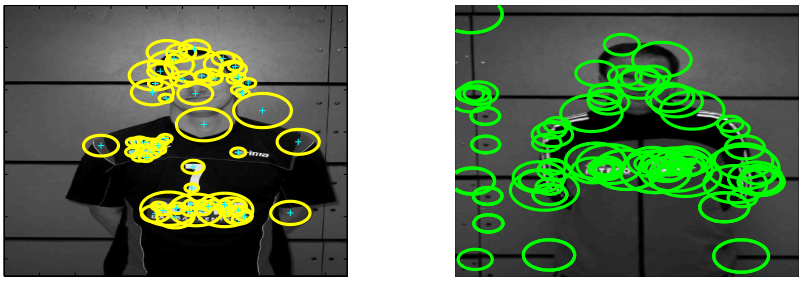

Figure 6.3: Examples of scaled features found by the KB (left) and multi-scale DoG (right) detectors on images from the 'persons' data-set located at http://www.emt.tugraz.at/ opelt/. Approximately the top 50 most salient detections are shown for both.

dimensional variance (we consider diagonal covariance matrices in our model) indicating the mean and variance of the position of the part. Each part thus has a four dimensional parameter array specifying its location. Now consider the appearance parameters of a part. The appearance of a part is specified by the mean and variance of the PCA components for that particular part. A 20 dimensional PCA representation thus consists of a total of 40 parameters, 20 for the mean and 20 for the variance of the part.

6.3.2 Interest-Point Detection

The constellation model requires the detection of interest points within an image. Numerous algorithms exist for extracting and representing these interest points. We considered several popular interest point detectors: the entropy based Kadir and Brady (KB) [KB01] detector, the multi-scale Difference of Gaussian (DoG) detector [Cro84], the multi-scale hessian detector (mHes), and the multi-scale Harris detector(mHar). Figure 6.3 shows typical interest points found within images. All detectors indicate the saliency of interest points, and only the most salient interest points are used. The KB interest point detector was used in all experiments below unless otherwise specifically noted.

6.3.3 Generative Model Learning

We train our generative constellation models using the EM algorithm [DLR76] as computed explicitly for the constellation model by Weber et al. [WWP00]. The algorithm involves iteratively calculating the expected values of the unobserved variables of the model and then maximizing the parameters. The algorithm was terminated after 50 iterations or after the

log likelihood stopped increasing. A typical 3-part model optimized on 100 images with 25 detections in each image took on the order of 20 minutes to optimize using a combination of C (mex) and Matlab code.

6.4 Fisher Scores and Fisher Kernels

For supervised learning, such as regression and classification, kernel methods are often the method of choice. As argued in the introduction, our interest is in *combining* generative models with a discriminative step for the purpose of visual object recognition. We chose support vector machines (SVM) [Vap98] as our kernel machine. The SVM (like all kernel methods) process the data in the form of a kernel matrix (or Gram matrix), a symmetric and positive definite $n \times n$ matrix of similarities between all samples. A simple way to construct a valid kernel matrix is by defining a set of features, $\boldsymbol{\phi}(x_i)$, and to define the kernel matrix as,

$$K_{i,j} = K(x_i, x_j) = \boldsymbol{\phi}^T(x_i)\boldsymbol{\phi}(x_j) \tag{6.2}$$

The kernel represents the similarities between samples: relatively large kernel entries correspond to two samples which are similar while small (possibly negative) entries correspond dissimilar samples. Kernels defined by inner products such as the one in Equation 6.2 produce positive-definite kernel matrices [Vap98].

The generative model will have its impact on the classifier through the definition of these features. We will follow [JDH99] in using "Fisher Scores" as our features. Given a generative probabilistic model the 'Fisher Scores' $\phi(x_i)$ are defined as

$$\boldsymbol{\phi}(x_i) = \frac{\partial}{\partial \boldsymbol{\theta}} \log p(x_i|\boldsymbol{\theta}^{\mathrm{MLE}}) \tag{6.3}$$

where $\boldsymbol{\theta}^{\mathrm{MLE}}$ is the maximum likelihood estimate of the parameters $\boldsymbol{\theta}$. By definition, $\boldsymbol{\theta}^{\mathrm{MLE}}$ is obtained by maximizing the likelihood. A necessary condition is that the gradient of such likelihood (or log-likelihood) is zero, which is equivalent to 'balancing' the Fisher Scores,

$$\sum_i \frac{\partial}{\partial \boldsymbol{\theta}} \log p(x_i|\boldsymbol{\theta}^{\mathrm{MLE}}) = \sum_i \boldsymbol{\phi}(x_i) = 0 \tag{6.4}$$

Hence, samples "pull" on the parameter values through their Fisher Scores which can be interpreted as "forces". At the MLE all forces balance. Two data-items that exert similar

'forces' on all parameters have their feature vectors aligned resulting in a larger positive entry in the kernel matrix.

Since it is not a priori evident that the data can be separated using a hyperplane in this feature space, it can be beneficial to increase the flexibility of the separating surface (making sure that the problem is properly regularized) as shown in [Vap98]. This is achieved by applying non-linear kernels such as the RBF kernel or the polynomial kernel in this new feature space, i.e. $K(\boldsymbol{\phi}(x_i), \boldsymbol{\phi}(x_j))$ with,

$$K_{\mathrm{RBF}}(x_i, x_j) = \exp\left(-\frac{1}{2\sigma^2}||\boldsymbol{\phi}(x_i) - \boldsymbol{\phi}(x_j)||^2\right) \tag{6.5}$$

$$K_{\mathrm{POL}_p}(x_i, x_j) = (R + \boldsymbol{\phi}(x_i)^T \boldsymbol{\phi}(x_j))^p \tag{6.6}$$

Where σ represents the variance of the RBF kernel and p represents the degree of the polynomial kernel being used.

To remove scale differences between the features we normalized the features before we computed their inner product,

$$\phi_a(x_i) \rightarrow \frac{\phi_a(x_i)}{\sqrt{\frac{1}{N}\sum_{j=1}^{N} \phi_a^2(x_j)}} \tag{6.7}$$

Why do we bother going through a two-stage process where we first train generative models for each object category and then train another classifier based on a kernel derived from those models, where we could also classify using log-likelihood ratios? The intuitive answer to this question is that a classifier is trained to find an optimal decision boundary, i.e. it focusses its attention to what is relevant to the task. Here, the samples which are close to the decision boundary carry much more information than the ones away from the boundary. In contrast, classifying according to likelihood ratios simply derives the decision boundary as a by-product from fitting models for every category. The objective of this fitting procedure is to maximize the probability of all samples for every category and not deriving a good decision boundary for the classification task at hand. This intuition has been made more precise in numerous papers. Most relevant to Fisher kernels is the theorem in [JDH99] stating that asymptotically (in the large data limit) a classifier based on the Fisher Kernel can be shown at least as good (and typically better) as the corresponding naive Bayesian procedure (i.e. likelihood ratios or maximizing $p(y|x)$). Similar results have

been obtained in e.g. [NJ02] and [TAKM03]. It should be mentioned that for small numbers of samples the naive Bayesian procedure may act as a regularizer and avoids the kind of over-fitting that can be observed in discriminative approaches.

Given a kernel matrix and a set of labels $\{y_i\}$ for each sample, the SVM proceeds to learn a classifier of the form,

$$y(x) = \text{sign}\left(\sum_i \alpha_i y_i K(x_i, x)\right) \tag{6.8}$$

where the coefficients $\{\alpha_i\}$ are determined by solving a constrained quadratic program which aims to maximize the margin between the classes. For details we refer to [STC04] and [SS02]. In our experiments we used the LIBSVM package (available at *http://www.csie.ntu.edu.tw/~cjlin/libsvm/*).

There are a number of design parameters: the free parameters in the definition of the kernel (i.e. σ in the RBF kernel and R, p in the polynomial kernel) and some regularization parameters in the optimization procedure of the $\{\alpha_i\}$. For an SVM the regularization parameter is a constant C determining the tolerance to misclassified samples in the training set. Values for all design parameters were obtained by cross-validation or by learning them on a bound of the leave-one-out error (see Section 6.7).

In Figure 6.4 we compare the performance of the various kernels defined above on two data-sets. The performance is similar, with some variability between data sets. We used linear and RBF kernels in the following experiments as there was no appreciable difference in the performance of the various kernels and because these kernels are popular within the machine learning community.

6.4.1 Fisher Scores for the Constellation Model

In order to train an SVM we require the computation of the Fisher Score for a model. Recall that the Fisher Score is the derivative of log likelihood of the parameters for the model, i.e.:

$$\frac{\partial}{\partial \theta_s} \log\left(p(I_i|\theta)\right) = \sum_h p(h|I_i, \theta) \frac{\partial}{\partial \theta_s} \log p(X_i, h|\theta_s) \tag{6.9}$$

$$\frac{\partial}{\partial \theta_a} \log\left(p(I_i|\theta)\right) = \sum_h p(h|I_i, \theta) \frac{\partial}{\partial \theta_a} \log p(A_i, h|\theta_a) \tag{6.10}$$

where both $\{\theta_a, \theta_s\}$ consist of mean and variance parameters of Gaussian appearance and shape models. Despite a potentially variable number of detections in each image I_i its Fisher

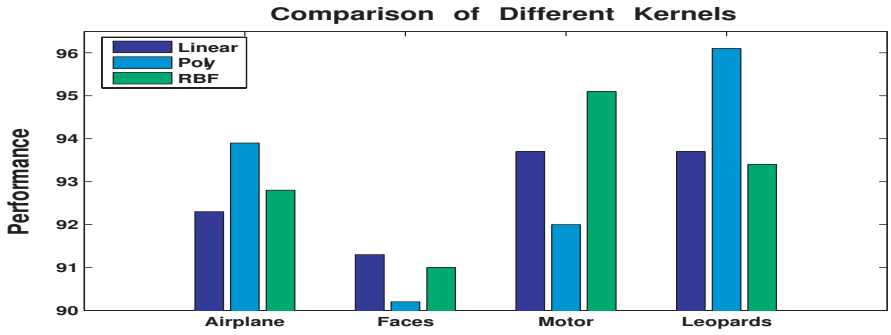

Figure 6.4: Performance comparison of various kernels on several data-sets. The parameters used to train and test these models are described in the experimental section. The polynomial kernel was of degree 2. The y-axis indicates the classification performance, note that the scale starts at 90%. These results were averaged over 5 experiments. 100 train/test examples used. First bar in each set: Linear kernel. Second bar: Polynomial kernel. Third bar: RBF kernel.

Score has a fixed length. This is because the hypothesis h maps features to a pre-specified number of parts and hence there is a fixed number of parameters in the model.

Most of the execution time of the algorithm is spent during the computation of the Fisher kernels as well as the training of the generative models. In comparison, the SVM training, even with extensive cross-validation, is quite short due to the relatively small number of training images.

6.5 Comparing Generative and Hybrid Approaches

Our first set of experiments was designed to compare the performance of our hybrid method with generative models on a commonly used benchmark of 4 object categories with diverse appearances from one another. We chose the same categories used by [FPZ03] in order to directly compare with their results (their results were obtained using a more sophisticated generative Constellation Model than the one we used).

Some details of the SVM training: Fisher Scores were normalized to be within the range [-1,1]. We performed 10x cross-validation to obtain estimates for the optimal values of C. For the RBF kernel, which contains the additional hyper-parameter σ, we found the optimal values of C and σ by performing an exhaustive search over the parameters. We varied the cross-validation search space for both parameters on a log base 2 scale from -7,9 in steps of 1 for C and -8,0 in steps of 2 for σ. We chose this range because the best results were typically within these parameter settings.

6.5.1 Experiments on Caltech Data Sets

Table 6.1 illustrates the performance on these data-sets[1]. Images were normalized to have the same number of pixels in the x-dimension in order to prevent the algorithm from learning meaningful information from the absolute size of the images. We did not crop the images. We recognize that although these data-sets are used extensively by the vision community, they exhibit some deficiencies. In particular the object of interest is usually in the center of the image and the objects are mostly in standardized poses and good lighting conditions. In these experiments a single generative model was created of the foreground class from which Fisher Scores were extracted for both the foreground and background classes for

[1]The data-sets, including the background data-set used here, can be found at: http://www.vision.caltech.edu/html-files/archive.html. The Leopards data-set is from the Corel Data-Base

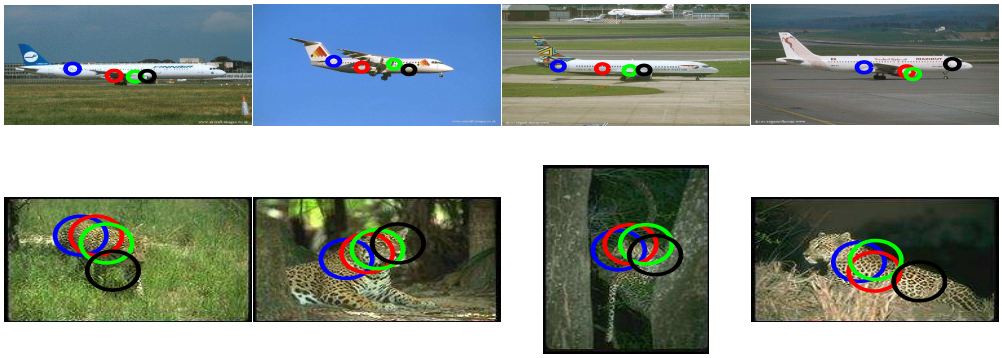

Figure 6.5: Localization of objects within images using the generative constellation model. Each unique colored circle represents a different part of the model. This is a 4-part model. The positions of the circles represent the hypothesis, h, with the highest likelihood. The generative framework approximately localizes the position of each object. We note that the images shown here do not exhibit excessive amounts of clutter.

Category	Hybrid	Shape	App	ML	Prev
Faces	**91**	77.7	88.9	83	89.4 [Fer05]
Motorcycles	**95.1**	74.5	91.2	74.2	96.7 [Fer05]
Airplanes	**93.8**	95.3	84.2	72.4	92.2 [Fer05]
Leopards	**93**	71.8	91.3	68.1	88 [Fer05]

Table 6.1: Performance comparison for some Caltech data-sets in a 2-alternative task whether the test image either contains an object from a given class, or contains no object (background class). We used 100 training and test images for each class. The background class was the same used by [Fer05]. All scores quoted are the total number correct for both the target class and the background over the total number of examples from both classes. The second column shows the performance of our hybrid discriminative algorithm. The third and fourth columns show performance using only the Shape and Appearance Fisher Scores respectively. The Fifth column is the performance using a likelihood ratio on the underlying generative models. The final column shows previous performances on the same data-sets [Fer05]. Our underlying generative model contained 3 parts and used a maximum of 30 detected interest points per image. Results were averaged over 5 experiments. In comparing with [Fer05] note that in that study approximately twice as many training images were used, as well as a more sophisticated generative constellation model (6 parts, scale-invariance, occlusion modelling), hence the higher performance of [Fer05] with respect to our baseline generative constellation model (ML). On the other hand, our hybrid method models relies on the background images for SVM training while the ML method of [Fer05] does not make explicit use of the background images

.

True Class ⇒	Motor	Leopards	Faces	Airplanes
Motorcycles	**96.7**	7.3	1.3	.3
Leopards	1.3	**90.7**	.7	0
Faces	1.3	0	**97**	.3
Airplanes	.7	2	1	**99.3**

Table 6.2: Confusion table for 4 Caltech data-sets in a 4-way classification experiment. The main diagonal contains the percent correct for each category. Perfect performance would be indicated by 100s along the main diagonal. We use the same classes as used in [FPZ03] which utilizes a purely generative constellation model and resulted in performances of 92.5, 90.0, 96.4, and 90.2 across the main diagonal for a 4-way discrimination task. When performing classification using only the Shape and Appearance Fisher Scores we achieve an average performance (average across the main diagonal of the confusion table) of 72.6 and 94.8. 100 training and testing images were used in our experiments. Averaged over 3 experiments.

training. The SVM was trained with the Fisher Scores from the foreground and background class. Testing was performed using an independent set of images from the foreground and background class by extracting their Fisher Scores from the foreground generative model and classifying them using the SVM. We refer to these experiments as 'class vs. background' experiments, as they involve a discrimination task between one foreground class and one background class.

In addition to class vs. background experiments, we conducted classification experiments using multiple object categories. First a generative model was constructed for all classes of interest. Fisher Scores for both train and test images were obtained by concatenating the Fisher Scores from each model. Only the training images were used to create both the SVM classifier and the generative distribution. Since an SVM is inherently a two-class classifier we train the multi-class SVM classifier in a 'one-vs-one' manner. For each pair of classes a distinct classifier was trained. A test image was assigned to the category containing the largest number of votes among the trained classifiers. 'One-vs-one' classification requires that $\frac{N(N-1)}{2}$ classifiers be created and used during classification. 'One-vs-one' clearly introduces a large number of classifiers which must be evaluated during run-time. However, we prefer it to a 'one-vs-all' strategy as 'one-vs-all' is not as amenable to unbalanced data-sets and the potential large computational cost is not as evident for small numbers of training examples. All other parameters for training were kept the same as above. Table 6.2 illustrates a confusion table for 4 Caltech Data-Sets. The discriminative method again outperforms its generative counterpart [FPZ03] despite using a much simpler underlying generative model.

There are several interesting points of note. First, the hybrid method works well even on dissimilar categories. In fact it performs significantly better than the corresponding generative ML method and slightly better than a more sophisticated ML method. Second: the classification results in Table 2 are comparable with the best current results in the literature. It is fair to say that the Caltech-4 data-set is easy (see Figures 6.5 and 6.6) and may not be the best data-set to use when comparing algorithms. Further results on more challenging data-sets, among them the 'People' and 'Bikes' Graz data-sets[2], were reported by us in [HWP05a] and the technique performed well in these cases as well. Three, Experiments conducted using only the Shape and Appearance Fisher Scores mostly indicate

[2]Available at http://www.emt.tugraz.at/ opelt/

that the combination of the two is more powerful than either in isolation. Interestingly, by comparing the shape-only and appearance-only performance one can see that the relative importance of these two terms varies with the category. In particular we notice that the Leopard data-set, which exhibits stereotyped appearance information but large articulations in shape, performs best when using only appearance information while airplanes, which exhibits fairly uniform shape information, yields good performance when using only shape information.

The hybrid approach uses an underlying generative constellation model to generate Fisher Scores which are in turn used for classification. This underlying model can be used to localize objects within an image by selecting the hypothesis with the highest likelihood (the same technique was demonstrated to localize objects by Fergus et. al in [FPZ03]). Figure 6.5 demonstrates the localization ability of our implementation of the constellation model on several different categories.

6.5.2 Background Classes

In this section we explore the effect of a particular "background" training set on learning. Consider the detection tasks described above (Figure 6.1) in which we build a hybrid classifier which detects whether or not an object is present in an image. In this setting, the algorithm must be trained using both a 'positive' or foreground training set and a negative or 'background' training set. In principle the background set should represent any images that does not contain the object of interest. This set of images has a very broad statistical variation which may not be well represented by a limited training set as used in our experiments. If the statistics of the background data-set are not appropriately chosen, we run the risk of over-fitting to those background images used for training.

In order to further explore this potential confound, we performed experiments using several common data-sets used as background images to determine how well background data-sets generalized to one another. We considered 3 standard sets of background images: (1) the Caltech background data-set used in [FPZ03] (Caltech1), (2) the Caltech background data-set used [LFP06] (Caltech2), (3) the Graz data-set used in [OFPA] (Graz). A random sample of images from the three sets is shown in Figure 6.6. We performed experiments by first generating a model for one foreground class. This generative model was then used to create Fisher scores for the foreground class and a particular background class and an SVM

classifier was trained using these scores. We *tested* the classifier on images from all three background classes.

Results for these experiments are summarized in Table 6.7. This table illustrates that the statistics of a particular background data-set can influence the ability of the classifier to generalize to new sets. These results should be seen as a caveat when using discriminative learning for detection tasks as the statistics of the background images play a crucial role in generalization, especially when relatively few background examples are used. Even if one has access to a large number of background images it is, in principle, difficult, to obtain a set of images which model the distribution of any arbitrary background. Furthermore, some discriminative methods, such as SVMs, are not readily amenable to training with unbalanced data-sets, making the choice of background images to use during training even more problematic.

6.6 Semi-Supervised Learning

In computer vision it is often easy to obtain unlabelled images while labelled images often require a significant investment of resources. In this section we explore how to leverage unlabelled and labelled images within our hybrid generative-discriminative framework described above. Using labelled and unlabelled data is often referred to as *semi-supervised learning* in the machine learning community. In particular we show how semi-supervised learning can be used to learn classifiers with far fewer training examples than the corresponding supervised framework. Figure 6.8 illustrates a schematic of the proposed semi-supervised learning algorithm. Some of these results were presented in [HWP05b].

Many interesting methods have been proposed for semi-supervised learning (see e.g. [See02] for an approach relevant to Fisher Kernels). We decided to implement a relatively simple idea that attempts to learn the kernel using the available unlabelled data. In the context of Fisher Kernels this boils down to learning a probabilistic model on the unlabelled data-set. We subsequently extract Fisher Scores based on this model, but evaluate it on the *labelled* data. These Fisher Scores are combined into a kernel matrix and provide input to the SVM. To see why this is a sensible approach we consider the task of classifying images of faces of two different people. The Fisher Scores represent the derivatives of the log-likelihood and hence the tendency of a particular sample to change the model. If the

Figure 6.6: Examples of background images: (top) Caltech 1 is a collection of indoor and outdoor digital photographs taken on the Caltech campus, (middle) Caltech 2 is a collection of images obtained from the Google image search engine by typing 'things' as a search string, (bottom) Graz is a collection of outdoor and indoor images, some of which focus on specific objects. There are noticeable differences in the image statistics from the different background classes.

Trained BG ⇒	Caltech1	Caltech2	Graz
Caltech1	**93**	86.5	82
Caltech2	83	**90.5**	78
Graz	83.5	88	**91**
Caltech1	**92**	82	83.5
Caltech2	83	**92.5**	87
Graz	85	85	**91.5**

Figure 6.7: Generalization of different background statistics: Top, Airplane vs. BG experiments. Bottom, Leopards vs. BG experiments. The top row indicates the background data-set trained with. The rows indicate the test set used. The columns indicate the performance of the algorithms. The bold scores indicate the performance on the test examples from the same background class which was trained on, these tend to be the highest performing test sets. We trained and tested with 100 images for each foreground and background category. Results were averaged over 2 experiments. The mHar detector was used.

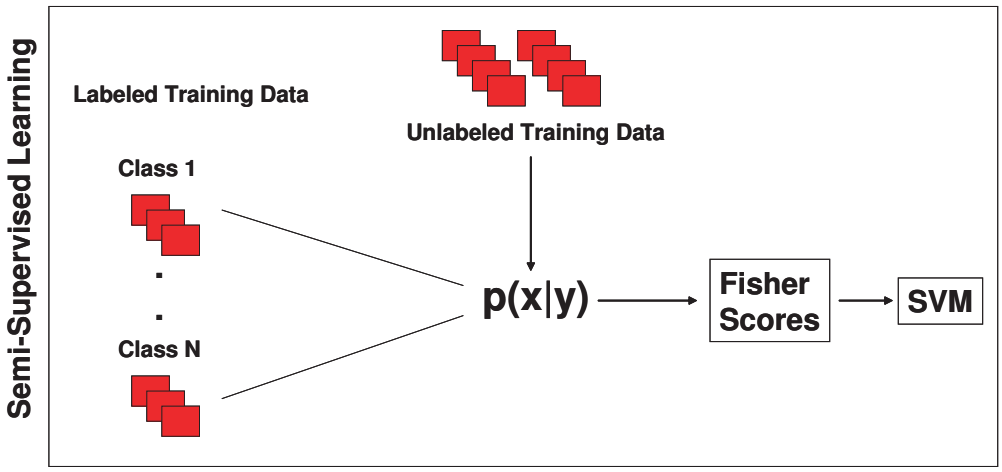

Figure 6.8: Schematic overview of the semi-supervised learning algorithm employed. Semi-supervised learning employs both labelled and unlabelled data. Note that only unlabelled data is used to create the generative model but that Fisher Scores are extracted from the generative model using labelled data.

underlying model is one of a completely unrelated class of objects (say leopards), then we expect any face image to roughly change the model in a similar fashion, so the Fisher Scores for different faces are expected to be similar. If however, the underlying model is an average face model, then we expect the changes to the model for person A and person B to be different, resulting in different Fisher Scores and hence small kernel entries. So clearly, a good kernel should be based on a probabilistic model for the class of objects we are trying to classify.

In the following sections will investigate a number of issues: 1) does the method we propose work at all, 2) what is the effect of using different sets of unlabelled data, 3) how does the performance depend on the number of unlabelled examples and 4) how does the performance depend on the number of labelled (training) examples.

6.6.1 Caltech Faces-Easy Categories

We employ a similar experimental paradigm as described above in Section 6.5. However, we consider situations when there is a wealth of unlabelled data available which will be used to construct the model from which we extract Fisher Scores. In order to be clear on our terminology we refer to the 'unlabelled training set' as the set of unlabelled data used to generate the classifier and the 'labelled training set' as the set of labelled data used to generate the classifier.

The Caltech Faces-Easy (see Figure 6.9) consists of about 400 images of human faces. It is composed of 20 or more photographs of 19 individuals, while the remaining 40 or so images contain fewer exemplars. Thus we can divide the entire face category into smaller categories corresponding to individuals.

A linear kernel was used in all experiments below. Using an exponential kernel is difficult due to the inability to accurately tune the scale parameter σ: there are not enough exemplars to perform cross-validation.

6.6.2 Results

We compared performance using various different sets of unlabelled training images to create the generative model, where each set of images was more or less related to the set of labelled training images. We considered the following sets of unlabelled training images: (1) Faces-Easy data-set. Note that our unlabelled training set contained images from all individuals

except those used for the labelled training set. (2) the Caltech 1 Background Images (see Figure 6.7 above), (3) the Leopards data-set, (4) Images of printed pages from a copy of Homer's 'Odyssey'. Examples of the features found for classes (1) and (4) are shown in Figure 6.9. A generative model trained on background will result in broad distributions, while generative models trained on specific classes will have a more distinct distribution reflecting the statistics of the unlabelled training images.

First a model was trained using images from the unlabelled training data-sets. Next, 2 individuals from the Faces-Easy set were selected at random to train the classifier. Fisher Scores were extracted from the model. We did not include any images used for training or testing in the unlabelled data-set. Median and 25th/75th quantiles were computed over 20 experiments by selecting to individuals at random.

Results are shown in Figure 6.10. Several interesting points can be made: (1) The nature of the unlabelled training data-set is critical: using a data-set unrelated to the classification problem at hand, e.g. the 'The Odyssey' or 'Leopards' data-sets in the context of face classification, results in the worst performance. Using a very general data-set, e.g. the Caltech 1 Background (BG1) data-set, results in better performance than using an unrelated data-set. This may be due to the broad distribution which results from training on the background data-set, which, although not as beneficial as using the images from the same class, is better than using a generative model from a completely different class. Finally, using a data-set which describes the distribution of the images involved in the classification problem well, e.g. using the remaining face images from "Easy-Faces" to classify the faces of two held-out individuals, results in the best performance. (2) Discrimination performance increases as we add more unlabelled training examples. This is particularly true for the faces data-set, in which we notice a large increase in performance from 10 to 100 prior examples. The same qualitative effect is observed for the BG1 data-set. We observed that with few training examples the BG1 unlabelled images creates overfitted models with small variances, resulting in comparable performance to using the 'Odyssey' or 'Leopards' data-sets. As more training examples are added, the model becomes more general, and its utility as a prior improves.

A reasonable measure for how appropriate a kernel is for a particular classification task

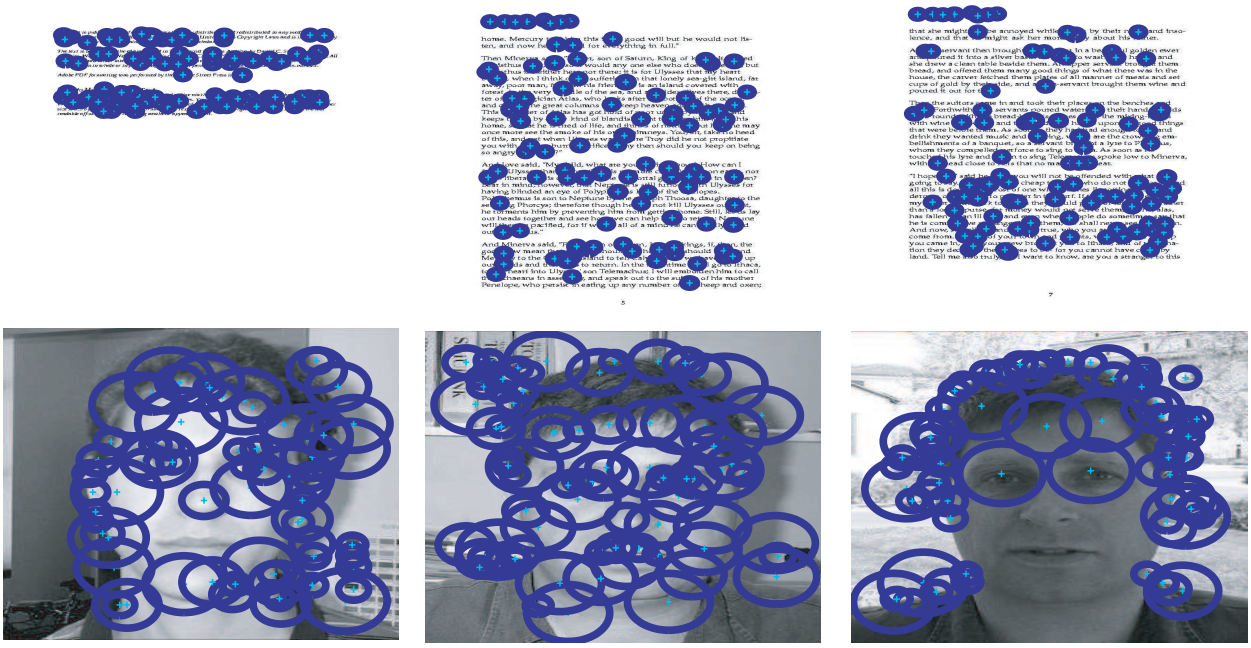

Figure 6.9: Features detected by the Kadir and Brady detector on image of (Top) the Odyssey text and (Bottom) the Faces-Easy data-set. Note that the Faces-Easy data-set is equivalent to the Faces data-set used in the experiments above except that the Faces-Easy data-set contains faces which are more cropped. Also note that the features found in the two sets of images have drastically different appearance statistics.

is the "kernel-alignment" proposed in [STC04],

$$A(\mathbf{y}\mathbf{y}^T, K) = \frac{\mathbf{y}^T K \mathbf{y}}{N\sqrt{\mathbf{tr}(K^T K)}} \tag{6.11}$$

where N is the number of training cases and $\mathbf{y}$ is a vector of $+1$ and -1 indicating the class of each data-case[3]. Figures 6.11 and 6.12 illustrate that training using more suitable sets of unlabelled data results in more appropriate kernels.

6.7 Combining Multiple Generative Models

In the previous section we showed how to leverage unlabelled data within our hybrid framework. In this section we show how the hybrid framework can be used to combine multiple generative models into a single classifier. Why is this useful? Consider that visual data

[3]Why did we not choose the perfect kernel according to this metric, namely $K = \mathbf{y}\mathbf{y}^T$? The reason is that this kernel would overfit on the training data and exhibit poor generalization performance. The alignment measure is therefore only useful provided we do not overfit, which we would not expect given the fact that we learn the kernel on a separate, unlabelled data-set.

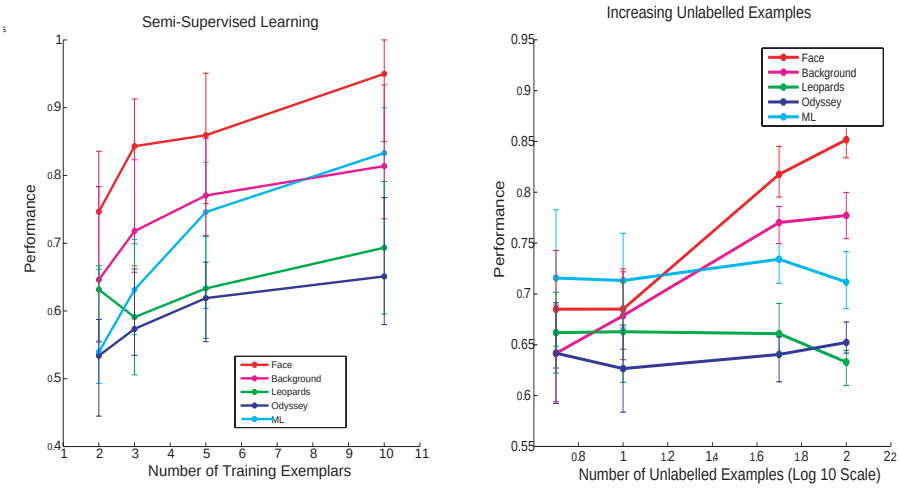

Figure 6.10: (Left) Performance on the 2-way faces classification problem as we vary the number of labelled training examples per class. Number of unlabelled examples was fixed at 100. (Right) Classification performance as a function of the number of unlabelled examples. 5 labelled training examples per class were used to train the SVM classifier. In both plots we show the median and 25th/75th quantiles computed over 20 experiments. Each line represents a different unlabelled data-set except for the cyan line which shows the maximum likelihood performance on the labelled training set only (i.e. it did not use any unlabelled data and classified by comparing the likelihood scores, the fluctuations in this line are due to using randomized training/test sets). Note that the nature of the unlabelled data-set has an important impact on the classification performance.

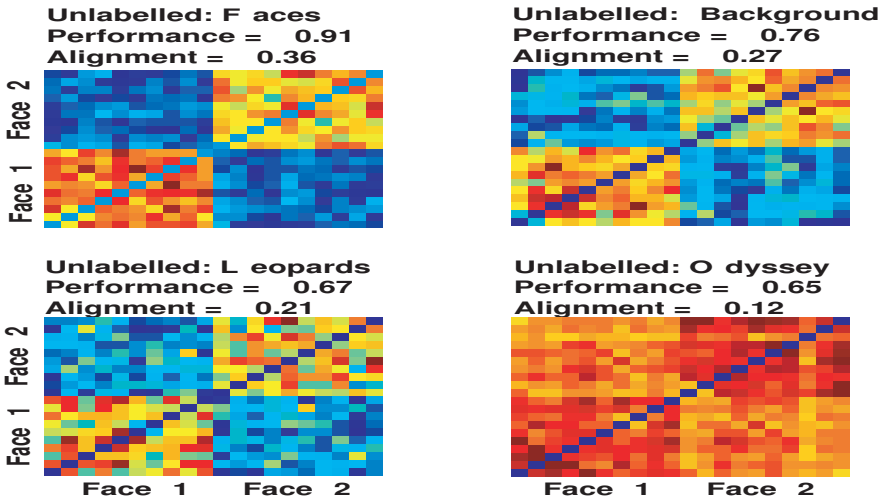

Figure 6.11: Kernel matrices computed using different sets of unlabelled training data. Each model underlying a kernel was trained on 200 unlabelled data-cases. A kernel was computed as $\boldsymbol{\phi}^T(x_i)\boldsymbol{\phi}(x_j)$ with normalized Fisher Scores and averaged over 20 experiments. Diagonal entries are zeroed out to improve resolution. When the Faces data-set was used as an unlabelled set we can easily discern a block-structure where the images in the same class are similar to each but dissimilar to the images of the other class (images in the same class correspond to first 10 entries and second 10 entries respectively and brighter colors indicate higher similarity between the data-points). Note that the block structure is not evident when a a dissimilar set of unlabelled training examples is used, i.e. the 'Odyssey' unlabelled data-set. Test performance and alignment values are also indicated.

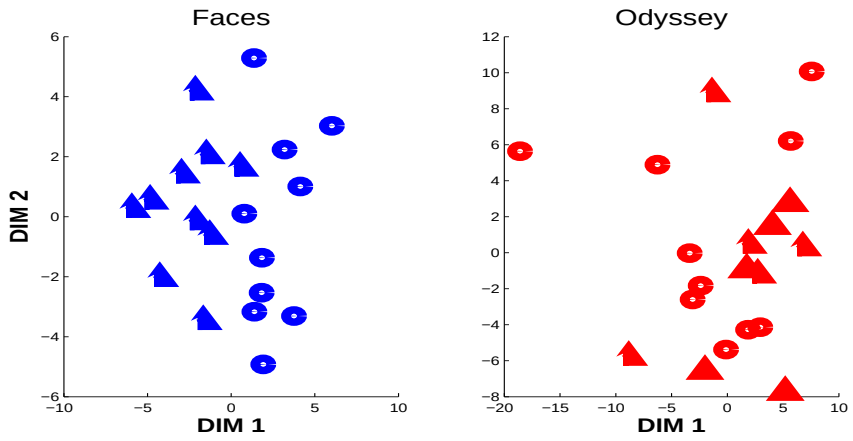

Figure 6.12: Visualization of Fisher Scores extracted from models created using different sets of unlabelled training data. Each plot was the result of learning with a different set of unlabelled training data, namely 'Faces' (Left) and 'Odyssey' (Right). The circle and triangles are two different individuals for which Fisher Scores were extracted from the generative model. Note that none of these images were in the set of unlabelled training data used to create the 'Faces' generative model. Distances in feature space were computed as $d_{ij} = ||\boldsymbol{\phi}(x_i) - \boldsymbol{\phi}(x_j)||$, where $\boldsymbol{\phi}$ is normalized and is Fisher Score. These were input to the MDS procedure which embeds the data in a 2-D Euclidean space while minimizing distance distortion. One can clearly see that the model learned on the Faces data (left) results in an embedding which is linearly separable (even in 2 dimensions), while the embedding obtained from the Odyssey data-set (right) is not linearly separable (different classes are represented by circles and triangles). This indicates that using 'Faces' as an unlabelled training set will result in a more robust classifier.

is heterogeneous in nature. Different 'front-ends' (feature detectors, feature descriptors) work best on different types of data. For instance, the techniques used for optical character recognition (OCR) vary greatly from those used to detect cars. Also, it is clear that some features should be described with 'brightness' templates, others with 'texture' descriptors, still others with parameterized edges or curves. It would be useful to remain initially agnostic as to which front-end is most useful and allow the learning algorithm to decide which to use.

In the previous section we learned the kernel on unlabelled training data under the assumption that very few labelled training data were available. However, it is also possible to tune the kernel based on the labelled training examples, provided the labelled data-set is sufficiently large. In this setting, one has to be careful not to "overfit" the kernel on the training examples and achieve poor generalization performance. A standard approach to determine regularization and kernel parameters is by cross-validation. Unfortunately, when the number of parameters is large this method becomes infeasible. An alternative approach that has been proposed in the literature is to optimize the kernel on approximations or bounds of the test-error. The approach we will follow here was described in [CVBM02] (see also [OW00] and [JH99]) which derive gradients for the span bound of the leave-one-out error. Other authors, e.g. [GHS05] offer an alternate solution to this problem.

Consider the choices we face when modelling a particular object category using the Constellation Model. Which front-end interest point detector should we use? How many parts should the model contain? Should one model the geometry and appearance of a part? Each of these choices leads to a different model and hence to a different set of Fisher Scores. Instead of choosing a particular type of model, one could argue to create multiple models, and incorporate information from all these models into a single Fisher Score by concatenating the Fisher Scores from each individual model. However, this is an unsatisfactory procedure because we do not know how to weight the different contributions, which may operate at different scales. Hence, a natural idea is to weight the Fisher Scores of each component model. Taking this one step further, one could decide to attach a different weight to each individual dimension of the entire feature vector, not just to each individual component model. This choice of parametrization is problem dependent: too many parameters may still lead to overfitting and may be too computationally expensive.

In this section we explore the potential of learning the weights for the various contribu-

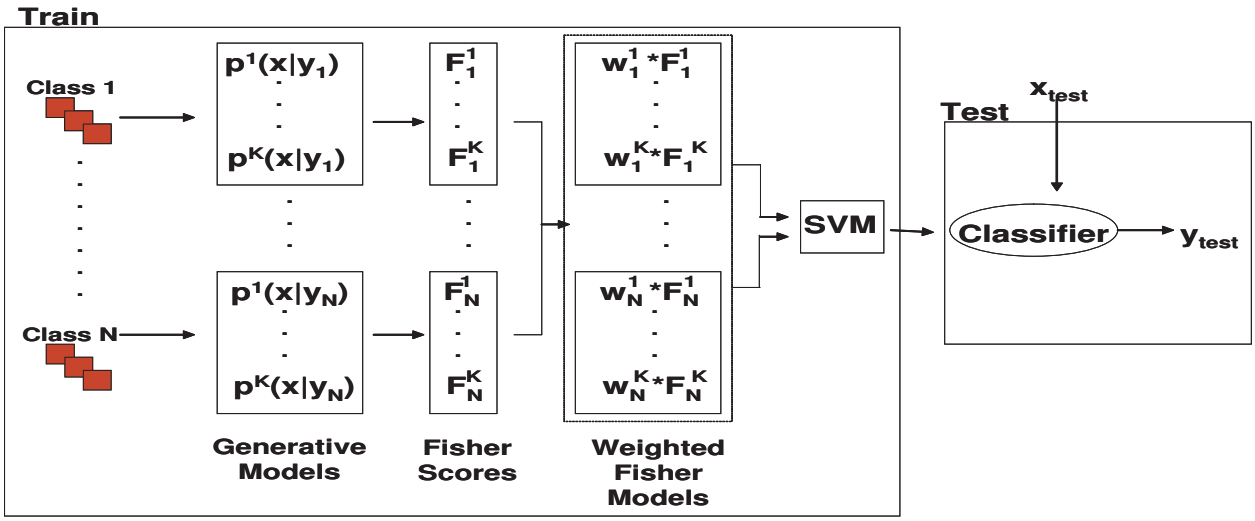

Figure 6.13: Overview of proposed visual category recognition algorithm which automatically weights the importance of different models.

tions to the kernel. This has lead to interesting insights; for instance that a certain type of detector is much better to build a face model than another type of detector. In some sense, by learning the optimal combination of kernels we are partially relieved from the task to pick the best components to model a certain object class. Instead we let the data decide which components are most important for the classification task. This information could subsequently be used to remove irrelevant components, thus improving the computational efficiency of the resulting classifier.

6.7.1 Leave-One-Out Span Bound

Our proposed method consists of three steps which are illustrated in Figure 6.13: 1) Train an ensemble of generative models separately for each class, 2) extract Fisher Scores to construct Fisher Kernels for all the models separately, 3) train an SVM classifier and "learn the kernel" by weighting the different kernels. Finding the weights is achieved by minimizing a bound on the leave-one-out error [CVBM02]. Next, we will provide details for these steps.

The general form of the kernel that we consider is,

$$K^{\mathrm{RBF}}(I_i, I_j) = \exp\left(-\frac{1}{2}\sum_{\mathcal{M}} w_{\mathcal{M}} ||\phi_{\mathcal{M}}(I_i) - \phi_{\mathcal{M}}(I_j)||^2\right) \tag{6.12}$$

where we have introduced separate weights $w_{\mathcal{M}}$ for each model. To tune the weights $w_{\mathcal{M}}$ and the regularization constant C in the SVM, we adopt the method proposed in [CVBM02]. In

this method a smoothed version of the the "span-bound" of the leave-one-out (LOO) error[4] is minimized,

$$T(\mathbf{w}, C) = \frac{1}{N}\sum_{n=1}^{N}\sigma_T(\alpha_n S_n^2 - 1) \qquad \text{with} \quad S_n^2 = \frac{1}{[(K_{SV} + \mathrm{Diag}(\eta/\boldsymbol{\alpha}))^{-1}]_{nn}} - \eta/\alpha_n \quad (6.13)$$

where $\sigma_T(x) = 1/(1+e^{-x/H})$ is the sigmoid function, H is the temperature (set to $H = 100$), η is a smoothing parameter (set to $\eta = 0.1$) and K_{SV} the kernel evaluated at the support vectors. The parameters $\boldsymbol{\alpha}$ are the dual weights in a 2-norm soft margin SVM. The 2-norm SVM is convenient because the regularization parameter C can be considered as a parameter of the kernel function, alongside the weights $w_{\mathcal{M}}$. (hence the notation $\boldsymbol{\theta} = (\{w_{\mathcal{M}}\}, C)$ in the following). We invite the reader to explore [CVBM02] for more details and a more complete explanation of this procedure.

The chief advantage of this smoothed span bound is that it can be efficiently minimized using gradient descent. The gradients can be written as,

$$\frac{\mathrm{d}T(\boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} = \frac{\partial T(\boldsymbol{\theta})}{\partial K_{SV}}\frac{\partial K_{SV}}{\partial \boldsymbol{\theta}} + \frac{\partial T(\boldsymbol{\theta})}{\partial \boldsymbol{\alpha}}\frac{\partial \boldsymbol{\alpha}}{\partial \boldsymbol{\theta}} \quad (6.14)$$

While the first partial derivative is straightforward, the second requires some more thought because the dual weights $\boldsymbol{\alpha}$ are the solution of the 2-norm SVM. For more details on how to compute it we refer to [CVBM02]. Discontinuities in this derivative are expected when data-cases jump in or out of the support vector set. However, the gradient descent with line search procedure works well in practice. Note that for each gradient step we need to solve a QP for the SVM. However, this procedure is much more efficient than cross-validation which would need to check a number of grid-points exponential in the number of parameters, which is at least a dozen in our experiments.

6.8 Experiments with Combinations of Kernels

We tested our system which weights different models on numerous object categories. Each image within a category is associated with a class label, however the objects are not segmented within an image. We used some of the classes from previous experiments and and collected an additional set of classes consisting of 200 images of 3 different faces taken

[4] We show the case without bias term for simplicity, but bias was included in our experiments.

	Unweighted	Weighted	10%	20%	−10%	Others
leopards vs. BG	91.3	94.3	89	90	62	88 [Fer05]
P1 vs. P3	86	89.8	82.2	81.1	67	–

Table 6.3: The effect of weighting and removing features on performance. Unweighted: performance using XVal on a single weight for all dimensions and a slack parameter. Weighted: performance after minimizing the LOO-span bound on the weights of each model. 10%/20%: performance using only the top 10% and 20% of the relevant dimensions. −10%: performance using the worst 10%. Others: the performance of previous constellation model algorithms. Note that [FPZ03] uses 6-part models including occlusion and typically $2-4$ times more training examples.

against varying backgrounds and lighting conditions. Examples of the face images are shown in Figure 6.1. For the 2-part models up to 100 detected interest points were used, and for the 4-part models up to 20 interest points. Typically 100 training images and up to 250 testing images were used.

Training a classifier proceeds as follows: First, learn a suite of generative models for each class using the training data only. In case we classify against background we only train models on the foreground class. Next, extract Fisher Scores from all models and train weights $w_{\mathcal{M}}$ by minimizing the span bound. The weighted scores are used to construct a single kernel K_{RBF} which is used to train a 2-norm SVM, still only using training data. The kernel and dual weights $\boldsymbol{\alpha}$ for the support vectors are then used in the usual way to predict the class label of test cases.

6.8.1 Feature Selection

The generative models contain numerous parameters from which Fisher Scores are extracted and it is unclear, in general, which components of the model(s) are important for classification tasks. Figure 6.14 illustrates the resulting weights after minimizing the LOO bound. We observe that the relative weight of features vary for each classification task, thereby giving us a deeper understanding of the importance of these model components for a particular task. Table 6.3 illustrates the effects on performance as subsets of 'good' and 'bad' features are removed. We observe that the weights obtained from the LOO procedure are good indicators of generalization ability of a particular feature. Parameter selection for the "unweighted" procedure was done using 10-fold cross-validation (XVal) by varying both C and a single weight for all the models W in log steps from $[-9:3:9]$.

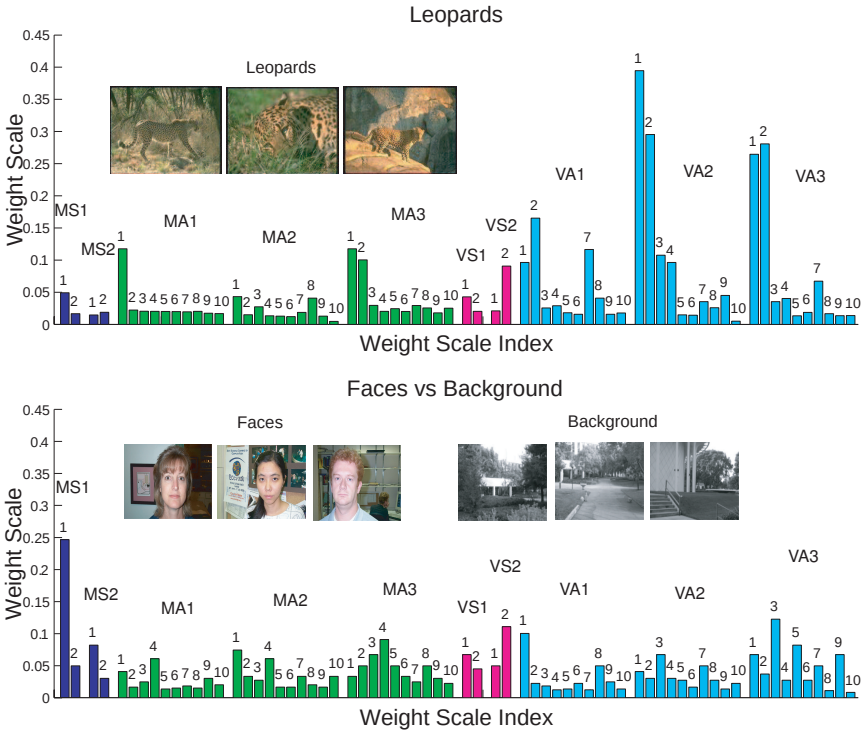

Figure 6.14: Determining which model parameters are most useful for a particular classification task. Higher weights indicate more important features. Fisher Scores are extracted from a 3 part diagonal covariance model with 10 dimensional appearance model of the foreground class using the KB detector. 100 training examples were used. Position is conditioned on location of first model part so the shape model is only optimized over 2 parts. M/V indicates mean/variance, S/A indicates Shape/Appearance and 1/2/3 indicates the part within a model. Each group has 2 or 10 bars corresponding the number of dimensions. E.g. "VA2" indicates a group of 10 variance parameters for the appearance model of part 2. (Top) Leopards vs. Background classification task. Early coefficients in the appearance model are the most useful dimensions for classification. (Bottom) Faces vs. Background classification task. The Shape model appears to be more useful for classification. The consistent detection of facial features by the KB detector makes the shape model relatively more important.

	Unweighted	Weighted	1	2	−1	Others
motorcycles vs. BG	92.6	92.6	77.2	89.6	72	96.7 [Fer05]
leopards vs. BG	93.4	94.8	90.5	93.7	69.1	88 [Fer05]
airplanes vs. BG	89.8	91.4	80.6	87.2	65.4	93.3 [Fer05]
P2 vs. P3	83.3	92.4	92.7	93.0	67.9	–

Table 6.4: The effect of weighting and removing models on performance. Unweighted/Weighted/Others: same as Table 6.3. 1/2: the performance using only the best model/performance using the first and second best model. −1: performance using the worst model. 100 training images were used.

6.8.2 Model Selection

We have argued above that selecting for individual features becomes computationally difficult and subject to over-fitting when the number of LOO optimized parameters is too high. We address this issue by assuming each model has the same weight for each of its extracted fisher scores (i.e. the weights are tied across features within a model). We apply the LOO Span Bound described above to appropriately weight the models for maximum generalization performance. We generate separate models based on: (1) Shape and Appearance, (2) 2/4 model parts, (3) 4 different interest point detectors. This results in 16 models for each class. Figure 6.16 shows the evolution of the weights and the corresponding change in test error on a typical classification task.

Figure 6.15 illustrates that the LOO procedure selects different combinations of models depending on the particular classification task at hand. These optimized combinations of models yield an increase in classification performance over the XVal procedure on the unweighted models (see Table 6.4 and Table 6.5).

In addition, we study the effect of training on a subset of the models based on their optimized weights from the LOO procedure (see Figure 6.16 Left, and Table 6.4). We notice that training the Xval procedure with only the best models results in good performance, while training using only the models with poor weights results in low classification performance. This procedure can be used to select the best models for a particular classification task, allowing for computational savings during detection.

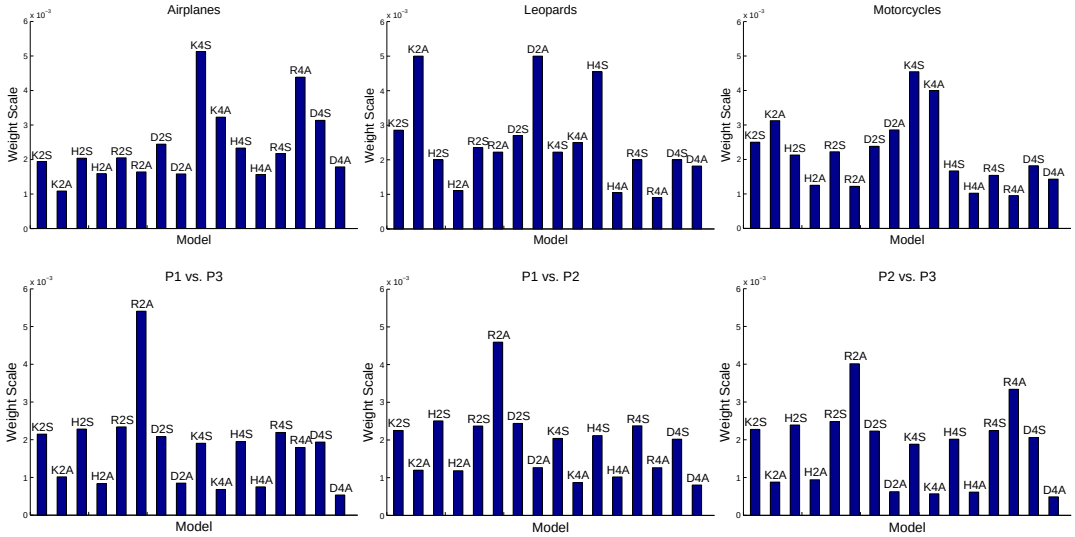

Figure 6.15: The effects of weighting models on different classification tasks. K/H/R/D indicate KB/mHes/mHar/DoG detectors. 2/4 indicates Two/Four part models. S/A indicates Shape/Appearance models. E.g. "K2A" corresponds to a 2-part appearance model using the KB detector. Top Row. (Left) Airplanes vs. Background. (Center) Leopards vs. Background. (Right) Motorcycles vs. Background. Different combinations of models are selected for a particular classification task. Bottom Row. Performance on People Faces classification tasks. Notice that models created using mHar seem to be the best predictors of generalization ability. Performance using weighted models, from left to right, of 91.45, 92, and 92.42 respectively.

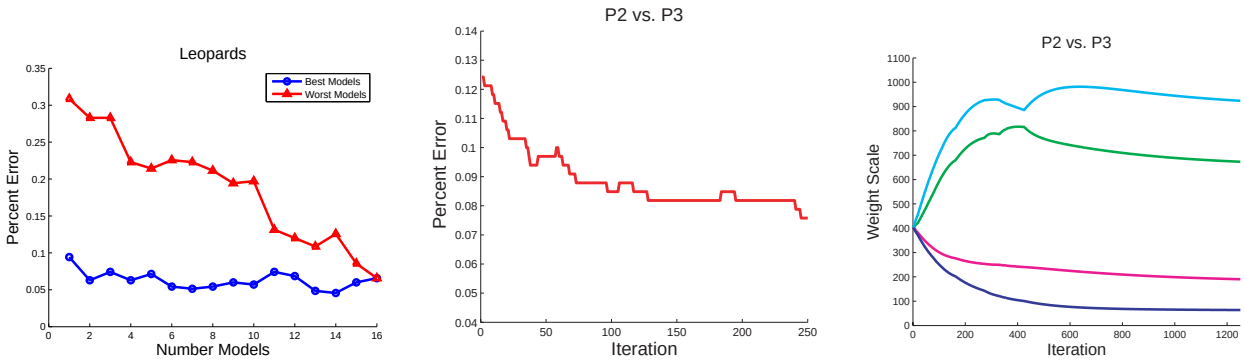

Figure 6.16: (Left) Performance on the Leopards vs. Background classification task as a function of the number of models used. Circles indicates when the best models are used, i.e. the index '2' would indicate that the two best models were selected. Triangles indicate that the worst models were used, i.e. index '2' indicates that the worst two models were used. Models were weighted equally during classification. Each point generated by conducting Xval over the weights and slack parameter using only the specified models. (Center) Typical example of test error change during the LOO procedure using initialization parameters from Xval as a function of the number of iterations. (Right) Typical evolution of model weights for a combination of 4 models as a function of the number of iterations. Plot of weight scales where the weight scale is defined as as function of the model weight w: $\sigma^2 = 1/w_{\mathcal{M}}$. Note that certain models become weighted more or less heavily emphasized as we progress in optimization.

	KB	mHar	mHes	DoG	Unweighted	Weighted
P1 vs P2	76.6	82.8	72.3	61	88.2	92.8
P1 vs P3	64.3	85.2	73.6	69.1	89.7	93.6
P2 vs P3	74.6	82.4	81.1	64.7	90.1	92.8

Table 6.5: Performance on discriminating between faces in the People Face data-set. KB/mHar/mHes/DoG: performance using Appearance/Shape models created using only the the given detector. Unweighted: performance using models of all detectors with a single weight. Weighted: performance after LOO optimization of the weights for the models. Notice the relatively poor performance using only the KB or mHes on these classification tasks. 50 training images used, 2-part, 20 PCA coefficient models.

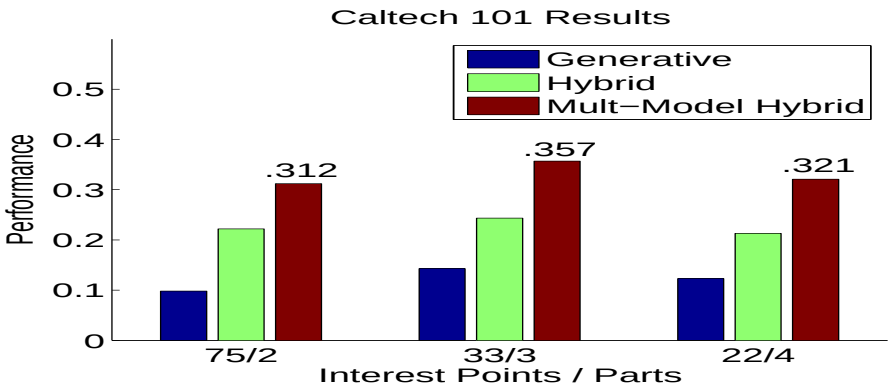

Figure 6.17: Performance on the Caltech 101 object category data-set when different numbers of parts and interest points are used. Each set of columns indicate a different number of parts and interest points used, i.e. 33/3 indicates that a maximum of 33 interest points were detected in each image and that a 3 part model was used. Performance is measured by first calculating the percent of correctly classified points in each class and then taking the average over all classes (this corresponds to the average of the main diagonal of the confusion table). First column in each set: pure generative approach using interest points from the KB interest point detector. I.e. a generative model is created for all 101 categories and test examples are assigned to the generative model with the highest posterior probability. Second column: hybrid approach where Fisher Scores were extracted from a generative model made using the KB interest point detector. See text for more details. Third column: hybrid approach when three (KB, mHar, mHess) different interest point detectors are used and a generative model is created for each detector. Fisher Scores are concatenated into a single vector. Fei Fei et al. [LFP06] managed 16% using a constellation model with 3 parts and integrating prior information into their model. For a similar 3-part constellation model with no prior information we achieve 14.3% using only the generative models, 26.1% using our hybrid approach, and 35.7% using our multiple model hybrid approach. 15 training examples were used for all experiments.

6.8.3 Integrating Unlabelled Data and Kernel Combination – The Caltech 101

We next performed an experiment using our hybrid model which integrated both the use of unlabelled data in a semi-supervised learning paradigm as well as combining multiple models into a single classifier.

The Caltech 101 objet category data-set[5] consists of 101 object categories with varying numbers of examples in each category (from about 30 to over 1000). The challenge of this data-set is to learn representations for many different object classes using a limited number of training examples. The variability of this data-set is mostly evident between different categories of objects rather than within a single category. Objects within a particular category are often somewhat homogenous in both appearance and pose.

[5]Available at http://www.vision.caltech.edu/html-files/archive.html

For our experiments we used the following experimental paradigm. First we created a broad underlying generative model using 5 randomly selected training images from all 101 classes. In the parlance of Section 6.6, this set of images corresponds to our set of unlabelled trained examples. Generative models from this set of unlabelled training examples were created using interest points detected from three different detectors, KB, mHess, mHar. Fisher Scores were then extracted from the generative models for all classes within the Caltech 101. Fisher Scores from each generative model were concatenated to form a single, long vector to be used in the SVM classifier. To train our SVM classifier we used 15 training examples and up to 30 test examples. We used the same cross-validation procedure to find the hyper-parameters as described in the multi-class experiments above.

Figure 6.17 illustrates our results. These results show that utilizing the hybrid approach yields substantial improvements in classification performance over the generative approach and that additional performance gains are realized when generative models created using different underlying feature detectors are combined to form a single classifier. These experiments illustrate the utility in using a semi-supervised approach as well as combining kernels in improving classification performance.

6.8.4 Integrating Unlabelled Data and Kernel Combination – The Graz Data-Sets

We tested the performance of combining multiple kernels using the more challenging Graz data-sets[6], in particular the 'persons' and 'bikes' sets (Figure 6.18 shows examples from these sets). It is imperative to have large numbers of features when learning on these sets due to the large variability of the objects within the images. We first experimented with combining multiple generative models, with each model containing a different number of parts (2 and 3 parts). Both models were trained on the same data. Fisher scores were extracted from the foreground generative models on a training set of data for both the foreground and background classes on both 2 and 3 part models and combined into one large vector for SVM training. Test scores were extracted in the same way. Table 6.6 illustrates results combining simple 2 and 3-part constellation models. The performance training an SVM classifier on Fisher scores for each individual model was less than the combined performance. We anticipate using more features would result in higher performance on the

[6]These images can be obtained from http://www.emt.tugraz.at/~pinz/data/

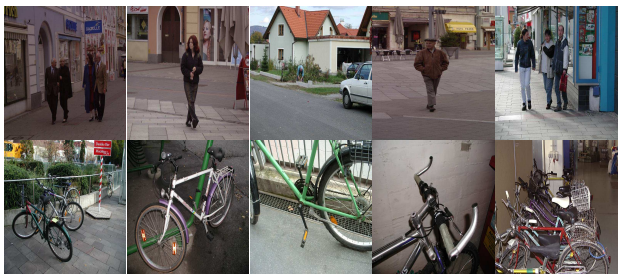

Figure 6.18: Example images from the Graz persons (top) and bikes (bottom) data-sets. Note the large variations in pose, lighting, occlusion, and scale.

Gen. Model ⇒	Comb	2-Part	3-Part	Prev
persons	**78.5**	75.2	76.4	80.8 [OFPA]
bikes	**75.3**	74.5	74.9	86.5 [OFPA]

Table 6.6: Effects of combining multiple generative models using the Fisher kernel. Results shown for the Graz 'persons' and 'bikes' sets. 200 images used for training. Note that the combined models outperform individual models. The 2-part and 3-part models used a maximum of 100 and 30 interest points respectively.

Graz sets and this is an active research area.

In addition we combined models trained using different interest point detectors. Each generative model was trained using different interest points detected on the same set of images. Table 6.7 shows the results on the same data-sets. There are many more possible combinations of models to explore, and combining models using different kernels is an exciting avenue of future research.

6.9 Discussion and Conclusions

We have explored a method for obtaining discriminative classifiers from generative models of visual categories. It works in two steps: first train generative models, then from those generative models calculate Fisher Kernels and use them for classification. This method achieves the best of both the generative and discriminative worlds: for generative approaches it is robust to occlusion and clutter, it may be trained from few examples, it benefits from prior

Gen. Model ⇒	Comb	KB	DoG	Prev
persons	**73.1**	65.3	77.5	80.8 [OFPA]
bikes	**79.0**	73.3	76.5	86.5 [OFPA]

Table 6.7: Effects of combining generative models trained using different feature detectors. We used a polynomial degree 2 kernel for experiments.

knowledge. Additionally, the generative part of the model may be trained incrementally. Its discriminative nature results in superior classification performance.

Our experiments in Section 6.5 show that the performance of our hybrid approach is not inferior to that of a traditional generative constellation model. Rather, performance is significantly better there and is in line with the current results in the literature. The advantage of the hybrid approach is particularly evident when the categories to be classified are very similar, such as the faces of different people. In addition, we controlled for possible overfitting to background statistics and found that this may be an issue.

Sections 6.6 show that our hybrid architecture lends itself readily to incorporating unlabelled examples. This is achieved by training generative models on large sets of unlabelled pictures which may contain relevant information. This process provides considerable performance improvement when learning specific categories (faces in our experiments) with very few training examples. As one would expect, we find that these results vary with the statistics of the images used to construct the generative model. Figure 6.10 shows that learning the statistics unrelated categories such as Text or Leopards will not help in distinguishing between Faces, while the most useful generative model has the same statistics as the Faces data-set.

In Section 6.7 we show that multiple models may be readily combined within our hybrid method. Our experiments in Section 6.8 suggest that the system is able to determine automatically which models provide the most valuable information for a given classification task. This indicates that a higher level of automation has been achieved: we do not need to ask an expert vision engineer to craft the best recognition strategy for a given task; rather, we can let our hybrid system self-tune to use whatever information is most valuable. Finally, we illustrate that combining both our semi-supervised learning approach and the ability to combine multiple models yields strong performance on the Caltech 101 object category data-set and that this performance is far superior to that of the corresponding generative approach.

Bibliography

[Ash92] F.G. Ashby. *Multidimensional models of Perception and Cognition.* Erlbaum, Hillsdale, NJ., 1992.

[BBE+04] Tamara L. Berg, Alexander C. Berg, Jaety Edwards, Michael Maire, Ryan White, Yee Whye Teh, Erik G. Learned-Miller, and David A. Forsyth. Names and faces in the news. In *Computer Vision and Pattern Recognition (CVPR) (2)*, pages 848–854, 2004.

[BCB98] B. Bartell, G. Cottrell, and R. Belew. Optimizing similarity using multi-query relevane feedback. In *American Society for Information Science*, volume 49, pages 742–761, 1998.

[BHK97] Peter N. Belhumeur, João P. Hespanha, and David J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[Bou03] G. Bouchard. The trade-off between generative and discriminative classifiers. Technical report, INRIA, 2003.

[BP96] Michael Burl and Pietro Perona. Recognition of planar object classes. *Computer Vision and Pattern Recognition (CVPR)*, page 223, 1996.

[CHL06] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition (CVPR)*, 2006.

[Cro84] J. L. Crowley. A representation for shape based on peaks and ridges in the difference of low pass transform. *Pattern Recognition and Machine Intelligence (PAMI)*, 1984.

[CVBM02] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46:131–159, 2002.

[DLR76] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. In *JRSS B*, volume 39, pages 1–38, 1976.

[DS04] Gyuri Dorko and Cordelia Schmid. Object class recognition using discriminative local features. *Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2004.

[ESZ06] M. Everingham, J. Sivic, and A. Zisserman. Hello! my name is... buffy – automatic naming of characters in tv video. In *Proceedings of the British Machine Vision Conference*, 2006.

[EZ06] Mark Everingham and Andrew Zisserman. Regression and classification approaches to eye localization in face images. In *FG*, pages 441–448, 2006.

[Fer05] R. Fergus. *Visual Object Recocogntiion*. PhD thesis, Department of Engineering Science, University of Oxford, UK, 2005.

[FJ04] H.-J. Zhang B. Zhang F. Jing, M. Li. Entropy-based active learning with support vector machines for content-based image retrieval. *IEEE International Conference on Multimedia and Expo*, 2004.

[FPZ03] Robert Fergus, Pietro Perona, and Andrew Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Computer Vision and Pattern Recognition (CVPR)*, pages 264–271, 2003.

[FPZ04] Robert Fergus, Pietro Perona, and Andrew Zisserman. A visual category filter for google images. In *European Conference on Computer Vision (ECCV)*, pages 242–256, 2004.

[GHP07] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report UCB/CSD-04-1366, California Institute of Technology, 2007.

[GHS05] Carl Gold, Alex Holub, and Peter Sollich. Bayesian approach to feature selection and parameter tuning for support vector machine classifiers. *Neural Networks*, 18(5-6):693–701, 2005.

[GHW06] Peter V. Gehler, Alex Holub, and Max Welling. The rate adapting poisson model for information retrieval and object recognition. In *International Conference on Machine Learning (ICML)*, pages 337–344, 2006.

[HEZP05] Alex Holub, Mark Everingham, Andrew Zisserman, and Pietro Perona. Combining principal component techniques and psychological spaces to find perceptually similar faces. In *Vision Science Society (Abstract)*, 2005.

[HhLP07] Alex Holub, Yun hseuh Liu, and Pietro Perona. On constructing facial similarity maps. In *Computer Vision and Pattern Recognition (CVPR)*, 2007.

[Hin89] V. Hinsz. Facial resemblance in engaged and married couples. *Journal of Social and Personal Relationships*, 6:223–229, 1989.

[HP05] Alex Holub and Pietro Perona. A discriminative framework for modelling object classes. In *Computer Vision and Pattern Recognition (CVPR)*, pages 664–671, 2005.

[HWP05a] Alex Holub, Max Welling, and Pietro Perona. Combining generative models and fisher kernels for object recognition. In *International Conference on Computer Vision (ICCV)*, pages 136–143, 2005.

[HWP05b] Alex Holub, Max Welling, and Pietro Perona. Exploiting unlabelled data for hybrid object classification. In *NIPS 2005 Workshop in Inter-Class Transfer*, 2005.

[HWP07] Alex Holub, Max Welling, and Pietro Perona. Hybrid generative-discriminative object recognition. *International Journal of Computer Vision (IJCV)*, 2007.

[Int] Intel. Opencv computer vision library. In *http://www.intel.com/technology/computing/opencv/*.

[JDH99] T. Jaakkola, M. Diekhans, and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in neural information processing systems (NIPS)*, volume 11, pages 487–493, 1999.

[Jeb01] Tony Jebara. PhD thesis, MIT, Cambdridge, MA, 2001.

[JH99] T. Jaakkola and D. Haussler. Probabilistic kernel regression models. In *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics*, 1999.

[KB01] Timor Kadir and Michael Brady. Saliency, scale and image description. *International Journal of Computer Vision (IJCV)*, 45(2):83–105, 2001.

[KH03] Sanjiv Kumar and Martial Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *International Conference on Computer Vision (ICCV)*, pages 1150–1159, 2003.

[Kru64] J.B. Kruskal. In *Psychometrika*, pages 115–129, 1964.

[LFP06] Fei-Fei Li, Robert Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machince Intelligence*, 28(4):594–611, 2006.

[Low04] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 2004.

[LS04] B. Leibe and B Schiele. Scale-invariant object categorization using a scale-adaptive mean-shift search. *DAGM-Symposium*, pages 145–153, 2004.

[LSP06] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories, 2006. Computer Vision and Pattern Recognition (CVPR).

[MN98] M. McCallum and K. Nigam. Employing em in pool-based active learning for text classification. 1998. International Conference on Machine Learning (ICML).

[NGOL06] J. Neiworth, A. Gleichman, A. Olinich, and K. Lamp. Global and local processing in adult humans, 5-year olds, and nw monkeys. *Journal of Comparative Psychology*, 120, 2006.

[NJ02] A. Ng and M. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems (NIPS)*, volume 12, 2002.

[OFPA] A. Opelt, M. Fusseneger, A. Pinz, and P. Auer. Generic object recognition with boosting. *Pattern Analysis and Machine Intelligence (PAMI).*

[OW00] M. Opper and O. Winther. Gaussian processes and svm: Mean field and leave-one-out. In *Advances in Large Margin Classifiers*, pages 311–326. MIT Press, 2000.

[Pla00] J. Platt. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. *Advances in Large Margin Classifiers, MIT Press*, 2000.

[RBK98] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 20(1):23–38, 1998.

[RH97] Y.D. Rubinstein and T. Hastie. Discriminative vs. informative learning. In *AAAI*, 1997.

[Rho88] G. Rhodes. Looking at faces: first-order and second-order features as determinants of facial appearance. *Perception*, 17:43–63, 1988.

[Sam69] J.W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Compututation*, C-18:401–409, 1969.

[SBB03] T. Sim, S. Baker, and M. Bsat. The cmu pose, illumination, and expression database. *Pattern Analysis and Machine Intelligence (PAMI)*, 25(12):1615–1618, 2003.

[Sch04] H. Schneiderman. Learning a restricted bayesian network for object detection. *Computer Vision and Pattern Recognition (CVPR)*, pages 639–646, 2004.

[Sea06] P. Sinha and et al. Face recognition by humans: nineteen results all computer vision researchers should know about. In *Proceedings of the IEEE*, Vol 94 N2, 2006.

[See02] M. Seeger. Covariance kernels from bayesian generative models. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14, pages 905–912, 2002.

[Sin02] P. Sinha. Identifying perceptually significant features for recognizing faces. In *Proc. SPIE*, volume 4662, pages 12–21, 2002.

[SK00] Henry Schneiderman and Takeo Kanade. A statistical method for 3d object detection applied to faces and cars. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1746–1759, 2000.

[SOS] H. Seung, M. Opper, and H. Sompolinsky. Query by committee. Proceedings of the Fiftth Workshop on Computational Learning Theory.

[SS02] B. Schoelkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.

[SSZ04] J. Sivic, F. Schaffalitzky, and A. Zisserman. Object level grouping for video shots. In *Proceedings of the 8th European Conference on Computer Vision (ECCV)*, May 2004.

[STC04] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[TAKM03] Koji Tsuda, Shotaro Akaho, Motoaki Kawanabe, and Klaus-Robert Mller. Asymptotic properties of the fisher kernel. In *citeseer.ist.psu.edu/tsuda03asymptotic.html*, 2003.

[TC01] S. Tong and E. Chang. Support vector machine active learning for image retrieval, 2001. Proceedings of the ninth ACM international conference on Multimedia.

[TK00] S. Tong and D. Koller. Support vector machine active learning with applications to text classification, 2000. International Conference on Machine Learning (ICML).

[TMF04] Antonio B. Torralba, Kevin P. Murphy, and William T. Freeman. Sharing features: Efficient boosting procedures for multiclass object detection. In *Computer Vision and Pattern Recognition (CVPR)*, pages 762–769, 2004.

[TP91] M. Turk and A. Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition (CVPR)*, 1991.

[UVNS02] S. Ullman, M. Vidal-Naquet, and E. Sali. Visual features of intermediate complexity and their use in classification. *Nature Neuroscience*, pages 682–687, 2002.

[Vap98] V. Vapnik. *Statistical Learning Theory.* Wiley-Interscience, 1998.

[VHM04] N. Vasconcelos, P. Ho, and P. Moreno. The kullback-leibler kernel as a framework for discriminant and localized representations for visual recognition. *European Conference on Computer Vision (ECCV)*, pages 430–441, 2004.

[VJ01] Paul A. Viola and Michael J. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition (CVPR)*, pages 511–518, 2001.

[WCG03] C. Wallraven, B. Caputo, and A.B.A. Graf. Recognition with local features: the kernel recipe. *International Conference on Computer Vision (ICCV)*, pages 257–264, 2003.

[Web00] Markus Weber. *Unsupervised Learning of Models for Object Recognition.* PhD thesis in Computation and Neural Systems, California Institute of Technology, Pasadena, CA, 2000.

[WWP00] M. Weber., M. Welling., and P. Perona. Towards automatic discovery of object categories. *Computer Vision and Pattern Recognition (CVPR)*, page 2101, 2000.

[ZH03] X. Zhou and T. Huang. Relevance feedback in image retrieval: A comprehensive review. *Multimedia Systems*, 2003.