Semi-Supervised Training of Models for Appearance-Based Statistical Object Detection Methods

Charles Joseph Rosenberg

CMU-CS-04-150

May 2004

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was supported in part by a fellowship from the Eastman Kodak Company.

Keywords: Object Detection, Semi-Supervised Learning, Computer Vision, Machine Learning, Weakly Labeled Data.

Abstract

Appearance-based object detection systems using statistical models have proven quite successful. They can reliably detect textured, rigid objects in a variety of poses, lighting conditions and scales. However, the construction of these systems is time-consuming and difficult because a large number of training examples must be collected and manually labeled in order to capture variations in object appearance. Typically, this requires indicating which regions of the image correspond to the object to be detected, and which belong to background clutter, as well as marking key landmark locations on the object. The goal of this work is to pursue and evaluate approaches which reduce the amount of fully labeled examples needed, by training these models in a semi-supervised manner. To this end, we develop approaches based on Expectation-Maximization and self-training that utilize a small number of fully labeled training examples in combination with a set of "weakly labeled" examples. This is advantageous in that weakly labeled data are inherently less costly to generate, since the label information is specified in an uncertain or incomplete fashion. For example, a weakly labeled image might be labeled as containing the training object, with the object location and scale left unspecified. In this work we analyze the performance of the techniques developed through a comprehensive empirical investigation. We find that supplementing a small fully labeled training set with weakly labeled data in the training process reliably improves detector performance for a variety of detection approaches. The outcome is the identification of successful approaches and key issues that are central to achieving good performance in the semi-supervised training of object detection systems.

Acknowledgments

First and foremost I would like to thank my co-advisor, Martial Hebert for all of his patience and support over the years, and for really giving me the push forward I needed to complete the work in this thesis and finish my degree. I also would like to thank Sebastian Thrun, my primary advisor during my early years at Carnegie Mellon and my co-advisor for this thesis work, for giving me the freedom to explore my own ideas and research directions.

I would like to thank the members of my committee for their great discussions and insights involving this work: Henry Schneiderman, Tom Minka and Avrim Blum. Especially Henry Schneiderman for providing his detector code and training data for this work, and for all of the time he spent helping me to understand the system so I could adapt it to my needs.

I would of course like to thank all of my friends through the years at Carnegie Mellon: Kevin Watkins, Dennis Strelow, Aleks Nanevski, Derek Dreyer, Rose Hoberman, Franklin Chen, Laurie Hiyakumoto, Mike Vande Weghe, Illah Nourbakhsh, Marti Louw, Nicolas Vandapel, Sanjiv Kumar, Daniel Huber, Diane Stidle, Nathaniel Daw, Francisco Pereira, Mark Fuhs, Goksel Dedeoglu, Catherine Copetas, Sharon Burks. I also want to thank Sanjiv Kumar for collecting the images used in my “color model” and “filter model” experiments. And Saneh Nasserbakht for her friendship and support during the early days of my thesis. And also Monica Brucker for providing comments on the final version.

I want to thank my parents, Freda and Stanley for the wonderfully supportive and intellectually stimulating environment I grew up in. And my sister Harriet for being a great friend and foil.

I would also like to thank Larry Ray from Kodak and the Eastman Kodak Company for providing me with funding during my final years at Carnegie Mellon.

Contents

List of Figures

List of Tables

Thesis Statement

The semi-supervised training of appearance-based object detection systems using a combination of a limited quantity of fully labeled data and “weakly labeled” data can achieve detection performance equivalent to systems trained with a larger quantity of fully labeled data alone. This can simplify the collection and preparation of training data for such systems and allow them to be used in a much wider variety of settings.

1 Introduction

1.1 Overview

Object detection systems based on statistical models of object appearance have been quite successful in recent years [Burl 95, Rowley 98a, Schneiderman 98,00a,00b,03,04, Schiele 00, Viola 01]. Because these systems directly model an object’s appearance in an image, a large amount of labeled training data is needed to provide good coverage over the space of possible appearance variations. However, collecting a large amount of labeled training data can be a difficult and time-consuming process. In the case of the training data for appearance-based statistical object detection, this typically entails labeling which regions of the image belong to the object of interest and which belong to the non-object part of the image, often called the clutter, and marking landmark points on the object, e.g. Figure 1.

Labeling pixel class membership is usually accomplished in one of two ways. The first, and most common, approach is to capture the training images of the objects on a featureless background. It is then relatively straightforward to automatically segment the object pixels from the background pixels. The second approach is to utilize training images which contain the object, as well as other objects (clutter), and to manually label the regions of the image which contain the object of interest. The first approach has the advantage that once the training data is collected, the data preparation step is relatively straightforward. The disadvantage is that the training images need to be collected under controlled conditions specifically for this task. The second approach has the advantage that the conditions under which the training images are captured are less stringent. Also a larger set of images becomes usable as training data, specifically images which were not necessarily captured for this purpose. The disadvantage is that the data preparation step can be quite time-consuming and is always a manual process. This thesis proposes to develop and evaluate an approach in that training images which have been labeled in a less stringent manner can be utilized, thereby reducing the complexity of this process.

The goal of the approach proposed here is to simplify the collection and preparation of this training data by utilizing a combination of data labeled in different ways. In its most basic form, the data can be fully labeled or unlabeled, where each region of the image is specifically labeled with its class or is labeled as having an unknown class, respectively. Note that the labels may be more complex than simply whether the object is present or not, but may include other latent variables like object size, location, position, etc. In what we call “weakly labeled” training data, the labeling of each of the image regions can take the form of a probability distribution over labels. This makes it possible to capture a variety of information about the training examples. For example, it is possible to indicate that, for a specific region of a specific image, there is a high probability that the object of interest is present and that the rest of the image is unknown. Or it is possible to encode the knowledge that a specific image has a high likelihood of containing the object (or that it must contain the object), but that the object’s position is unknown. The data preparation effort can be minimized by allowing data labeled in different ways to be utilized during training.

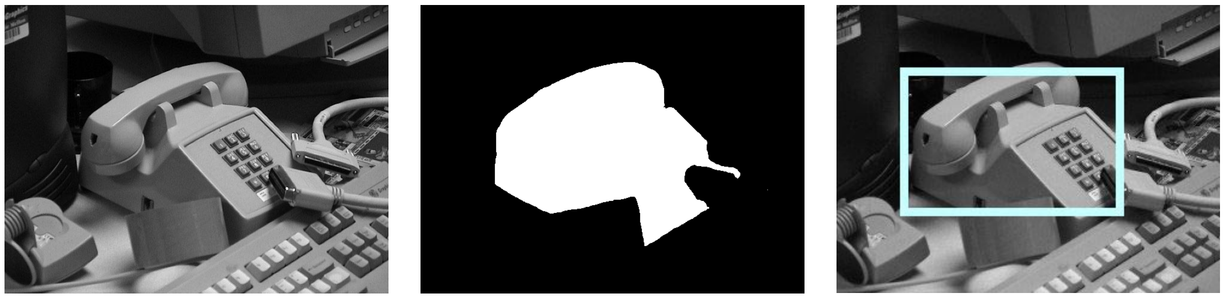


Figure 1: Example input images. The first image (a) is an unlabeled input image, (b) is label information in the form of a mask, where white pixels indicate the object, and (c) is an example detection plotted on the image.

Another possible benefit of using weakly labeled data is that the training images contain the objects in their natural context. This means that models can be built that use the natural context to aid in the detection process. This context could be local, such as bottles are usually resting on tables or faces are usually above bodies. Or the context could be global, such as horses are more likely to appear in outdoor scenes than indoor scenes. These contexts would condition the probability estimates used in the statistical models for object detection. In this scenario, a model of clutter (non-object image regions) in images that contain the object could be built to distinguish images that contain the object of interest from those that do not.

One scenario that illustrates the usefulness of this approach is to imagine that the goal is to train an object detector for a specific object and that images of that object have been captured over a large number of view points as well as a number of negative training examples, images that do not contain the object. In the approach described here, it should only be necessary to label a small number of images containing the object or, in the limit, possibly just one. The rest of the images containing the object would just be "weakly labeled." This information would be incorporated into the model which can then be used to interpret other training examples. This thesis works toward the development of an approach to accomplish this goal and to evaluate this approach using different object detection systems and detection tasks that involve real world images. This work is performed in the context of existing object detection models. It is not the goal of this thesis to design a novel object detection approach, but to extend and modify existing approaches to make them trainable using a mixture of weakly labeled and fully labeled data.

1.2 Definitions

In the context of this work we assign specific specific meanings to certain terms. In this section we provide definitions for those terms.

1.2.1 Statistical Appearance-Based Object Detection System

A detection system, which given an input in the form of a single two dimensional image, generates an output that indicates the presence or absence of a specific object in the given image. If present, the location and scale of the object and possibly its shape and pose may also be output. A system is "appearance-based" if it

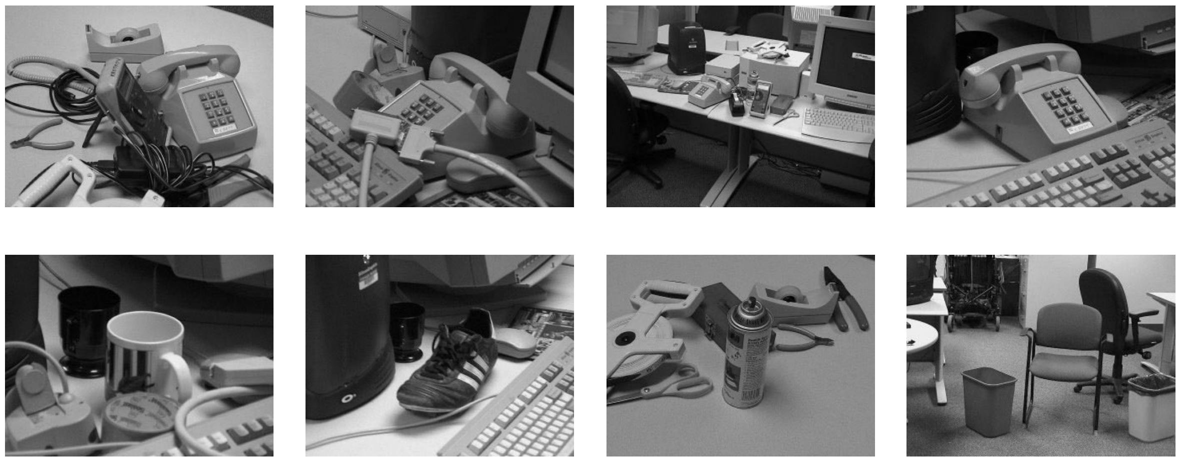


Figure 2: Examples of weakly labeled images. The images in the first row would be labeled as containing the "phone" object. The images in the second row would be labeled as not containing the object.

models the appearance of the object given a set of training images of the object. It is "statistical" because it uses a statistical model to capture that appearance.

1.2.2 Training Data Label Information

Training data label information refers to all of the information needed by the object detection system for training the parameters of its statistical model. Different object detection systems require different label information for each training example. Some common pieces of information are: object presence, object location, object scale, object pose, landmark location and pixel class membership. Given a training image, these are the latent variables whose values are needed to train the model parameters.

1.2.3 Fully Labeled Training Data

Fully labeled training data refers to training data with all of the training data label information necessary to train the model parameters fully specified. This means that exact values are provided for all of the latent variables. For example see Figure 1, where subfigure (a) is an example input image and subfigure (b) is an example of a pixel-wise labeling shown as a mask.

1.2.4 Weakly Labeled Training Data

In weakly labeled data, which is the focus of this work, only partial or probabilistic information is provided for the training data label information and the associated latent variable values. For example, this may mean that an image is labeled as containing a single instance of the object but its exact position and scale are left unspecified. By unspecified, we mean that a distribution over the values of the latent variables is given. For example, the only label information that would be provided for the images in the first row of Figure 2 would indicate that those are images which contain the "phone" object. The images in the second row would be labeled as not containing the phone object. No additional information would be provided.

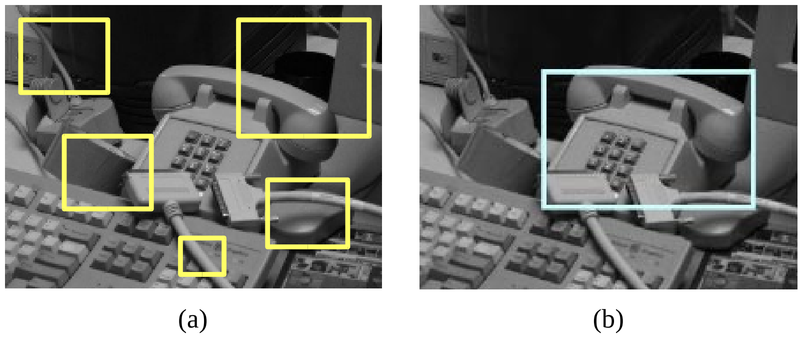


Figure 3: Example detection windows. There are many possible detection windows in each image as illustrated in (a); however, only a single (or a smaller number) of those windows constitute correct detections.

1.2.5 Unlabeled Training Data

Unlabeled training data is a special case of weakly labeled training data. In the case of unlabeled data, a specific distribution, typically a uniform distribution, is assigned to the latent variables. Another important difference is that, in the context of this work, weakly labeled images will contain one or more instances of the object, whereas, an unlabeled image can contain zero or more instances of the object. For example, the information given for the image in Figure 2 would only indicate that these are typical images which we are likely to encounter, but no information would be provided as to whether the phone object is present in the image or not.

1.3 Semi-Supervised Training and the Object Detection Problem

Semi-supervised training is not a new topic in the machine learning community. Many papers, for example [Nigam 98, Joachims 99, Szummer 02, Zhu 03a], and workshops have concentrated on this topic. Relatively few, for example [Selinger 01, Fergus 03], have addressed this issue in the context of object detection systems. And, to the best of our knowledge at this time, there has been no comprehensive investigation as to which approaches are best suited to this application domain and the factors that affect the performance, usefulness, and applicability of semi-supervised training.

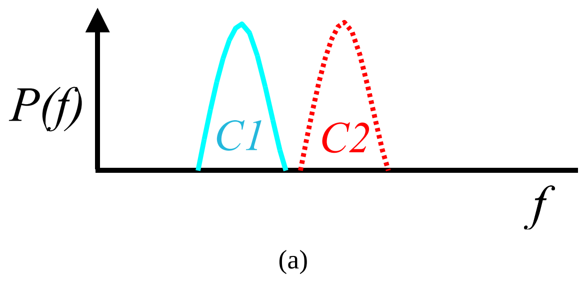
It is reasonable to ask why prior work in the area of semi-supervised training is not directly applicable and sufficient to solve the problem at hand. We believe that there are specific attributes of the image detection problem which are unique and make this particular application of semi-supervised training significantly different from other semi-supervised applications which have most frequently involved text.

One unique aspect is that a single image is potentially the source of many training examples, as was partially explored by Maron and others in the context of multiple instance learning, [Maron 98a, Maron 98b, Zhang 01, Zhang 02]. This is because each training example consists of a subwindow of the image and every image contains an extremely large number of subwindows, especially when one considers all possible locations, scales, and orientations that are possible in a single image, as illustrated in Figure 3a. Because the object to be detected only occurs infrequently in the image and, in any given image, the number of subwindows that would be labeled to correctly contain the object is very small, as illustrated in Figure 3b.

One consequence of the fact that objects are rare is that negative examples are typically plentiful and easy to acquire and label [Viola 01, Wu 03]. Because the objects to be detected are relatively rare, with high probability [Viola 01] any arbitrary image patch will not be a positive example of the object. Because of this, if we gather a large set of random image patches we will have many negative examples and even if we do happen to collect some image patches which contain an object, the number of false negatives will be very small. The second issue is that it is much easier to fully label negative examples than positive examples. A large set of negative examples can be generated by finding images which do not contain the object of interest. Any such image can be used to generate a large set of negative examples because any subwindow of the image will not contain the object. This, of course, relies on the assumption that the distribution of negative examples in images which do not contain the object is the same as images which do contain the object.

If one considered all of the large number of subwindows which are all potential training examples in a single image, this problem would be computationally intractable. However, the characteristics of the problem mean that we can take a simplifying approach which will not have an adverse effect on the final performance in the context of weakly labeled images. In this context, we will assume that each image weakly labeled to contain the object contains at least a single instance of the object. Potentially, we would have to enumerate and assign a label to every possible subwindow in the image. However, since we know negative examples are plentiful and positive examples are rare, we can restrict ourselves to just looking for the single subwindow which we believe to have the highest likelihood of being a positive example. This will greatly decrease our computational complexity, because now instead of enumerating all possible subwindows, we only need to estimate the maximum likelihood values of a set of latent variables which identify the subwindow which contains the object, for example: location, scale, and orientation. The disadvantage is that we discard additional positive examples in the image, but since we consider objects to occur only rarely, this will not greatly decrease the number of potential positive examples. The other disadvantage is that we typically discard the subwindows of the weakly labeled data as negative examples. But again, this should not be detrimental, because negative examples are generally plentiful. An interesting side effect of this approach is that, as it iteratively estimates the values of the latent variables using the “current” detector during the training process, the estimates for these latent variables will change. This means that the specific examples extracted from our weakly labeled data set will change with training and are dependent on our current model. This is unique to the object detection problem.

Another unique aspect of the detection problem which makes this a difficult domain for semi-supervised learning is that there is typically significant overlap between “object” and “clutter” (or “non-object”) classes. This is because in the detection problem everything that can possibly occur in the class of images likely to be observed that is not the object is a member of the clutter class. And, as one might imagine, there are many things that will look very similar to the object to be detected. For example, detection windows which partially overlap the object are classified as instances of the non-object class. This is illustrated schematically in Figure 4 for a two class classification problem. In this Figure, the x-axis represents a single feature value f and the y-axis represents the probability of that feature being observed. Figure 4a illustrates the situation where the two classes *C1* and *C2* are well separated. In this situation, the semi-supervised learning problem is easy. Given a set of unlabeled data points, they can easily be assigned to their corresponding cluster and then, given a single noise-free example, the class identity of both clusters can be determined. Unfortunately the object detection problem is more like the situation illustrated in Figure 4b where class *C1* is a schematic representation of the object features distribution and *C2* is a schematic representation of the clutter class feature distribution. In this situation a simple clustering algorithm will not be able to cleanly separate examples into two classes, a more complex approach will be necessary. This is the problem which we will address in this thesis.



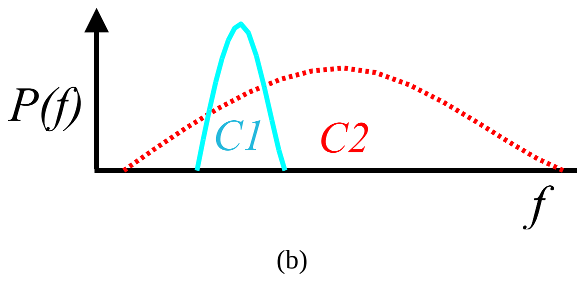


Figure 4: A schematic representation of two possible sets of distributions for a two class problem. In (a) the classes are well separated; however, in b there is significant class overlap.

1.4 Document Organization

The remainder of the document is organized into the following chapters:

- Prior Work
- Approach
- Color Based Detector Experiments
- Filter Based Detector Experiments
- Schneiderman Detector Experiments
- Discussion, Open Issues, Future Work and Conclusions

In Section 2.4, we introduce prior work in the field of semi-supervised training in general and its specific application to computer vision. We then move on to discuss the details of our approach and then analyze its behavior when applied to a synthetic data set in Section 2. In that section, we also motivate the key issues that guide our experimental investigation. We then move to the main experimental work of the thesis, where we apply semi-supervised training approaches in the context of the object for real world images. In Sections 3, 4, and 5 we present and evaluate three detectors in increasing order of complexity: a color based detection system, a filter based detection system, and in the context of the well known Schneiderman detector as described in [Schneiderman 03]. We then move on to a final discussion of the results, describe some possible future work and applications, and conclusions in Section 6.

2 Approach

2.1 Overall Approach

The goal of this work is to evaluate the performance of, and characterize the parameters related to the semi-supervised training of object detection systems. To that end, we have taken an empirical approach to characterizing the behavior of principled semi-supervised training techniques in the context of object detection. We believe that an empirical approach is justified because of the complex nature of image data. This parallels the investigations of the detection problem itself, where principled statistical techniques are used to construct detection systems, but it is often unclear whether a system will actually work until it is evaluated on real image data. We believe that this is also true in the context of semi-supervised training because semi-supervised training approaches are extremely sensitive to the underlying data distribution. If the classes are well separated then the problem can be easily solved; however, the behavior when the distributions are more complex is nearly impossible to predict.

In the recent literature, [Selinger 01, Fergus 03], anecdotal evidence has been presented which suggests that semi-supervised training can provide a performance improvement when applied to the object detection problem. However, we are left with very little understanding as to which techniques to apply in specific situations and how these techniques and their associated parameters affect final performance. In the work presented here, we perform a comprehensive empirical evaluation with the goal of characterizing and understanding these issues to facilitate the broad practical application of semi-supervised training to the object detection problem.

Specifically, the main questions we will address are:

- What is the sensitivity of object detection systems to training set size?
- How do the performances of specific semi-supervised training approaches compare and how do the parameters of those approaches affect final detection performance?
- How does the distribution of the fully labeled data affect the final performance?
- How do specific detectors interact with different semi-supervised training approaches?
- Are certain semi-supervised training approaches generally applicable across a broad range of detectors?
- Are the same techniques applicable across different objects?

The complex nature of image data makes these questions extremely difficult to answer in the abstract. However, answers to these questions will go a long way toward making semi-supervised training of object detection models a standard tool. And we believe that the only way to obtain these answers is through a careful and comprehensive empirical evaluation.

2.2 Semi-Supervised Training Approach Overview

2.2.1 Introduction

In this work, we examine two established approaches to semi-supervised training: Expectation-Maximization (EM) and self-training. The key issue in semi-supervised training is how to utilize training data when important latent variables, such as the class labels in the generic unlabeled case, or the object location in an object detection application, are missing. Accordingly, a fundamental component of most of these approaches is the assignment of distributions or specific values to these latent variables. At a very abstract level, the approaches which we will examine in this work iteratively refine the latent variable estimates and the statistical model parameters in a joint process.

2.2.2 Expectation Maximization

One approach is Expectation-Maximization (EM) as introduced in [Dempster 77] and as described in [Bishop 95] and [Mitchell 97]. This is a very generic method for generating maximum likelihood estimates of model parameters given unknown or missing data. Algorithmically, this is implemented as an iterative process which alternates between estimating the expected values of the unknown variables and the maximum likelihood of values of the model parameters. With each iteration, we are guaranteed to move the model parameters in a direction that either increases or does not change the likelihood of the data under the model. Mathematically this is accomplished by maximizing a lower bound of the likelihood of the data with each iteration. The caveat is that EM is not guaranteed to find the globally optimal set of parameter values. However, there are variants of EM which use annealing and randomization to reduce the likelihood of getting stuck at a local maxima.

It would seem that, local maxima issues not withstanding, EM would be the ideal approach to semi-supervised learning. Indeed, work using EM in the context of text classification by Nigam in [Nigam 98, Nigam 01] has found that EM is indeed a useful approach to training models using weakly labeled data. However, Nigam also found that there were instances in which EM did not perform well. One of the goals of this work is to explore the issues related to why and whether there are approaches which can do better than EM.

There are many reasons why EM may not perform well in a particular semi-supervised training context. One reason is that EM solely finds a set of model parameters which maximize the likelihood of the data. The issue is that the fully labeled data may not sufficiently constrain the solution, which means that there may be solutions which maximize the data likelihood but do not optimize classification performance. We may have additional information about the problem to be solved which EM does not take advantage of and therefore prevents it from solving our problem optimally. There have been a variety of approaches to incorporate new information into EM and to design alternate algorithms which can utilize additional prior information which we may have about a specific semi-supervised problem. Some examples are the work described in [Szummer 02], [Rachlin 02], [Cozman 03b]. These are discussed in more detail in the section on prior work. We chose to investigate the self-training approach which we discuss in the following section.

2.2.3 Self-Training

The second semi-supervised training approach which we will evaluate is often called self-training, incremental training, or bootstrapping. (This should not be confused with the technical definition of the term as used in the Statistics literature.) In self-training, an initial model is constructed using the fully labeled data. This model is used to estimate labels for the weakly labeled (or unlabeled) data. A selection metric is then used to decide which of the weakly labeled examples was labeled correctly. Those examples are then added to the training set and the process repeats. Obviously the selection metric chosen is crucial here; if incorrect detections are included in the training set then the final answer may be very wrong. This issue is explored throughout the thesis. Another choice is whether to re-estimate the labels (or other latent variables) each time the model is updated or to keep them fixed once they are estimated.

2.2.4 Self-Training and the Selection Metric

One of the most important issues of the incremental approach taken by self-training is the selection metric utilized to determine which image or images to add next. The most straightforward scoring function is the likelihood of the detection on the weakly labeled data. However, this may not be accurate given the small amount of data used to train the model. Also, if an image is chosen with an incorrect detection, it may be difficult for the model to recover from this mistake. Accordingly, we may want a scoring function which is conservative. That is, a metric which only gives an example a high score if it is very certain of its score value. One class of scoring functions are models which only give detections high scores if they are near existing labeled data, but have score values that quickly fall off as the distance increases. One possibility here is a model which is very conservative and only models a small portion of the feature space, assigning the highest scores to that portion of the space.

Another approach is to use a model which is different than the actual model we want to use for our detector. Training a more conservative detector makes the scoring function more conservative. Such a detector would not be able to handle large variations in image appearance, like scale, lighting, rotation. It would only be able to detect images which are very close in appearance to the existing training images. However, as more and more weakly labeled images are labeled, its ability to handle such variations would be increased. Of course, we would like our final detector to be able to handle such variations. To accomplish this, the set of weakly labeled images which have been labeled in the training of the conservative model can now be used, along with the fully labeled examples, to train a detector which has the invariance properties we seek.

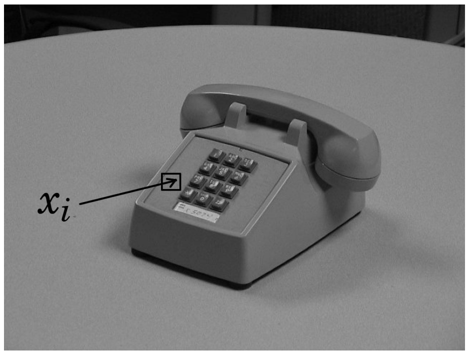
2.2.5 Comparison of Approaches

At first glance it may seem that the EM approach and the self-training approach are only superficially similar. In this section we attempt to draw parallels between the EM and the self-training approaches to demonstrate how they are actually different points in a common design space of algorithms. The two approaches are similar in that they are iterative algorithms which alternate between estimating the labels of the weakly labeled data and updating the model parameters. We have summarized the parallels between the two approaches in Table 1. The first point is label types. In EM, typically “soft” assignments are made, that is a distribution over label values is maintained; in self-training, typically “hard” assignments are made, that is a single label value is selected. Of course, one could implement a non-standard version of EM with hard labels and a non-standard version of self-training with soft labels.

The second point is training set increment. Typically in EM, all of the data is used at every iteration. In self-training at each iteration, a small amount of data from the weakly labeled set is labeled and added to the training set incrementally. A version of EM can bridge the gap between the two approaches by setting

	EM	Self-Training
Label Types	Soft	Hard
Training Set Increment	All at Once	Partial
Label Assignments	Updated	Fixed or Updated

Table 1: This table summarizes some of the differences in algorithm attributes for EM and self-training.



(a)

(b)

Figure 5: Example images. Image (a) contains the object with an input feature vector location labeled as x_i. Image (b) is an example of an image containing only clutter.

the weight of certain examples to zero. These examples correspond to those not selected by the self-training process.

The third point is label assignments. Typically in EM, the assigned labels are updated at each iteration when the model is updated. In self-training the assigned labels can be updated or they can be fixed once they are estimated.

Based on these three observations, one can see that self-training is exactly equivalent to EM if the training set increment is the entire data set at once, soft labels are used, and label assignments can be updated. The variant of self-training used in this work is not equivalent to standard EM, it uses hard labels, a partial training set increment, and fixed labels.

2.3 Semi-Supervised Training Approach Details

2.3.1 Framework Introduction

In this section we introduce the notation and general framework for our semi-supervised training approach in the context of a two class classifier for feature vectors based on a generative model. We present this in the context of an object detection system with the purpose of classifying an image as being a member of the "object" class or the "clutter" class; for an example see Figure 5. The system is based on the values of the feature vectors associated with each pixel in the image, where each pixel has been assumed to be generated independently.

To more formally describe this model, we designate the image feature vectors as X, with x_i being the data at a specific location in the image such as indicated in Figure 5a, where i indexes the image locations from

$i = 1 \ldots n$. Our goal is to compute $P(Y \mid X)$, where $Y = object$, an image containing an object, or not $Y = clutter$, an image without the object.

Associated with each image class is a particular generative model, which is equal either to the foreground model f or background model b. We use θ_f to indicate the parameters of the foreground model and θ_b for the background model. For example, in the case of a model consisting of a single Gaussian, the parameters θ would consist of the mean vector μ and the covariance matrix Σ. The prior probability that a particular model is selected is $P(M = m)$, where m take on the value either f or b.

Applying Bayes rule we can compute $P(Y \mid X)$ in terms of quantities from our generative model:

$$P(Y \mid X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Taking $P(Y)$ as uniform, we need to find expressions for $P(X \mid Y)$ and $P(X)$. For $P(X \mid Y = object)$, we note that given $Y = object$, we choose to model the probability of the observed feature data at each image location as independent:

$$P(X \mid Y = object) = \Pi_{i=1}^{n}(P(x_i \mid \theta_f)P(M = f) + P(x_i \mid \theta_b)P(M = b))$$

For images which contain both the object and clutter, pixels outside the object bounding box are modeled identically to clutter pixels. Because of this, only pixels inside the bounding box need to be considered when computing the likelihood ratio. If we set $P(Y = object) = P(Y = clutter)$, the likelihood ratio over the entire image is:

$$\frac{P(Y=object|X)}{P(Y=clutter|X)} = \Pi_{i=1}^{n}\frac{P(x_i|\theta_f)P(M=f)}{P(x_i|\theta_b)P(M=b)}$$

The value of this likelihood ratio could be thresholded to determine the presence or absence of an object. In practice, the detection is performed in a subwindow of the image. We make that explicit in Subsection 4.3.

2.3.2 Training the Model with Fully Labeled Data

The parameters of the probability distributions in this model can be learned from fully labeled training data in a straight-forward manner. Given the class, y_i, of every pixel in every image in the training set, the parameters of the object and clutter class distributions, $P(x_i \mid \theta_f)$ and $P(x_i \mid \theta_b)$, can be estimated independently using standard fully supervised training techniques by training the object class model only using the data points labeled as object and training the clutter class model only using the data points labeled as clutter.

2.3.3 Batch Training with Weakly Labeled or Unlabeled Data

In our approach, the data is weakly labeled in the sense that the object of interest is known to be in the training image, but the location of the object and which pixels correspond to the object are not known. We will present our approach in the context of unlabeled data, which is the simpler case because we have already introduced this model as a means of classifying an entire image. There are two variants to this approach which, as we will show, result in very different final models: *batch training* and *incremental training.* Batch

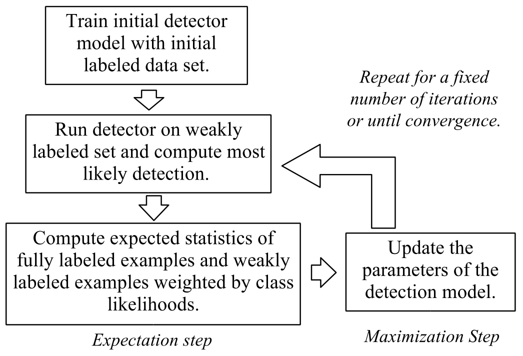

Train initial detector model with initial labeled data set.
Run detector on weakly labeled set and compute most likely detection.
Compute expected statistics of fully labeled examples and weakly labeled examples weighted by class likelihoods.
Update the parameters of the detection model.
Repeat for a fixed number of iterations or until convergence.
Expectation step
Maximization Step

Figure 6: A schematic representation of the batch training approach with EM. The labels for all of the weakly labeled data are re-estimated at each iteration and all of the data is used to update the model parameters.

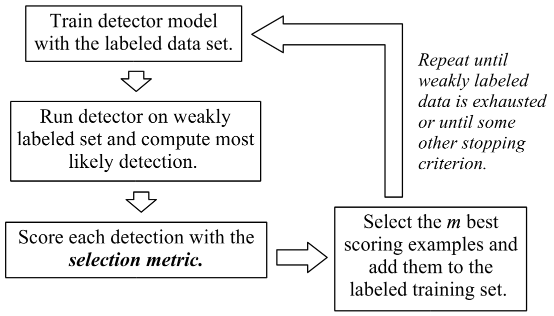

Train detector model with the labeled data set.
Run detector on weakly labeled set and compute most likely detection.
Score each detection with the selection metric.
Select the m best scoring examples and add them to the labeled training set.
Repeat until weakly labeled data is exhausted or until some other stopping criterion.

Figure 7: A schematic representation of the incremental training approach with EM. Weakly labeled examples are incrementally added to the training set based on their selection score.

training utilizes all of the data at every iteration and is described in this section. Incremental training selects new training images to use as training progresses and is described in the following section.

A schematic representation of the training procedure is presented in Figure 6. The complete training procedure for using a combination of unlabeled and fully labeled data is as follows:

1. Compute the expected values of the sufficient statistics of the fully labeled and weakly labeled object data, weighting the weakly labeled examples by $P(M_i = \theta_f \mid x_i)$ using EM.

2. Execute the "M" step of EM, by updating the parameters of the foreground model using the sufficient statistics computed in step 1.

3. Repeat steps 1-2 for a fixed number of iterations or until convergence.

In this work we repeat the inner loop of this algorithm for a fixed number of iterations.

2.3.4 Incremental Training Approach with Weakly Labeled Data

In contrast to the batch training described in the previous section, incremental training grows the pool of training examples as part of the training process. When discussing the incremental addition of training data, it is useful to define the following terms:

- The *initial labeled training set* is the initial set of fully labeled data: $L = \{L_1 \ldots L_{m_l}\}$
- The *weakly labeled training set* is the current set of weakly labeled data: $W = \{W_1 \ldots W_{m_w}\}$
- The *current labeled training set* is the seed training set in addition to any weakly labeled examples which have been assigned labels: $T = \{T_1 \ldots T_{m_t}\}$

We utilize the object detection framework detailed in the previous sections as the basis of our weakly labeled data approach. This approach begins with an initial set of model parameters trained using the initial labeled training set provided as detailed in the previous section, $M^0 = \{\theta_f^0, \theta_b^0\}$. This serves as the starting point for our weakly labeled data approach during which we modify foreground model, θ_f, but keep the background model, θ_b fixed.

Let us consider the case, in which we are training with T and we want to add W. We first want to adapt the foreground model, θ_f, using a combination of fully labeled, T, and weakly labeled, W, images using EM. Given a training set $T = L \cup W$, the main difference from the fully labeled data case is that the contribution of the features computed from each pixel of each training example to the sufficient statistics in the expectation step is weighted according to likelihood that the data belongs to the current foreground model, θ_f, given the current model, M, the desired probability at pixel i can be computed as follows:

$$P(M_i = f \mid x_i) = \frac{P(x_i|\theta_f)P(M_i=f)}{P(x_i|\theta_b)P(M_i=b)}$$

For fully labeled data the weight is 0 or 1.

The weakly labeled data approach relies on being able to estimate where the object is in the training image using the current model. However, since the initial model, M^0, is trained using a limited amount of data, this may not be possible, especially for weakly labeled training data which differs significantly in appearance from the training images. One approach is to immediately add all of the weakly labeled data, W, to the training set, T, as described in [Rosenberg 02]. However, incorrect labels can potentially "corrupt" the model statistics.

In the approach we detail here, we attempt to reduce the impact of this issue by labeling weakly labeled examples and adding them incrementally to the training set according to our confidence in those labels, similar to the methods described in [Blum 98, Nigam 00, Selinger 01]. Here, the order in which the images are added is critical to allow the model to first generalize to images which are most similar to the initial training set, and then incrementally extending to views which are quite different from those in the original training set.

A schematic representation of the training procedure is presented in Figure 7. The complete incremental training procedure for using a combination of weakly and fully labeled data is described as follows:

Initialization:

1. Train the parameters of the initial model, M^0, consisting of the foreground θ_f^0 and background θ_b^0 models using the fully labeled data subset and EM. Initialize the initial labeled training set, T^0, with the provided fully labeled data.

Beginning of Iteration j:

1. For each W_k in W^j compute the selection metric, $S_k = \mathrm{Sel}(M^j, W_k)$.
2. Select the weakly labeled example, $W_{\hat{k}}$ where $\hat{k} = \mathrm{argmax}_k S_k$ with the highest score and update both the current training set and the weakly labeled training set, $T^{j+1} \leftarrow T^j \cup \{W_{\hat{k}}\}$, $W^{j+1} \leftarrow W^j - \{W_{\hat{k}}\}$.

EM Loop:

1. Execute the "E" step of EM, by computing the expected values of the sufficient statistics of the fully labeled and weakly labeled object data, weighting the weakly labeled example pixel features vectors by $P(M_i = f \mid x_i)$.
2. Execute the "M" step of EM, by updating the parameters of the foreground model using the sufficient statistics computed in the "E" step to compute the new foreground model θ_f^{j+1}.
3. Repeat steps 1-2 for a fixed number of iterations or until convergence.

End of Iteration j: While $W \neq \emptyset$, start new iteration $j+1$.

A natural question which arises is how to make the choice of the *selection metric*, which is critical in controlling the order in which exemplars are inserted in the model at the start of each iteration.

2.3.5 A Specific Example: Weakly Labeled data for a Gaussian Mixture Model with EM

To better illustrate the operation of the training algorithm, we present in this section an example of using a Gaussian mixture to model the class conditional generative probability distributions; then expectation-maximization (EM) can be used to estimate the distribution parameters from weakly labeled data.

In this example we assume that a set of n_f continuous valued features are computed for training image s_i. The generative model captures the class conditional probability of observing a particular set of feature values. Again for simplicity, we assume that the generative model for each class is a single Gaussian with a diagonal covariance matrix over the observed features, f_{ik}, where i indexes the training example and k is the feature index, for example s_i. However, this approach is extensible to full covariance matrices and a larger number of Gaussians in a straightforward manner. In this approach the generative model for each class becomes:

$$P(s_i \mid l_i = c_j, \theta_j) = \prod_{k=1}^{n_f} \left(\sqrt{2\pi}\sigma_{jk}\right)^{-1} \exp\left(-\frac{1}{2}\frac{(f_k-\mu_{jk})^2}{\sigma_{jk}^2}\right)$$

Given the full mixture model, the likelihood of observing a specific piece of data becomes:

$$P(s_i \mid \theta_j) = \sum_{j=1}^{n_c} P(s_i \mid l_i = c_j, \theta_j) P(l_i = c_j) = \sum_{j=1}^{n_c} \prod_{k=1}^{n_f} \left(\sqrt{2\pi}\sigma_{jk}\right)^{-1} \exp\left(-\frac{1}{2}\frac{(f_k-\mu_{jk})^2}{\sigma_{jk}^2}\right) P(l_i = c_j)$$

Given a fully labeled training set, S, and introducing the indicator variable z_{ijk}, we can estimate the parameters of this model using EM. Where the expectation step is:

$$E[z_{ijk}] = \frac{\sigma_{jk}^{-1} \exp\left(-\frac{1}{2}\frac{(f_{ik}-\mu_{jk})^2}{\sigma_{jk}^2}\right)}{\sum_{v=1}^{n_c} \sigma_{vk}^{-1} \exp\left(-\frac{1}{2}\frac{(f_{ik}-\mu_{vk})^2}{\sigma_{vk}^2}\right)}$$

And the maximization step is:

$$\mu_{jk} \leftarrow \frac{\sum_{i=1}^{n_s} E[z_{ijk}] f_{ik}}{\sum_{i=1}^{n_s} E[z_{ijk}]}$$

$$\sigma_{jk}^2 \leftarrow \frac{\sum_{i=1}^{n_s} E[z_{ijk}] f_{ik}^2 - \left(\sum_{i=1}^{n_s} E[z_{ijk}] f_{ik}\right)^2}{\sum_{i=1}^{n_s} E[z_{ijk}]}$$

In the weakly labeled approach, the training data specifies a prior distribution over class labels for each training example, $P_i(l_i = c_j)$. This results in the following modified expectation step of the EM estimation procedure:

$$E[z_{ijk}] = \frac{\sigma_{jk}^{-1} \exp\left(-\frac{1}{2}\frac{(f_{ik}-\mu_{jk})^2}{\sigma_{jk}^2}\right) P_i(l_i=c_j)}{\sum_{v=1}^{n_c} \sigma_{vk}^{-1} \exp\left(-\frac{1}{2}\frac{(f_{ik}-\mu_{vk})^2}{\sigma_{vk}^2}\right) P_i(l_i=c_v)}$$

The maximization step of the EM algorithm remains unchanged. Using this modified version of EM, it is possible to estimate the parameters of a Gaussian mixture model using weakly labeled data.

This section detailed the steps for modifying a simple generative model to incorporate weakly labeled data in its training process. As part of the proposed work of this thesis, this model will be empirically evaluated and other models will be modified to incorporate weakly labeled data and will also be empirically evaluated. Experiments using the approach detailed in this section are described later in this document.

2.4 Prior Work

2.4.1 General Object Detection

In recent years, object detection methods based on statistical models of object appearance have been quite successful [Burl 95, Rowley 98a, Schneiderman 98, Schneiderman 04a, Schiele 00, Viola 01]. The work proposed in this thesis will be an approach for training the parameters of these methods, but will not be a new method in and of itself. Most of the methods described in the literature have a similar structure: a set of features is computed from an input image and a likely object classification is computed based on a statistical model of feature responses with parameters estimated from a fully labeled training set. Where these methods differ is in the specifics of the features computed, the details of the statistical model used, and the handling of invariants. Table 2 summarizes some of the most relevant work in this area.

2.4.2 Unlabeled / Weakly Labeled / Multiple Instance Data

In recent years there has been a good amount of activity in the incorporation of unlabeled data into the training process, [Nigam 98, Joachims 99, McCallum 00]. Most of this work has been focused on document

Reference	Features Used	Model Characteristics
[Burl 95, 96, 97,98]	small texture patches at interest points	Gaussian over joint feature positions
[Moghaddam 97]	raw pixels projected to low dimensional PCA space	Gaussian model of training data in PCA space
[Schiele 00]	Gaussian derivative filters over four scales	model of joint feature responses in an image region, no positional information
[Schneiderman 98, 00a, 00b, 03, 04a, 04b, 04c]	subregion PCA and wavelet coefficients	learn statistical structure of wavelet coefficients, subregion model varies with relative position
[Viola 01]	multiple features computed over an image pyramid	magnitude of features responses over an entire image

Table 2: This table summarizes some of the most relevant prior work in the area of object detection.

classification and was inspired by the existence of the Web and easy access to a large amount of unlabeled documents.

The multiple instance approach was introduced in [Dietterich 97] in the context of a drug discovery problem where many experiments were being performed each with a potential different mix of drugs. The success or failure of an individual experiment could be determined, but the goal was to determine which specific drug was the active one. The hypothesis class explored here was axis aligned rectangles.

The multiple instance approach was more extensively explored in [Maron 98a, 98b, 98c]. Here the concept of diverse density was introduced, which was a probabilistic measure of intersection between bags of examples. This approach was applied to a number of probabilistic models and tested on a number of problems including stock evaluation and image retrieval. Both generative and discriminative approaches were developed, but the consequences of using these different approaches was not evaluated.

The work which is most similar to this work is that described in [Weber 99, 00a, 00b]. In this work an object detection system is trained using images which are labeled indicating the presence or the absence of the object of interest. The model used is generative. A search approach is used to find likely correspondences between detected features and model features. Success is reported in building models for both frontal and profile faces. The work proposed in this thesis is similar, but extends the prior work in that an evaluation of a mix of labeled and multiple instance data will be performed, the problem will be examined in a discriminative context and the incorporation of other types of labeled data will be performed.

2.4.3 Weakly Labeled Data and Images

Work has been done in the past using weakly or unlabeled data to train object detection models. Some of the earliest work is described in [Baluja 98] . The authors used an Expectation-Maximization (EM) approach similar to one of the approaches which is explored here.

More recent work by Selinger in [Selinger 01] uses an incremental approach similar to the approach described here. One of the main differences is that her approach is contour based. Also she uses the model output as the scoring metric to decide which image to add next, whereas we found that other metrics tended to work better. As we will show, the choice of a selection metric suitable for training is critical; it will be the basis for much of our experimental investigation.

Recent work by Fergus, Perona, and Zisserman in [Fergus 03] utilizes an EM based approach and has demonstrated good results. However, we believe that specifics of their detector and experimental set up mean that the results in this work are not generally applicable

2.4.4 Graph based Semi-Supervised Approaches

A number of papers have taken the approach of representing the relationships between labeled and unlabeled data using a graph with weighted edges. Typically the weights on the edges of these graphs are inversely related to the distance or similarity between the different examples in feature space. Also typically, the graphs are not completely connected, but connections are only made to the k-nearest neighbors, or some other heuristic is used to remove long-distance connections. The removal of these edges is critical for these algorithms because the edges of the graph are used to capture the notion that similar examples should be labeled similarly. If distant connections are not severed, then large amounts of distant unlabeled data can swamp out the evidence from the labeled examples. That is, since the amount of unlabeled data increases to infinity if long-distance connections are not removed, the algorithm will not have the asymptotic behavior we desire.

One of the first papers in this area was by Blum and Chawla in [Blum 01]. In this paper the graph is augmented with special “class” nodes which represent the positive and negative classes. Infinite weight edges are put into the graph from the labeled examples of the appropriate class to the special “class” nodes. The algorithm they used to decide the labeling of the unlabeled data is the minimum cut algorithm. Because the special “class” nodes have infinite weight, the edges to those nodes will not be cut, so in a two class problem the graph will be separated into two pieces. Unlabeled examples will either end up connected to the “positive class” graph or the “negative class” graph. Since the minimum cut algorithm removes low weight edges, it cuts the graph between examples that are far apart from one another. An interesting aspect of this work is that an analysis was performed which showed that when the algorithm is applied to particular graph structures and edge weights, it corresponds to optimizing specific learning criteria.

The next set of ideas in this area again uses the graph to capture the idea that examples which are similar in feature space should have similar labels. These papers utilize the idea of a random walk. The intuition is that if a random walk was started at an unlabeled example and transitioned along edges of the graph with probability related to their edge strengths, how often would positive or negative labeled examples be encountered. This approach has two advantages. The first is that a likelihood of the label being assigned to the unlabeled example can be computed. The second is that it can follow data manifolds and handle problems like the two spiral problem.

Some of the earliest work here is that of Szummer and Jaakkola and in [Szummer 01]. Good results were shown in the paper. However, the algorithm was difficult to analyze because labeled examples in this model were considered to be absorbing states of Markov random walks of different lengths. Issues with this model were corrected by Zhu and Gharahmani in [Zhu 03a]. Their model does not suffer from the issues of Szummer’s model, and their analysis shows that their approach is equivalent to a Gaussian kernel on the graph. They have also extended this work to active learning, [Zhu 03b].

2.4.5 Information Regularization

A very promising recent direction is that of information regularization by [Szummer 02] and [Corduneanu 03]. The idea here is to formalize a notion which other semi-supervised data approaches take for granted. The notion is exactly what information do we hope to transfer from the unconditional underlying distribution $P(x)$ to the class conditional distribution $P(y \mid x)$. The idea is that the unlabeled data will constrain the final hypothesis in a particular way. Specifically, we want hypotheses that tend not to split high density regions in the underlying distribution. This makes intuitive sense as to what information the unlabeled data can provide, and formalizes our notion of what effect we think the unlabeled data will have on the final hypothesis.

2.5 Simulations

2.5.1 Introduction

The work described in this section will characterize the behavior of the semi-supervised training approaches, which we will evaluate in terms of synthetic data. As described previously, one of the motivating factors of this work is to understand and characterize the behavior of semi-supervised training approaches in the context of object detection. We argue that an empirical investigation is necessary because of the complexity of the image data. However, we believe that it is equally important to understand the behavior of these algorithms under controlled conditions when the data involved has been generated from known distributions. This is an important tool for motivating specific experiments and as a means of testing hypotheses which attempt to explain specific behavior observed during the empirical investigation using real world data.

2.5.2 Simulation Protocol

The simulation protocol we follow is relatively straightforward. It consists of generating synthetic data from a known distribution, applying a semi-supervised training technique to that data and recording the final and incremental results. Two types of results are generated, quantitative performance measures and data visualization. The quantitative performance measure we most frequently use is accuracy. Because we can generate low dimensional data, we can create two dimensional plots to understand how the labeling of the data evolves over time. We present some of these results in this section.

An example is helpful to better understand our simulation protocol. The following is a list of steps in the protocol for self-training:

1. The parameters of the generative data distribution are specified. In the case of a mixture of Gaussians, this would include the means, the covariances, and the mixture weights.
2. Three data sets are generated from the labeled data distribution. One is used as the fully labeled set, one as the unlabeled set, and one as the test set. Even though the labels are known for the unlabeled set, those labels will be hidden from the semi-supervised training algorithms, but they are useful for analysis.
3. The semi-supervised training algorithm is executed for a fixed number of iterations, until convergence or until some other stopping criterion. The intermediate and final performance results are recorded. Plots are generated when low dimensional data are used.

2.5.3 Simulation Results

The goal of this set of simulations was to better understand the behavior of the selection metric used in incremental self-training. To better describe these experiments, it is useful to introduce the following notation:

- X = the input feature vector
- C= the data class
- c_i =the class identifier, $i = 1 \dots m$

We evaluated three different selection metrics for the selection of unlabeled data:

- **Class conditional likelihood metric:** This metric is the likelihood of the data under its estimated generative model, $P(X \mid C = c_i, \theta_i)$. This is a class specific distance metric which weights the dimensions according to the class shape. Unlabeled data points in high likelihood regions of the class will be added first.
- **Confidence metric:** This metric is the confidence we have in a particular estimated class label, $P(C = c_i \mid X, \theta_i)$. Unlabeled data points whose labels are most certain will be added first.
- **Euclidean distance metric:** This metric is the minimum Euclidean distance from the current unlabeled point under consideration to one of the data points in the current labeled training set. This is a “one” nearest neighbor metric. Data points which are close to currently labeled data points will be added first.

To facilitate visualization, we chose a two class problem in which the generative model for each class was a single two dimensional Gaussian. Accordingly the generative model is as follows:

$$P(X \mid C = c_i, \theta_i) = \exp\left(-\tfrac{1}{2}(X-\mu_i)'\Sigma_i^{-1}(X-\mu_i)\right) / \left(2\pi|\Sigma_i|^{\frac{1}{2}}\right)$$

where θ_i= the parameter vector for the Gaussian for class i which includes its mean $\mu_i$and its covariance matrix Σ_i

For the two class problem the data likelihood becomes:

$$P(X \mid \theta_1, \theta_2) = P(X \mid C = c_1, \theta_1)P(C = c_1) + P(X \mid C = c_2, \theta_2)P(C = c_2)$$

In the specific experiment we report here, we chose two Gaussians with diagonal covariance matrices and with means such that there would be a moderate amount of overlap between the two distributions. The specific parameters were:

- **Class 1:** $P(C = c_1) = 0.33$, $\mu_1 = [-1, -2]$, $\Sigma_1 = \begin{bmatrix} 5 & 0 \\ 0 & 4 \end{bmatrix}$

- **Class 2:** $P(C = c_1) = 0.67$, $\mu_2 = [4, 8]$, $\Sigma_1 = \begin{bmatrix} 3 & 0 \\ 0 & 8 \end{bmatrix}$

In this simulation we randomly generated 10 labeled data points total from both classes and 500 unlabeled data points. Both the labeled and unlabeled data generated are plotted in Figure 8a. (Note that this figure is intended to be reproduced in color.) In Figure 8b we also display the hidden class values for the unlabeled data. We plot the labels generated for the unlabeled data at iteration 5 in Figures 8c, 8d, and 8e, for the class conditional likelihood metric, the confidence metric, and the Euclidean distance metric respectively where 50 unlabeled data points are added per iteration. We believe that these plots provide considerable insight into the behavior of these algorithms. We see in Figure 8c that as was expected, data points which are near the mean of the distribution have been labeled first. The behavior in Figure 8d is reasonable, but that was immediately obvious before examining these plots. The data points which have been labeled first are those far from where the two distributions overlap and far from the decision boundary. This is interesting because these data points are very unlikely under the generative distribution, but because they are far away from the other class, they have high confidence labels.

In Figure 8e again we see what we might expect, that the labeled points cluster around existing data points. We believe that these results bring to light an important aspect of the the self-training (and active learning) process which seems to have been overlooked in previous work. The issue is that during the training process, the distribution of the labeled data at any particular iteration does not match the actual underlying distribution of the data. This may have a large effect on final performance, depending on the learning algorithm being paired with the semi-supervised training approach and its sensitivity to this issue. In the empirical results detailed in later sections of this document, we did indeed find that final performance was extremely sensitive to the selection metric used in the self-training process. These results seemed to show that, of the metrics evaluated, a confidence-based metric often performed worse when compared to other metrics which more closely approximate the class conditional likelihood metric or Euclidean distance metric evaluated here. An examination of Figure 8d suggests that this might be because the labeled data distribution created by this metric is quite different from that of the underlying distribution. We believe a closer examination of this issue from both a theoretical and practical standpoint would be an interesting topic for future research, but it is beyond the scope of this thesis.

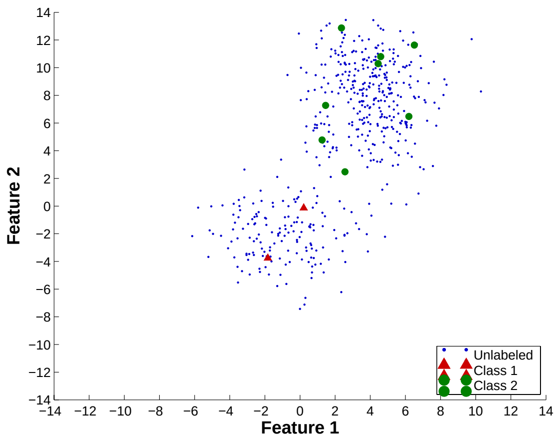
2.6 Related Considerations

2.6.1 Asymptotic Analysis

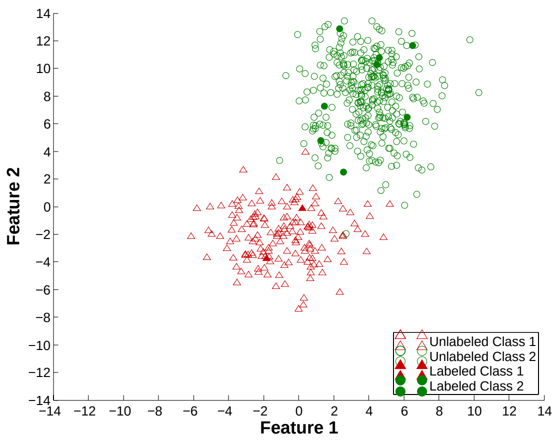
It is helpful to understand the behavior of the approaches presented here as the amounts of unlabeled data and labeled data approaches infinity. For the EM algorithm, one can look to the original work in [Dempster 77] to understand the behavior. The self-training approach has not been analyzed in detail in the literature but is close enough to the nearest neighbor approach than we can adopt the analysis in [Cover 67].

2.6.2 Are there "correct" labels?

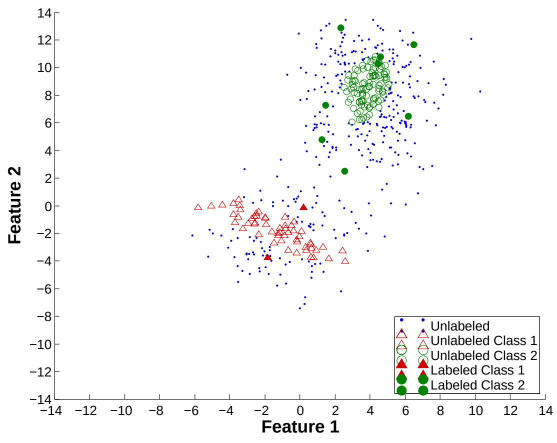
An interesting question to ask when analyzing these approaches is whether there is such a notion as "correct" labels for the unlabeled data. It is also useful to clearly define the information that can be extracted from unlabeled data and how it can be used to improve classification performance. One recent approach,



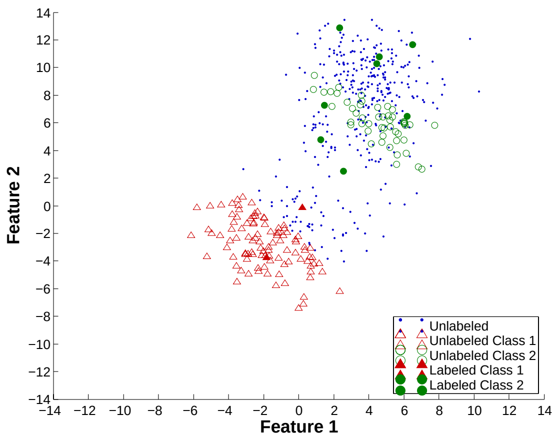
(a)



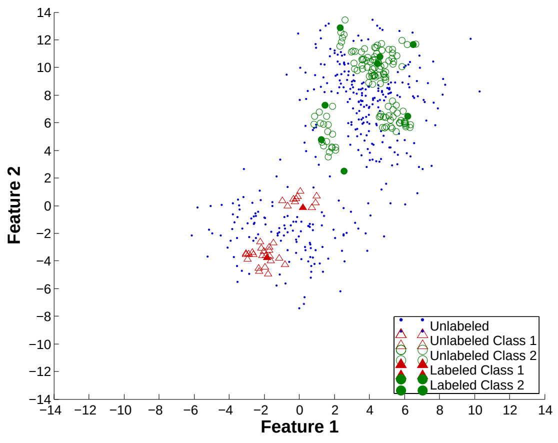
(b)



(c)



(d)



(e)

Figure 8: Plots of simulation results. Plot (a) is the original unlabeled data and labeled data. Plot (b) displays the true labels for the unlabeled data. Plots (c), (d), and (e) display the unlabeled points that are labeled by the incremental algorithm after 5 iterations. Plot (c) displays results for the class conditional likelihood metric. Plot (d) displays the results for the confidence metric. Plot (e) displays results for the Euclidean distance metric.

as introduced in [Szummer 02], makes this explicit. In that framework, the information gained from the unconditional data distribution $p(x)$ learned from the unlabeled data is used to improve the estimates of the parameters of the conditional distribution $p(y \mid x)$. This is the heart of the unlabeled data problem; however, there is no universally "correct" answer for all problems. (Correct could be defined as an estimate which minimizes some measure of error or loss.) Obviously, the ultimate goal is a method which would allow us to use the unlabeled data to reduce our expected error on the test set. In [Szummer 02], the information extracted from the unlabeled data is in the form of a regularizing prior. This prior captures the notion that it is preferable for classifier boundaries to pass through low density regions of the input feature space, and that the measurement of the density of those regions can be improved using unlabeled data. It is interesting to note that in transductive approaches, such as [Joachims 99], the preference for using a classifier decision boundary to create a large margin when the unlabeled data is taken into account, results in a very similar prior. The self-training approach which we take here is slightly different in that the asymptotic boundaries between classes will be influenced by inflection points in the distribution of the unlabeled data. Like so many other machine learning problems, there is no correct answer, "no free lunch" from unlabeled data. The successful utilization of unlabeled data only comes from the outside knowledge which we inject into the problem, but we can hope to find techniques that require as little of this knowledge as necessary and which are broadly applicable in our application domain.

2.6.3 Classifier and Unlabeled Data Labeling

One important point to clarify is that the final output of the approaches explored in this work is a classifier (or the associated generative models) for classifying future data points. An alternate approach, which we chose not to explore here, is to use the semi-supervised training process as a means of classifying data points. In this approach, the new unseen point is inserted into the training process as a new unlabeled or weakly labeled data point. One potential advantage of this approach may be improved classification performance. One major disadvantage of this approach may be computational complexity because re-running the semi-supervised training process is typically more computationally complex than running the associated classifier generated by the training process.

2.7 Key Issues

The practical application of the semi-supervised approaches described in this section is not a straightforward application of the techniques detailed in the previous sections of this chapter because of the complex nature of real world image data. In this thesis we will address and provide insight into specific key issues which are fundamental to the semi-supervised training of object detection systems. These issues were identified in the description of the approach above in this section:

Data Set Size The issue here is how final detector performance is affected both by the fully labeled and the weakly labeled training set size. Obviously semi-supervised training will have no effect if the detector already has sufficient fully labeled data to estimate its parameters. However, one would also assume that too little fully labeled data can also be problematic for semi-supervised training approaches. We explore this issue in the context of a filter based detector in Subsection 4.11.3 and in the context of the Schneiderman detector in Subsection 5.8.2.

Incremental vs. Batch Training As described in Sections 2.3.3 and 2.3.4, there are two natural ways of merging the labeled and unlabeled data - batch and incremental. The issue here is whether one technique has a performance advantage over the other, under what conditions that might be the case and why. Again this is not obvious given the nature of these techniques; the important issue is their interaction with the specific data distributions associated with the object detection problem and specific object detection approaches. We explore this issue in the context of a filter based detector in Subsections 4.11.4 and 4.11.6 and 4.11.7.

Selection Metric The issue here is what metric should be used when deciding which data points to select to add to the training set during incremental training as introduced in Sections 2.2.4 and 2.3.4. The selection of the correct metric has a very large effect on final performance. Our simulation results, described in a later section in this chapter provide some insight, but again given the nature of the object detection problem, empirical studies are necessary. We explore this issue in the context of a filter based detector in Subsections 4.11.6 and 4.11.7; and in the context of the Schneiderman detector in Subsections 5.8.3 and 5.8.4.

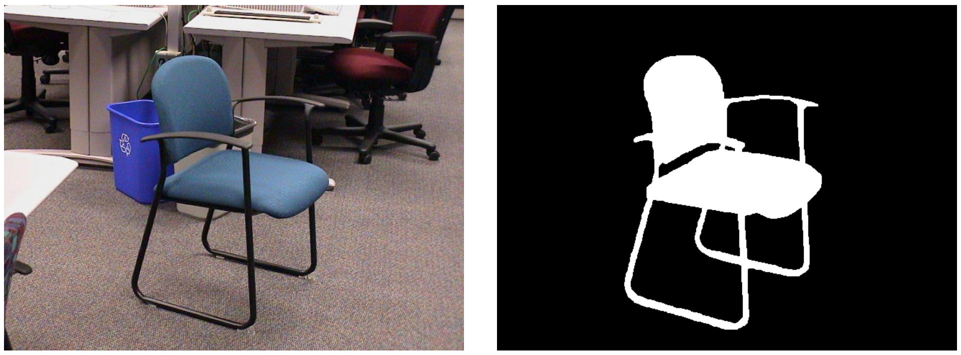

(a) (b)

Figure 9: Subfigure (a) is an example training image for the chair model. Figure (b) is the corresponding manually-generated mask image, which is used to label the individual pixels in the training image.

3 Color Based Detector Experiments

3.1 Overview

The experiments described in this section utilize a simple color based detection approach. These experiments were designed to explore various concepts related to the goals of this thesis in the context of a relatively simple detector framework.

The data set used in these experiments is a particular subset of a set of images collected at Carnegie Mellon University by a fellow graduate student Sanjiv Kumar. The full data set consists of office objects taken indoors on cluttered backgrounds, 39 images of each object, of the following objects: a chair, a coffee mug, a multimeter, a desktop phone, a remote control, a sneaker, a paint spray can, and a tape dispenser. In the experiments described here, the images of the "chair" object were used because the back and seat of the chair has a single color. The chair images were manually labeled and used as the basis of these preliminary experiments. An example of a positive training image for the chair class and its associated image mask (pixel labeling) can be seen in Figure 9.

The object images used in these experiments were collected in three groups. The first group contains the object imaged in various poses on a neutral, relatively uncluttered background, with no other objects in the scene. This set of images is the "no clutter" group. The second group contains the object imaged in various poses with other objects in the scene. This set of images is the "clutter" group. The third group, called the "occluded" set, contains the object imaged in various poses with other objects in the scene, and the object is partially occluded by other objects in the scene. All three sets of images were taken in the same office environment and so differences in lighting conditions are minimal. In all three sets, the various poses of the objects were captured by walking around the object at a fixed distance and capturing the image at regularly spaced intervals so scale variation of the object is minimal. In all of these experiments, the training set of positive examples consisted either of the "no clutter" or the "clutter" set, for a total of 13 images in each case. The images in the test set, used to evaluate performance, were the 13 images in the "occluded" set. The negative training examples in these experiments were all of the "clutter" images for all of the objects, a total of 96 images.

3.2 Color Based Object Detection Model and Learning from Weakly Labeled Data

The goal of this set of experiments was to empirically evaluate the usefulness of color as the feature set in an object detection system and to examine how weakly labeled data could be used in the context of training this model. In general, color constancy across multiple images can be a significant issue that affects detection performance. We do not explicitly address this issue here, although it was addressed in our earlier work [Rosenberg 00,01,03].

In these experiments, the images were broken up into blocks of 4×4 pixels in size and the average RGB value of those blocks was computed. That RGB triplet was then converted into a hue, saturation, value (HSV) triplet. Only the hue and saturation values were used, so block brightness was discarded. The hue angle and saturation value were converted from polar to Cartesian coordinates in the range of $+1$ to -1 and these values were used as the features in the object detection system. The result was a two dimensional continuous feature space.

Two different generative models were evaluated in these experiments. In the first model, the color space was divided up into 16 bins in each dimension, for a total of 256 bins. A multinomial model was used as the generative model to compute the class conditional probability that a specific color would be observed given a specific class: $P(D \mid l_d = c_i, q_{ij})$, where q_{ij} is the probability that the color mapped to bin j would be observed by data generated by class c_i at pixel location i, which is either object or clutter. In this approach, the image data D is broken up into blocks d_k, where $k = 1 \ldots n_d$. The model assumes that each block is generated and labeled independently, so the generative distribution for each block becomes: $P(d_k \mid l_k = c_i, q_{ij})$. The maximum likelihood labeling for a block can be computed as follows:

$$\arg\max_i P(d_k \mid l_k = c_i, q_{ij}) P(c_i)$$

The maximum likelihood parameters of the multinomial distribution can be computed in a straightforward manner from a histogram of the fully labeled training set counts, where N_{ij} is equal to one plus the count of the number of blocks labeled class i and with color j:

$$\widehat{q_{ij}} = \frac{N_{ij}}{\sum_{j=1}^{n_j} N_{ij}}$$

As is commonly done, a count of 1 was added to all of the histogram counts collected from the training data in order to implement a smoothing prior, which allows us to gracefully handle test set colors mapped to bins that were not observed in the training set.

The second model was a Gaussian mixture model with two full covariance Gaussian mixture components. One component was used to model the distribution of colors in the object class, and another was used to model the distribution of colors in the clutter class. The Gaussians were estimated in the two dimensional color space described previously. No discretization was necessary because Gaussians are a continuous model. In this model, pixel blocks were assumed to be generated independently from one another. The parameters for the generative model for a block is are μ_i and Σ_i, which are the parameters of the Gaussian model for class c_i, and F is the two dimensional feature vector (color space coordinates) for example D:

$$P(D \mid l_d = c_i, \theta_i) = \left((2\pi)^{dim} |\Sigma_i|\right)^{-\frac{1}{2}} \exp\left(-\tfrac{1}{2}(F - \mu_i)' \Sigma_i^{-1} (F - \mu_i)\right)$$

The maximum likelihood parameters of the multinomial distribution can be computed in a straightforward manner from the fully labeled training set counts, where N_i = the count of the number of blocks labeled with class i, and F_k is the color space feature vector for training set example s_k:

$$\widehat{\mu_i} = \frac{1}{N_i} \sum_{k=1}^{n_s} F_k \delta(l_k = c_i)$$

$$\widehat{\Sigma_i} = \frac{1}{N_i} \sum_{k=1}^{n_s} \left[\begin{array}{cc} (F_{k1} - \widehat{\mu_{i1}})^2 & (F_{k1} - \widehat{\mu_{i1}})(F_{k2} - \widehat{\mu_{i2}}) \\ (F_{k1} - \widehat{\mu_{i1}})(F_{k2} - \widehat{\mu_{i2}}) & (F_{k2} - \widehat{\mu_{i2}})^2 \end{array} \right] \delta(l_k = c_i)$$

3.3 Performance Evaluation

The performance of the algorithms was evaluated on the “occluded” set using the manually generated image mask for each image as the ground truth. Each pixel in the image was labeled by the algorithm, and that labeling was compared to the manual labeling of the image pixels in the image mask. Given that information, accuracy, precision, recall, true positive rate, and false positive rate were collected for each image in the test set. We use the standard definitions for precision and recall, respectively: the fraction of pixels that are labeled positive that are truly positive and the fraction of pixels that are truly positive that have been labeled positive. True positive rate is the fraction of positive pixels correctly labeled as such. False positive rate is the fraction of pixels labeled as positive that were actually members of the negative class. We found that computing and comparing all of these values allowed us to better understand the performance trade-offs for a particular algorithm. Also, since this is a classification task, these measurements were collected at a number of classification thresholds to better characterize these results. As is commonly done, this information has been plotted as a precision-recall curve, where positive precision in plotted on the y-axis and positive recall is plotted on the x-axis. We also computed the precision-recall break-even point, where precision and recall were equal. Because of the limited size of the test set and the performance characteristics of the algorithms evaluated, there was not always a single point where precision and recall were equal. To handle this occurrence, the precision-recall break-even point was computed as the average of the precision and recall when the absolute difference between precision and recall was at a minimum. We also generated an ROC curve which plots the true positive rate versus the false positive rate.

3.4 Experimental Results

Table 3 details the results of these experiments. The performance of the models are reported in two ways. The first is as the best average accuracy achievable over the training set using a single threshold. The second is as the precision-recall break-even point. Both the histogram-based and Gaussian models were evaluated. All models were trained separately on the “clutter” and “no clutter” data sets. Also, varying mixes of fully labeled and weakly labeled data were utilized, as detailed in Table 3. It is interesting to note that, as the fraction of weakly labeled data increases, the performance smoothly improves toward the fully labeled case. There are two notable exceptions. The first is the high performance of the model trained with only a single training example. I believe that this is caused by the fact that this object is predominantly a single color and therefore quite a bit of the model can be learned with just a single example. The second exception is the low performance of the models trained on the “no clutter” data and most weakly labeled data. This performance is much worse than that of the corresponding models trained with the “clutter” data. We believe this to be because, in the case of the “no clutter” training examples, a large proportion of the image is a single background color and therefore this color is assigned to the object class which results in a model with bad

Training Set	Fully Labeled	Weakly Labeled	Histogram Best Acc	Histogram PR BE	Gaussian Best Acc	Gaussian PR BE
no clutter	0	13	0.910	0.003	0.898	0.022
no clutter	1	12	0.923	0.003	0.898	0.022
no clutter	4	9	0.934	0.395	0.898	0.241 [r1]
no clutter	8	5	0.953	0.507	0.949	0.750 [r2]
no clutter	13	0	0.963	0.770	0.957	0.787 [r3]
no clutter	1	0	0.957	0.662	0.948	0.728
clutter	0	13	0.941	0.386	0.905	0.408
clutter	1	12	0.941	0.386	0.925	0.459
clutter	4	9	0.948	0.435	0.952	0.726
clutter	8	5	0.955	0.498	0.959	0.766
clutter	13	0	0.963	0.789	0.957	0.790
clutter	1	0	0.957	0.765	0.957	0.790

Table 3: This table summarizes the performance of two of the color object detection systems, one based on color multinomial (histogram) and the second on a single Gaussian per class. Different mixes of fully and weakly labeled data were evaluated. Performance is reported as the best accuracy over all possible thresholds and the precision-recall break-even point. The precision-recall data in the cells labeled [r1], [r2], [r3] are plotted in Figure 10.

performance. Precision-recall curves and ROC curves are plotted for a subset of this data in Figure 10. As this graph shows, the model trained with only fully labeled data outperforms models trained with both fully and weakly labeled data.

In the second set of experiments, a mix of fully labeled and weakly labeled data was used. In the case of weakly labeled data, all data in the data set were assigned a fixed distribution over class labels. Both the multinomial and Gaussian models were trained in this manner. In the case of the Gaussian Model, the parameters were estimated using EM as described in a previous section. A fixed number of EM iterations, specifically 20, were executed. In the cases evaluated, it appeared that model parameters had converged after that number of iterations.

Table 4 details the results of these experiments. The performance of the models is reported in the same two ways as the first set of experiments, the best average accuracy achievable over the training set using a single threshold and the precision-recall break-even point. All models were trained separately on the “clutter” and “no clutter” data sets and varying mixes of fully labeled and weakly labeled data were utilized, as detailed in Table 4. In this set of experiments EM was limited to re-estimating the probability of class membership only for the positive examples in the weakly labeled data; the negative examples in the weakly labeled set and all of the fully labeled data were fixed with its given class. A label distribution was assigned to the positive examples to utilize the fully labeled data. Image blocks in weakly labeled examples in the positive class were assigned a 0.20 probability of being in the positive class, and a 0.80 probability of being in the negative class. It is interesting to note that, in most cases, the performance stayed the same or slightly improved after twenty iterations of EM. The most notable exceptions are the cases trained on the “no clutter” data set where no or only one fully labeled training example was used. In these cases, the initial model was quite poor and EM was not able to improve upon it. Another interesting set of results is that the performance of the “clutter” models trained with 4 or 8 fully labeled examples, and the “no clutter” model trained with 8

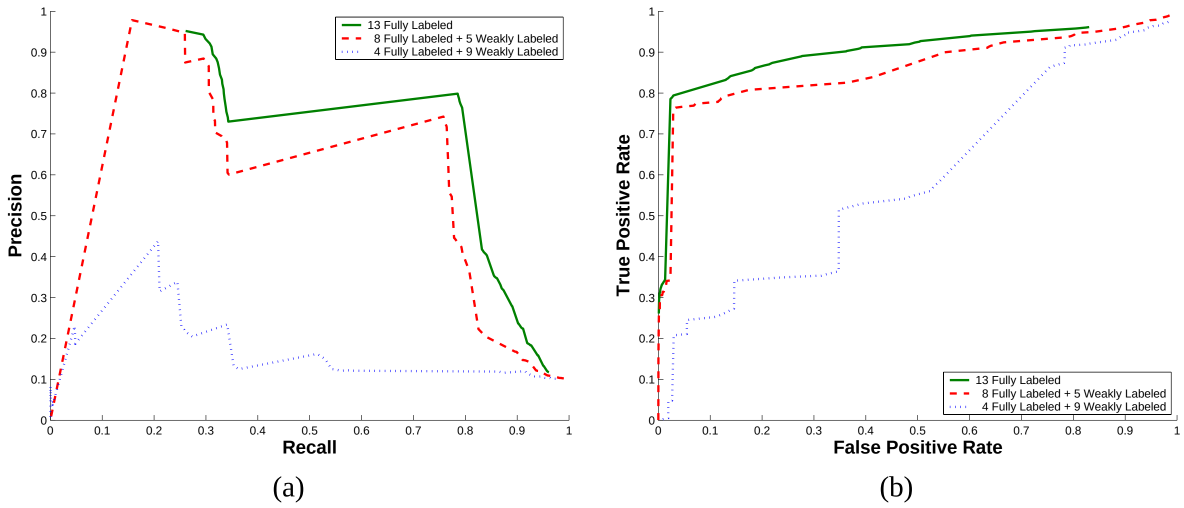


Figure 10: Subfigure (a) is a precision-recall plot of a subset of the Gaussian model results obtained with the “no clutter” training set detailed in Table 3, and (b) is an ROC plot of the results. The blue dotted line plots the data for the model detailed in table cell [r1], trained with 4 fully labeled and 9 weakly labeled data items. The red dashed line plots the results for the model of cell [r2], trained with 8 fully labeled and 5 weakly labeled data items. The green solid line plots the results for the model of cell [r3], trained only with 13 fully labeled data items.

fully labeled examples, is equal to that of a model trained with the complete set of fully labeled data. (The improvement is not statistically significant.)

In a second set of experiments utilizing EM, the fully labeled training data remained fixed, but EM was allowed to re-estimate class membership for all of the weakly labeled data. The image blocks in weakly labeled examples in the positive class were assigned a 0.20 probability of being in the positive class, and a 0.80 probability of being in the negative class. The image blocks in the negative weakly labeled examples were given a 0.999 probability of being in the positive class and a 0.001 probability of being in the negative class. The results of these experiments are detailed in Table 5. Performance in these experiments was basically identical to that achieved in the experiments detailed in Table 4. A subset of these results is plotted in Figure 11. In the precision-recall curve in this figure, one can see that, after 20 iterations, the EM model has improved the performance of the detector in the region we are potentially most interested in, where both precision and recall are near their break-even point. Interestingly, performance even rises above the fully labeled case. One possible explanation for this might be the removal of outliers in the training set. The ROC curve performance for iteration 20 is very close to that of iteration 0. To better understand the relative performance of the models at high true positive, low false positive rates, we have plotted that portion of the ROC curve on a set of expanded axes in Figure 12. One can see that, similar to the performance observed in the precision-recall plot, the model at iteration 20 outperforms the other two models in the plot and can achieve close to twice the true positive rate of the other two models at a false positive rate near 1.5%.

It is also interesting to directly examine the models constructed by the system. Figure 13 plots the final model after twenty EM iterations trained on the “no clutter” data set, with eight fully labeled and five weakly labeled examples, as detailed in Table 5. The color space is discretized for clarity, and the area of each block is proportional to its probability of class membership. Also plotted in this figure are the odds ratios for particular colors in the color space; the blue color of the chair has clearly been assigned to the object class.

Training Set	Fully Labeled	Weakly Labeled	Initial Best Acc	Initial PR BE	Final Best Acc	Final PR BE
no clutter	0	13	0.898	0.022	0.898	0.009
no clutter	1	12	0.898	0.018	0.898	0.009
no clutter	4	9	0.946	0.740	0.929	0.365
no clutter	8	5	0.960	0.783	0.963	0.794
no clutter	13	0	0.957	0.787	0.957	0.787
no clutter	1	0	0.948	0.728	0.948	0.728
clutter	0	13	0.904	0.408	0.935	0.496
clutter	1	12	0.953	0.708	0.939	0.492
clutter	4	9	0.960	0.785	0.960	0.794
clutter	8	5	0.962	0.785	0.964	0.797
clutter	13	0	0.957	0.790	0.957	0.790
clutter	1	0	0.957	0.790	0.957	0.790

Table 4: This table summarizes the performance of a color object detection system based on a single Gaussian per class. Different mixes of fully and weakly labeled data were evaluated. Initial performance is reported before any EM iterations occurred, and the final performance is reported after 20 EM iterations. The EM algorithm re-estimated the likelihood of weakly labeled positive examples; all other examples were fixed. Two performance measures are reported, the best accuracy over all possible thresholds and the precision-recall break-even point.

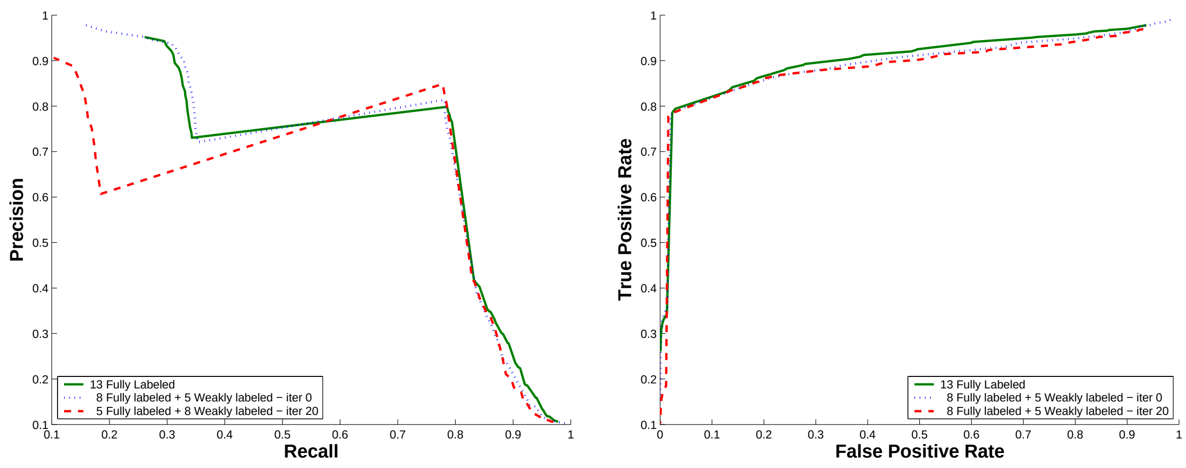
One can see that after 20 iterations, the algorithm has succeeded in making the color distribution for the object more compact and hence no longer incorrectly classifies some non-object colors as object. Figure 14 shows the results of running this classifier on two example images from the test set at EM iteration 0 and 20. By comparing subfigures (a) and (c) to (b) and (d) respectively, one can see that after 20 iterations the algorithm has succeeded in removing many incorrect classifications in the background clutter. In terms of overall performance, the model also seems to be doing the correct thing; the blue of the chair has been assigned to the object class. Interestingly, the blue of the recycling bin has been classified as background, but the model is less certain of that classification. It is also useful to compare the distribution of the object class to the clutter class. The object class completely overlaps the clutter class, though the distribution of colors for the object is less spread out and hence has a higher density in certain parts of the color space. We believe that the general relationship and the high degree of overlap between the distributions of colors for the object class and the clutter class are typical of the feature distributions of the object detection problem in general. This is because of the nature of the clutter class, which spans the space of all possible objects except the object of interest. Of course, a typical detection problem is much higher dimensional, but the two dimensional nature of the features used here facilitates visualizing this relationship.

3.5 Conclusions

The goal of the experiments described in this section was to explore the use of weakly labeled data in the context of a basic detection problem using a basic detector. We detailed how our semi-supervised training approach would be used in this framework and evaluated its performance. In the domain of this relatively simple detection problem, a small but measurable performance improvement can be achieved with the use

Training Set	Fully Labeled	Loosely Labeled	Initial Best Acc	Initial PR BE	Final Best Acc	Final PR BE
no clutter	0	13	0.898	0.018	0.898	0.009
no clutter	1	12	0.898	0.018	0.898	0.009
no clutter	4	9	0.946	0.740	0.929	0.365
no clutter	8	5	0.960	0.783 [r4]	0.962	0.799 [r5]
no clutter	13	0	0.957	0.787 [r6]	0.957	0.787
no clutter	1	0	0.948	0.728	0.948	0.728
clutter	0	13	0.905	0.458	0.934	0.496
clutter	1	12	0.953	0.730	0.926	0.273
clutter	4	9	0.960	0.785	0.961	0.795
clutter	8	5	0.962	0.793	0.963	0.794
clutter	13	0	0.957	0.790	0.957	0.790
clutter	1	0	0.957	0.790	0.957	0.790

Table 5: This table summarizes the performance of a number of color object detection system based on a single Gaussian per class. Different mixes of fully and weakly labeled data were evaluated. Initial performance is reported before any EM iterations occurred, and the final performance is reported after 20 EM iterations. The EM algorithm re-estimated the likelihood of both positive and negative weakly labeled examples; all fully labeled examples were fixed. Two performance measures are reported, the best accuracy over all possible thresholds and the precision-recall break-even point. The precision-recall data in the cells labeled [r4], [r5], [r6] are plotted in figure 11.



(a) (b)

Figure 11: Subfigure (a) is a precision - recall plot of a subset of the Gaussian model results obtained with the “no clutter” training set detailed in table 5, and (b) is an ROC plot of the results. The green solid line plots the data for the model detailed in table cell [r6], trained only with 13 fully labeled data items. The blue dotted line plots the data for the model detailed in table cell [r4], trained with 8 fully labeled and 5 weakly labeled data items at EM iteration zero. The red dashed line plots the results for the model of cell [r5], trained on the same data after 20 iterations of EM.

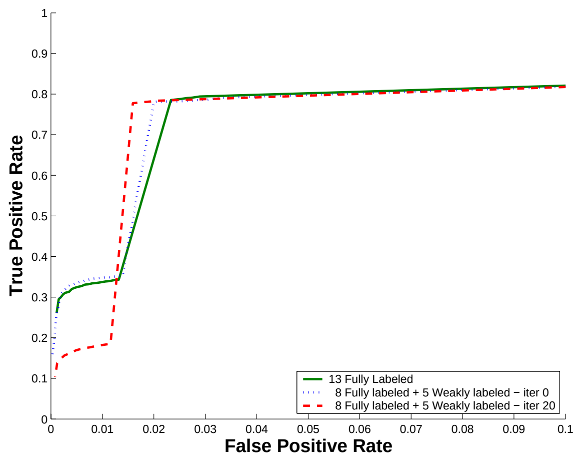


Figure 12: This is a “zoomed in” version of the ROC plot in Figure 11 to expose detail. The green solid line plots the data for the model detailed in table cell (c), trained only with 13 fully labeled data items. The blue dotted line plots the data for the model detailed in table cell (a), trained with 8 fully labeled and 5 weakly labeled data items at EM iteration zero. The red dashed line plots the results for the model of cell (b), trained on the same data after 20 iterations of EM.

of weakly labeled data. In subsequent sections we will explore variations of our semi-supervised training approach applied to more complex detection problems and more complex detectors.

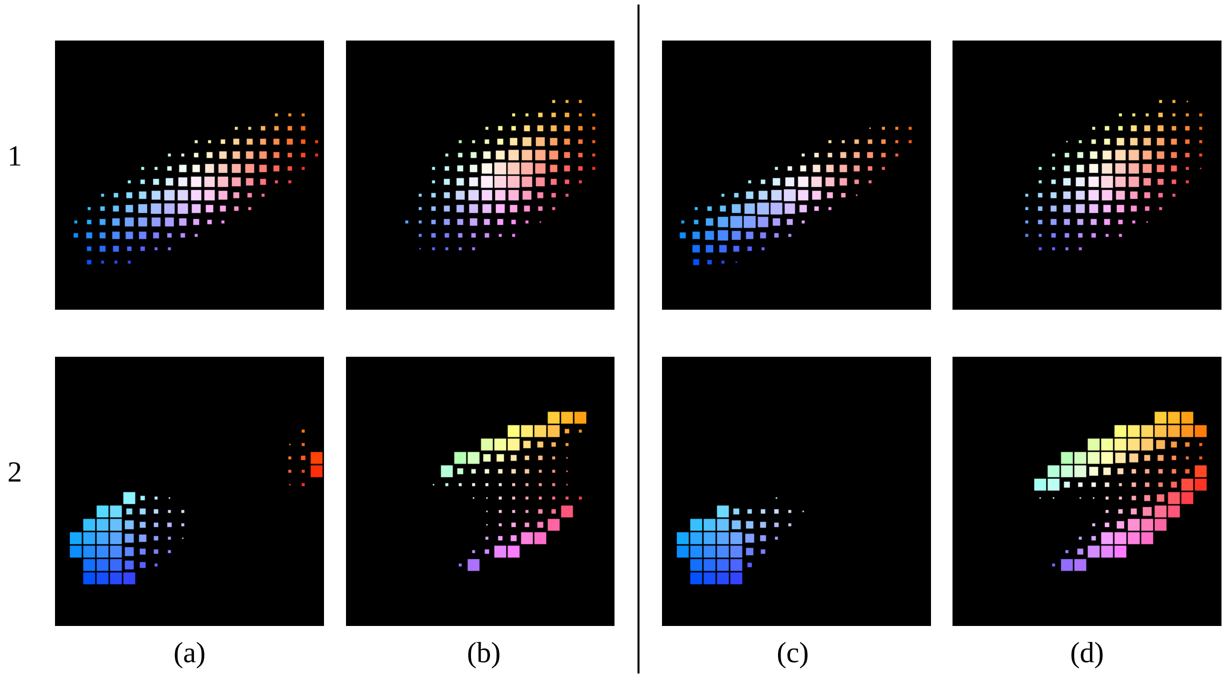
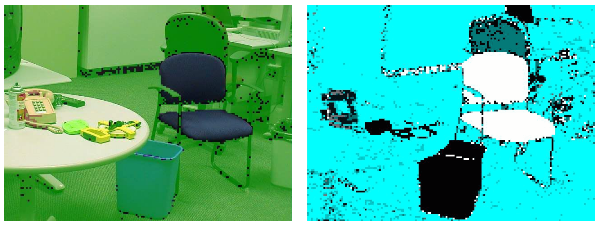
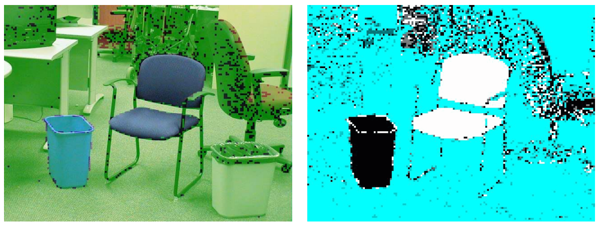
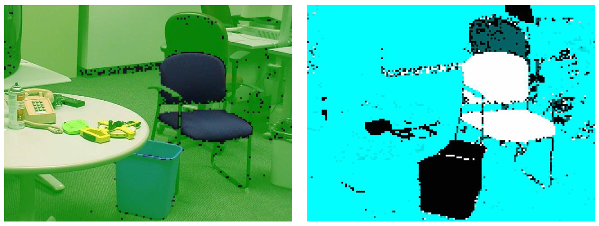


Figure 13: Each subfigure plots the HSV color space in polar coordinates, where saturation increases with distance from the center of the plot and hue is plotted as angle. The color space is discretized to visualize the distribution value for that region of the color space, where the area of each square is in proportion to the magnitude of the value. The data in this figure is for the models whose performance is detailed in Figure 11. Columns (a) and (b) plot the models for EM iteration 0, for the include and exclude class respectively. Columns (c) and (d) plot the models for EM iteration 20. Row 1 plots the generative color distributions and row 2 plots the odds ratio, a visualization of the discriminative model resulting from the generative models. (Note that this is a color figure.)


(a)


(c)


(b)

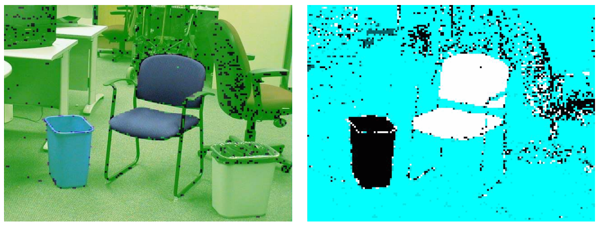

(d)

Figure 14: This figure plots the classification of a test image resulting from the model in Figure 13. Subfigures (a) and (c) present examples of the model performance at EM iteration 0. The original image is presented on the left, with pixels which are not part of the object tinted light green and on the right is a graphic representation of odds ratios of those parts of the image. Pixel blocks colored white are most likely to be object, pixels colored light cyan are most likely to be background. Indeterminate pixels are colored with darker shades or black. Subfigures (b) and (d) plot the results for the same images at EM iteration 20. Notice how the amount of misclassified background clutter is reduced.

4 Filter Based Detection Experiments

4.1 Introduction

A new object detection system was developed for the experiments in this section. The goal of the design was to develop an appearance-based detector based on a statistical model which would facilitate the exploration of the use of weakly labeled data, as well as exhibit good detection performance; however, the detector need not exhibit state of the art detection performance such as the detectors described in [Moghaddam 97, Rowley 98, Schiele 00, Viola 01, Schneiderman 03, Schneiderman 04]. The focus was to design a detector which was relatively quick in training, and which has a structure that was simple and easy to analyze.

The detector that was developed is similar to [Schiele 00]. The features of this detector consist of the outputs of a group of filters applied to a grayscale version of the image. Also, feature vector likelihoods are aggregated over the object without regard to position. Note that local structure can still be captured by such a set of features because the feature vector at each pixel location consists of the output of many filters at different scales computed at that location. This means potentially that, relatively large structures can be captured, since each filter has large support. The disadvantage is that the size of the structures whose shape can be captured is limited by the size of the filters used to generate the feature vector. One advantage of this approach is that the feature vector is simple to compute (of course this depends on the size of the support of the filters used). Another advantage is that each training image provides many example feature vectors for training because the model does not vary with the location of the feature vector on the object.

One major difference between the model used in this work and that of [Schiele 00] is that the work described here uses a mixture of Gaussians as the generative model to capture the feature vector statistics and [Schiele 00] uses a histogram. A mixture of Gaussians was chosen because it is more amenable for use with larger (higher dimensional) feature vectors. Another major difference is the introduction of what we call a “spatial model” as a principled means of aggregating the model statistics across the object. The “spatial model” used here takes into account the general shape of the object itself and therefore potentially can provide additional information beyond simple response aggregation over rectangular regions.

In the following sections, we detail the detector structure, the semi-supervised training approach applied to this detector, and the results of a set of experiments exploring different aspects of semi-supervised training on a real world data set.

4.2 Detector Details

Our object detection framework is based on a generative image model which captures the statistics of a feature vector computed at each pixel location. The features in our system are the outputs from a set of oriented separable Gaussian derivative filters constructed using the filter design introduced by [Freeman 91]. An overview of this system is presented in Figure 15. Specifically, we utilized four filter scales and five filters at each scale which included the first and second derivatives, computed according to the following formulation in [Freeman 91] and plotted in Figure 16:

$$G_{2ax} = (2x^2 - 1) \times \exp(-x^2),\ G_{2ay} = \exp(-y^2)$$
$$G_{2bx} = x \times \exp(-x^2),\ G_{2by} = y \times \exp(-y^2)$$
$$G_{2cx} = \exp(-x^2),\ G_{2cy} = (2y^2 - 1) \times \exp(-y^2)$$

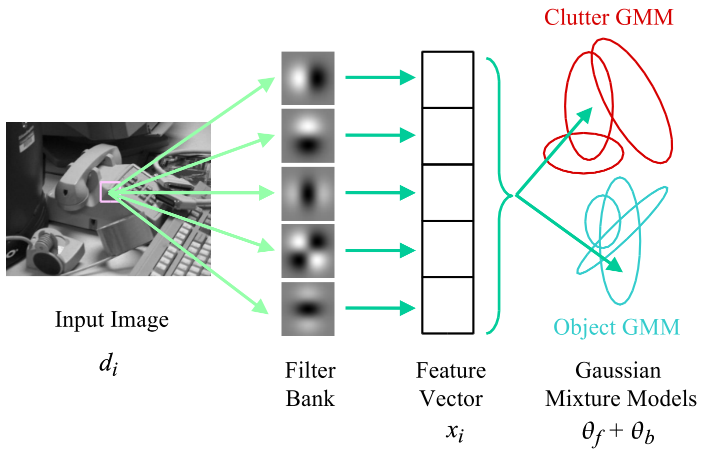


Figure 15: An schematic overview of the filter based detector. Input pixels are passed through a filter bank to generate feature vectors which are then evaluated under a pair of Gaussian mixture models.

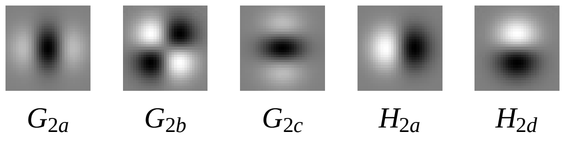


Figure 16: Plots of the five filter kernels used as features for the generative model. Positive values are indicated as light colored pixels, negative values as black, and zero as gray.

$$H_{2ax} = (x^3 - 2.254x) \times \exp(-x^2), H_{2ay} = \exp(-y^2)$$
$$H_{2dx} = \exp(-x^2), H_{2dy} = (y^3 - 2.254y) \times \exp(-y^2)$$

The following kernel sizes were used: 11×11, 17×17, 23×23, and 33×33, for a total of 20 filter values at each pixel location.

In our generative model, the distribution of values over this twenty dimensional vector of filter outputs is modeled using a mixture of full covariance Gaussians. In the experiments detailed here, we use one mixture model with 40 components to capture the filter output statistics for the object to be detected, and another mixture model with 20 components to model the clutter class. (Empirically we found that more mixture components were needed to accurately model the object distribution.) Even though each object class is represented with a different mixture model, the feature vector statistics are modeled with a single mixture of Gaussians within each object class, irrespective of their relative location on the object of interest. As described previously, this is similar to the model described in [Schiele 00], but in that work, the authors utilize a histogram-based model which typically requires a larger number of parameters to handle high dimensional feature spaces.

This type of model is appealing in its simplicity; however, one issue is how to map from filter response probabilities at individual pixel locations to the desired quantity which is the probability of the object of interest being present at a particular location in the image. One approach is to combine the probabilities from nearby pixels into a single quantity; a principled method for achieving this is described in the following section.

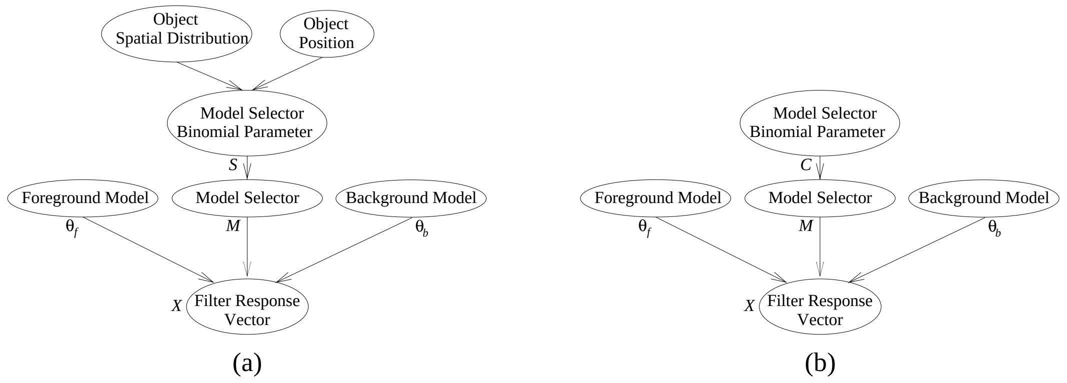


Figure 17: Diagram (a) depicts a graphical model, which corresponds to the generative model of the spatial distribution of filter responses and its associated dependencies for an image containing an object. Diagram (b) depicts the generative model for an image that contains only clutter.

4.3 Model of the Spatial Distribution of Filter Responses

To move from a pixel-based generative model to an object-based one, we utilized the generative model, depicted as a graphical model in Figure 17. Figure 17a shows the model for an image containing an object with background clutter and Figure 17b shows the model for an image containing solely clutter. Note that this model can be generalized to handle multiple objects.

The selection of one of the two models in Figure 17 determines whether an image will consist of both the object and clutter or just clutter alone. First we will discuss the model in Figure 17a, which depicts an image containing an object. It is first important to note the top down nature of the model. The top two nodes determine the location of the object in the image and the distribution of the features generated in the image. The *object spatial distribution* determines the probability that a feature vector will be generated from the *foreground model* distribution or the *background model* distribution at each location in the image. Note the difference between these distributions. The foreground and background models are the location independent distributions described in the previous section. The foreground model captures the distribution of the feature vectors that are only generated by the object, and the background model captures the distribution of feature vectors that are only generated by the background. As described previously, the foreground and background models are each composed of a separate mixture of Gaussians. The purpose of the *spatial distribution* is to capture the approximate shape and size of the object. The object's spatial distribution specifies the parameters of a probability distribution at each location in the image and the probability distribution selects whether the feature vector at that specific location in the image is generated by the foreground distribution or the background distribution. In our implementation the *spatial distribution* consists of a collection of parameters for a set of Bernoulli distributions, and each feature vector is generated independently of the others. Therefore, each Bernoulli distribution has a different set of parameters at each image location. Examples of such a model for multiple poses of a desktop telephone can be seen in Figure 18c, where lighter pixels indicate locations that are more likely to be foreground and darker pixels indicate locations that are more likely to be background. The foreground probability will be close to 1 for pixels near the center of an object. In our model, pixels outside the bounding box of the object are all modeled by the clutter class Bernoulli distribution.

The case where an image consists solely of clutter, as depicted in Figure 17b, is simpler. In this case, all of

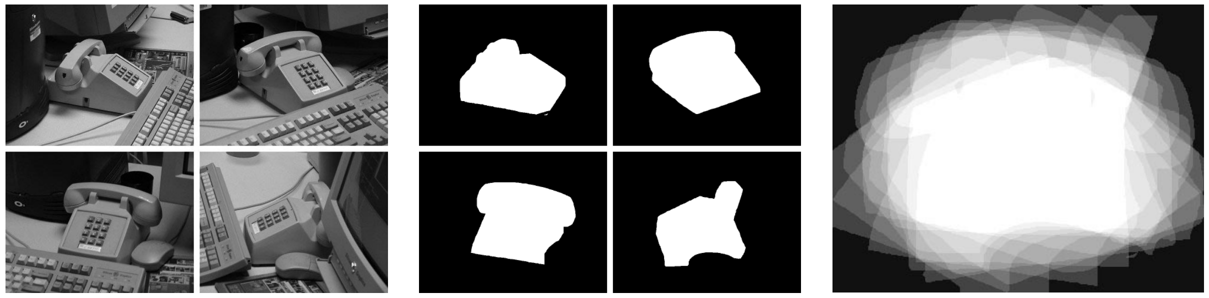

(a) (b) (c)

Figure 18: Four example training images (a) and corresponding mask images (b). Image (c) is a plot of the spatial distribution of the object viewed from a wide range of different angles. Lighter pixels indicate pixels that are more likely to be part of the phone, darker pixels indicates pixels that are less likely to be part of the phone.

the pixels in the image are simply generated by a fixed mixture of the *background model* and the *foreground model*. The foreground model is included to capture incorrect labeling of foreground pixels as background.

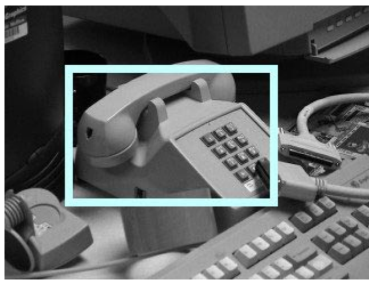
To more formally describe this model, we designate a subwindow of the image at location ℓ in the image being processed as the detection window. The pixels in the detection window are labeled D, with d_i denoting the pixel value at a specific detection window location i, where i indexes the window locations from $i = 1 \ldots n$ relative to ℓ. The feature vectors generated from D we call X, with x_i being the feature vector at location i in the image generated from pixel d_i. Our goal is to compute $P(Y \mid X, \ell)$, where $Y = object$, an image containing an object, or $Y = clutter$, an image not containing the object. (To simplify the notation we will typically omit ℓ; in practice we search over offset locations in the image and evaluate the model given a specific offset location.)

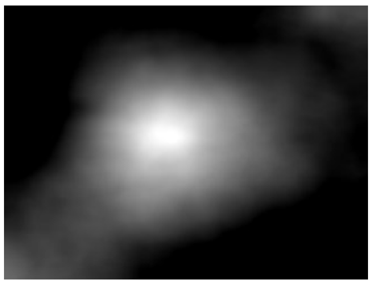
For the model that captures the presence of an object, we designate the spatial distribution over model selection probabilities when the object is present as S, the object spatial distribution. The parameter of the Bernoulli distribution at a specific image location is indicated by S_i, where i indexes image location relative to ℓ. So the probability that a particular model is selected, $M_i = m_i$, at a specific location i is $P(M_i = m_i) = S_i$, where m_i is equal either to the foreground model f or background model b. In the case of an image which solely contains clutter, C is used to signify the parameters of a fixed Bernoulli distribution over the image. We use θ_f to indicate the parameters of the mixture of Gaussians for the foreground model and θ_b the background model. Note that, as described previously, the parameters of these models do not vary with image location. Applying Bayes rule, we can compute $P(Y \mid X)$ in terms of quantities from our generative model:

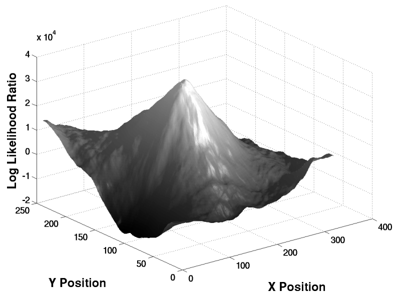
$$P(Y \mid X) = \frac{P(X \mid Y)P(Y)}{P(X)}$$

Taking $P(Y)$ as uniform, we need to find expressions for $P(X \mid Y)$ and $P(X)$. For $P(X \mid Y = object)$ we note that, given $Y = object$, the probability of the observed feature data at each image location is independent:

$$P(X \mid Y = object) = \Pi_{i=1}^{n}(P(x_i \mid \theta_f)P(M_i = f \mid S_i) + P(x_i \mid \theta_b)P(M_i = b \mid S_i))$$


(a)


(b)



(c)

Figure 19: These images are examples of detections of the desktop phone object on the test set. The (a) image is the detection, the (b) image is a grayscale plot of the log likelihood ratio, where larger values are plotted with lighter pixels, and the (c) image is a 3d plot of the same data.

For images which contain both the object and clutter, pixels outside the object bounding box are modeled identically to clutter pixels. Because of this, only pixels inside the bounding box need to be considered when computing the likelihood ratio. If we set $P(Y = object) = P(Y = clutter)$, the likelihood ratio over the entire image is:

$$\frac{P(Y=object|X)}{P(Y=clutter|X)} = \Pi_{i=1}^{n} \frac{P(x_i|\theta_f)P(M_i=f|S_i)+P(x_i|\theta_b)P(M_i=b|S_i)}{P(x_i|\theta_f)P(M_i=f|C)+P(x_i|\theta_b)P(M_i=b|C)}$$

The maximum likelihood location of the object can be found by varying the location of object spatial distribution in the image and finding the location with the maximum likelihood ratio. The value of this likelihood ratio can also be thresholded to determine the presence or absence of an object. Figure 19 shows an example detection in (a), a grayscale plot of the log likelihood ratio in (b), and a surface plot of the same data in (c).

It is also important to note that the standard ad hoc method of summing up the log likelihood value over a rectangular window, as described in [Schiele 00], is actually a special case of the technique described in this section when the spatial distribution consists of a simple rectangular window with a constant set of mixture weights over the window.

4.4 Detector Implementation Efficiency Issues

Conceptually, detecting objects using the generative model described above is relatively straightforward: the likelihood ratio at each possible x, y location in the image is computed and then thresholded to find detections. Otherwise the most likely location can be determined by locating the maximum of the likelihood ratio. This computation can be expensive because of the spatial distribution computation which is very similar to a correlation. These spatial distributions can be quite large. For example, the spatial distribution for the phone object for a 640×480 image is 405 pixels wide × 314 pixels high and encompasses over 120,000 pixels. Introducing an incremental approach to computing the desired likelihood ratio speeds up this computation. (Note also that there are other signal processing methods for speeding up this computation, such as frequency domain techniques, as for example described in [Frey 01].)

In the incremental approach, we take advantage of the fact that we scan the detection window over the image and only move it a small distance each time. This means that the region of support of the spatial

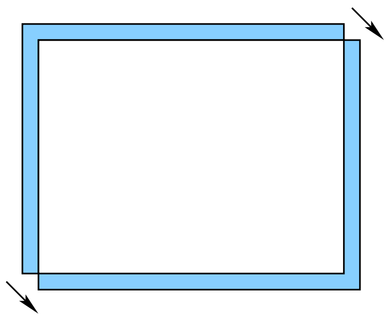


Figure 20: By only recomputing the values in the shaded region as the template is scanned over the image, a large decrease in computational complexity during detection can be realized.

distribution only changes a small amount at each step, as is illustrated in Figure 20. To compute the sum of log likelihood values over the pixel locations in the detection window, we only need to subtract the values that are leaving the region of support and add the new values which are entering the region of support. This works when the function that combines the object and clutter class likelihood values is constant over the detection window, as is the case with the simple rectangular aggregation scheme. However, in the case of the spatial distributions we have introduced, this function changes at every location within the detection window, so the incremental approach can not be used. We introduce an approximation to the notion of a spatial distribution, which allows us to utilize an incremental approach.

We approximate the parameters of the Bernoulli distributions at different locations in the spatial distribution by using only a small number of discrete values. In this way we can perform the same incremental computation described in the previous paragraph, one for each set of parameter values, with the complexity increasing linearly with the number of quantized values we use. This allows us to trade-off complexity for accuracy. In our experiments, we used only two values, a background rate and a foreground rate. A bitmap is generated for the detection window, which indicates which pixels belong to a specific quantization level. As the bitmap is moved across the image, not all of the values enter or leave the sum or product being computed; only a small number of values change, as is illustrated in Figure 20. Taking advantage of this can greatly speed up the computation.

In our experiments, when a two level quantized approximation was combined with the incremental approach, a speed up of 45 times over the naive approach was observed, reducing the run time per 640×480 image from 90 minutes to under 2 minutes.

4.5 Training the Model with Fully Labeled Data

The parameters of the probability distributions in this model can be learned from training data. When the training data is fully labeled, this can be done in a relatively straightforward manner because of the structure of the generative model, as depicted in Figure 17. The structure indicates that the parameters of S_i can be computed simply by knowing which pixels are part of the object, and which are background in the training examples. So, given an aligned mask for each training image that indicates the class membership of each pixel, it is possible to compute the parameters of the Bernoulli distribution, for each location (see Figure 18c) simply by counting the number of times that location was indicated as belonging to the object class.

Also, because of the structure of the model, once the class of a pixel is known, the probability distributions

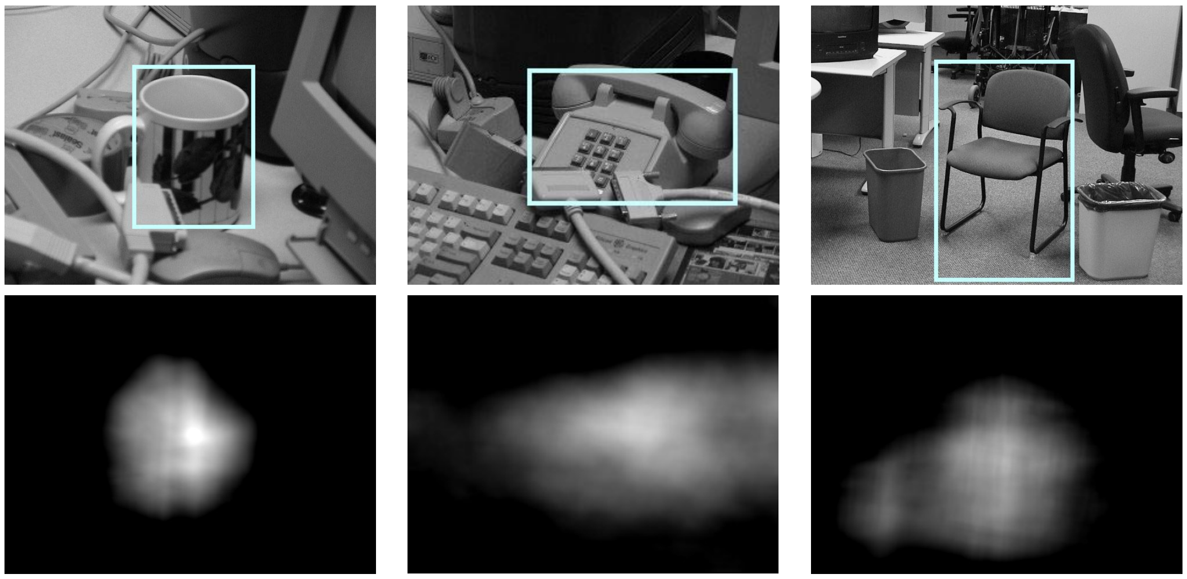


Figure 21: Example detections of the mug, phone, and chair objects using a detector trained with fully labeled data. In the top row light colored boxes have been plotted centered at the detection locations. In the bottom row the corresponding log likelihood ratio data values are plotted, brighter pixels indicate that the object is more likely to be at that location.

$P(x_i \mid \theta_f)$ and $P(x_i \mid \theta_b)$ can be computed. In our implementation, as described previously, each of these distributions is modeled by a mixture of Gaussians. Because of the structure of the model, these distributions do not vary with image location, so it is only necessary to learn a single set of parameters for each. We learn the parameters of these Gaussians in the standard manner using Expectation-Maximization, as described in [Bishop 95].

Some example detection results achieved when training this model with labeled data can be seen in Figure 19 and Figure 21.

4.6 Batch Training with Weakly Labeled Data

In our approach, weakly labeled data is weakly labeled in the sense that the object of interest is known to be in the training image, but the location of the object and which pixels correspond to the object is not known. For the work described here, we assume that the objects are all present at approximately the same scale.

We use the object detection framework detailed in the previous section and train it using EM. We first train the spatial distributions and the foreground and background models as described in the previous section with the fully labeled data subset. This serves as the starting point for our weakly labeled data approach. During training with weakly labeled data, we fix the background model and the spatial distributions.

We then train the foreground model with a combination of fully labeled and weakly labeled images using EM. In contrast to the fully labeled data case, we weight the contribution of each training example by the sufficient statistics in the expectation step according to our confidence in that data. The weight for fully labeled data is 1. For weakly labeled examples, the weight is the probability that the observed feature vector was generated by the foreground model, given the current model. Filter responses outside of the bounding box of the object spatial distribution were not included in the weakly labeled training set for the foreground model.

To compute this probability, we first use the current model to determine the most likely location of the object in each weakly labeled data example. Once the most likely location of the object in each weakly labeled examples is known, the desired probability can be computed as follows:

$$P(M_i = \theta_f \mid x_i, S_i) = \frac{P(x_i|\theta_f)P(M_i=\theta_f|S_i)}{P(x_i|\theta_f)P(M_i=\theta_f|S_i)+P(x_i|\theta_b)P(M_i=\theta_b|S_i)}$$

The complete training procedure for using a combination of weakly and fully labeled data is as follows:

1. Train the object and clutter spatial distributions and the foreground and background models using fully labeled data subset and EM.
2. Compute the most likely object location for each weakly labeled data example using the current model.
3. Execute the "E" step of EM by computing the expected values of the sufficient statistics of the fully labeled and weakly labeled object data, weighting the weakly labeled examples by $P(M_i = \theta_f \mid x_i, S_i)$.
4. Execute the "M" step of EM by updating the parameters of the foreground model using the sufficient statistics computed in step 3.
5. Repeat steps 2-4 for a fixed number of iterations or until convergence.

In this work we repeat the inner loop of this algorithm for a fixed number of iterations, typically 40.

4.7 Incremental Training with Weakly Labeled Data

4.7.1 Approach

In this experiment, weakly labeled data is weakly labeled in the sense that the object of interest is known to be in the training image, but the location of the object ℓ is not known and the pixel mask $P(M_i = f \mid x_i, S_i)$, the likelihood that a pixel corresponds to the object, is not given.

The basis of our weakly labeled data approach is the object detection framework detailed in section 4.3. The initial labeled training set is used to estimate a set of model parameters, $M^0 = \{\theta_f^0, \theta_b^0, S^0\}$. Our weakly labeled data approach adapts the foreground model θ_f, but keeps the background model θ_b and the spatial distribution S fixed.

As previously described in section 2.3.4, given the current labeled training set T, we want to incorporate the images in the weakly labeled training set W. The foreground model θ_f is adapted with EM using a combination of fully labeled T images and weakly labeled W images. Given a training set $T = L \cup W$, the contribution of the features computed at pixel i of each training example to the sufficient statistics in the expectation step is weighted according to the likelihood that the data belongs to the current foreground model θ_f, given the current model M and its most likely location in a weakly labeled data example, $\hat{\ell}$, as follows:

$$P(M_i = f \mid x_i, S_i) = \frac{P(x_i|\theta_f)P(M_i=f|S_i)}{P(x_i|\theta_f)P(M_i=f|S_i)+P(x_i|\theta_b)P(M_i=b|S_i)}$$

Since, the initial model M^0 is trained using a limited amount of data, accurately estimating the object location in a weakly labeled training image may not always be possible. These incorrect labels can potentially "corrupt" the model statistics. Approaches which immediately add all of the weakly labeled data W to the training set T are particularly susceptible to this problem. We believe that incrementally adding weakly labeled examples to the training set according to our confidence in those labels, similar to the methods described in [Blum 98, Nigam 00, Selinger 01] can reduce the impact of this issue. The order in which the images are added is critical. We want the model to first generalize to images that are most similar to the initial training set, and then incrementally extend to views that are quite different from those in the original training set.

4.7.2 Selection Metric

The selection metric we choose determines which of the current set of weakly labeled images W are added to the current labeled training set T. There are many potential methods we can use to compute this selection score value. The most straightforward is to use the value of the likelihood generated by our current model, $P(Y \mid X)$. However, this can be problematic in that our model may provide inaccurate probability values for the weakly labeled data we care the most about, the data which our current model correctly generalizes to, but the model is not confident in its detection.

Another possible method builds a new model including the candidate training example and computes a score for the new model. In a traditional cross validation approach, the candidate example would be tentatively added to the current labeled training set T and a new model would be trained. This model would be evaluated using a "hold out" data set that is set aside and not used for training. The likelihood of this data under the new model is used as the score for the new model. This is problematic in that the labeled data is assumed to be scarce so it is highly advantageous to use all of it for training. (It may be possible to use a form of leave one out cross validation to minimize this effect, but that would greatly complicate the system.) Also, we would like to be able to use the weakly labeled data to generalize to appearance changes, which are potentially quite far from our original training set. A cross validation based approach would only capture our initial viewing conditions and so would not be helpful in this case.

Another possibility is to use the training data as if it were a "hold out" set and examine its likelihood under the new model. The problem is that images which are far away from the initial training set may significantly decrease the current training set likelihood, but those images may be very useful in helping the system generalize to new views.

We propose a method that uses the labeled training examples in a novel manner similar to cross validation which we call the "reverse odds ratio." There are two differences between our method and traditional cross validation. The first is that we build a model using only the candidate example to compute its score. The second is that we compute the score by evaluating the likelihood of the examples in the current labeled training set T for which we have actual or estimated ground truth information. We take the maximum of this likelihood over the labeled training data as our similarity measure, which measures how close the candidate example is to its closest neighbor in the current training set. Note that this "distance" metric is specific to each candidate example. A schematic representation of this is presented in Figure 22. This allows for generalization to examples far away from our initial labeled training set. The advantages are that we do

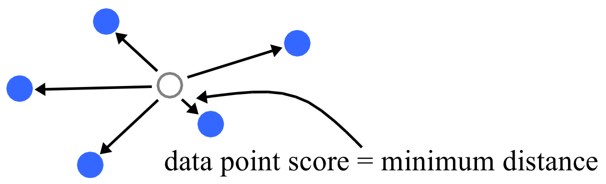


Figure 22: A schematic representation of the distance computation. The filled circles represent data points in the labeled set and the unfilled circle represents the candidate weakly labeled data point under consideration. The score assigned to it will be its minimum distance to the labeled points.

not need a separate hold out set, and that, because we only compute the score based on the nearest neighbor, the resulting model can generalize to views which are very different than the initial training set. Specifically, the steps in this method are as follows.

Given $\mathcal{M}$ and a new image W from $\mathcal{W}$; compute $C = \text{ConfidenceScore}(\mathcal{M}, W)$ as follows:

1. Compute the most likely object location $\hat{\ell}$ for the weakly labeled data example W using the current model $\mathcal{M}$.
2. Build a new foreground model θ_f^W for the weakly labeled example at its hypothesized detection location, as if it were a fully labeled example.
3. Given this new model $\mathcal{M}^W$ for each new example T_l in $\mathcal{T}$, compute the likelihood ratio $C^l = \frac{P(Y_l=object|X_l)}{P(Y_l=clutter|X_l)}$.
4. Return the maximum likelihood ratio as the score C, $C = \text{argmax}_l C^l$.

4.7.3 Incremental Training Procedure

The complete incremental training procedure for using a combination of weakly and fully labeled data is described as follows.

Initialization:

1. Train the parameters of the initial model $\mathcal{M}^0$ consisting of the object spatial distribution S^0 the foreground θ_f^0 and background θ_b^0 models using the fully labeled data subset and EM. Initialize the initial labeled training set $\mathcal{T}^0$ with the provided fully labeled data.

Beginning of Iteration j:

1. Compute the most likely object location $\hat{\ell}_k$ for each weakly labeled data example W_k using the current model, $\mathcal{M}^j$.
2. For each W_k in $\mathcal{W}^j$, compute the selection score value $C_k = \text{ConfidenceScore}(\mathcal{M}^j, W_k)$.
3. Select the weakly labeled example $W_{\hat{k}}$ where $\hat{k} = \text{argmax}_k C_k$ and its associated detection location $\hat{\ell}_{\hat{k}}$ with the highest score and update the current training set and the weakly labeled training set: $\mathcal{T}^{j+1} \leftarrow \mathcal{T}^j \cup \{W_{\hat{k}}\}$, $\mathcal{W}^{j+1} \leftarrow \mathcal{W}^j - \{W_{\hat{k}}\}$.

EM Loop:

1. Execute the "E" step of EM by computing the expected values of the sufficient statistics of the fully labeled and weakly labeled object data, weighting the weakly labeled example pixel features vectors by $P(M_i = f \mid x_i, S_i)$.
2. Execute the "M" step of EM by updating the parameters of the foreground model using the sufficient statistics computed in step to compute the new foreground model θ_f^{j+1}.
3. Repeat steps 1-2 for a fixed number of iterations, or until convergence.

End of Iteration j: While $W \neq \emptyset$, start new iteration $j+1$.

4.8 Overview

Important parts of this work include the empirical evaluation of the different conditions under which weakly labeled data might be used and different approaches to the incorporation of weakly labeled data.

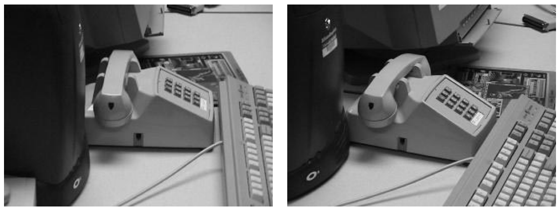
The following semi-supervised data approach variations were considered:

- standard expectation maximization
- incremental approach
- different distance metrics for incremental approaches - odds ratio, reverse odds ratio
- annealing the weight of the weakly labeled examples

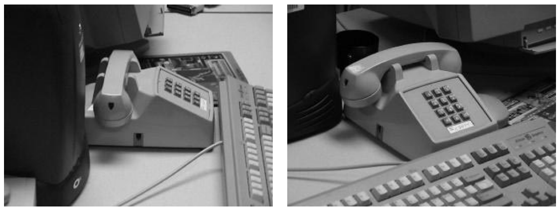
The following data set variations were considered:

- sensitivity of the set of detectors used to the amount of labeled data when labeled data are used only
- sensitivity of the set of detectors used when different mixes and and amounts of labeled and weakly labeled data is used
- effects of particular viewpoints of the labeled data
- effects of the distribution of the labeled examples

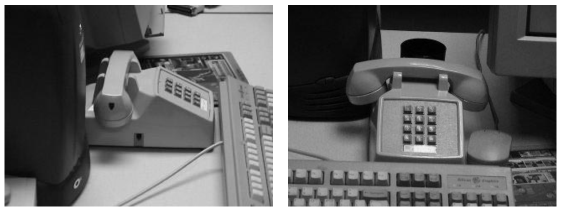
The set of experiments presented attempts to explore this joint space of parameters and its effects on object detection performance.



(a)



(b)



(c)

Figure 23: The pair of images labeled (a) are representative of a “Close Pair” of training examples, close to one another in pose. The pair of images labeled (b) are representative of a “Near Pair” of training examples, further away from one another in pose. The pair of images labeled (c) are representative of a “Far Pair” of training examples, even further away from one another in pose.

4.9 Data Description

4.9.1 Introduction

We call the data set use by the experiments detailed in this section the “telephone” data set. This data set consists of images of a desktop telephone, as can be seen in Figure 23. The telephone is positioned in a cluttered scene on a table with scale and lighting held relatively constant. The camera viewpoint is varied over the images from +/- 90 degrees of frontal with camera height held relatively constant. There are 12 training images and 12 test images in this data set. The test images are similar in pose, scale and lighting to the training images and contain additional clutter and partial occlusions of the object. Images were collected with a standard consumer quality digital camera.

In the experiments described in the following sections, a complete data set of fully labeled data was used. The partially labeled data set groups consist of varying amounts of fully labeled and weakly labeled data. So, for example, one group consists of a single labeled image in each subset and all 12 possible such subsets are generated. This allows 12 different experiments to be performed under the same conditions so that issues such as the variance of the final measured performance and sensitivity to the pose of the labeled data can be evaluated.

To aid in the explanation that follows, we assign an index to each training image from 1 to 12. The views are ordered sequentially as the viewpoint (camera) moves around the phone, from -90 degrees to +90 degrees. A single model captures all of the viewpoints. The goal of semi-supervised training will be to generalize the model from a limited range of viewpoints or a single viewpoint to other viewpoints.

4.9.2 Single Image Group

In this case, there are 12 different labeled subsets in this group, each consisting of a single labeled image. This group is designated the “single” group in the experiments.

4.9.3 Two Image Close Pair Group

There are 12 different labeled subsets in this group, each consisting of a pair of labeled images, an example of which can be seen in Figure 23a. In this group, each pair is a set of sequential images, e.g. 1 and 2, 2 and 3, 3 and 4, etc. This set was constructed to explore the scenario where the labeled images are quite similar. The labeled data will only span a small portion of the appearance space, but will cover that part of the space densely.

This group is designated as the “close pair” group in the experiments.

4.9.4 Two Image Near Pair Group

There are 12 different labeled subsets in this group, each consisting of a pair of labeled images. An example is shown in Figure 23b. In this group, each pair is a set of sequential images approximately three views apart, e.g., 1 and 4, 2 and 5, etc. This set was constructed to explore the scenario where the labeled images are somewhat similar. In this case, the labeled data will only span a portion of the appearance space slightly larger than that of “close pair” group, but will cover that part of the space less densely.

This group is designated as the “near pair” group in the experiments.

4.9.5 Two Image Far Pair Group

There are 12 different labeled subsets in this group each consisting of a pair of labeled images. In this group, each pair is a set of sequential images approximately five views apart, e.g. 1 and 6, 2 and 7, etc. This set was constructed to explore the scenario where the labeled images are quite different. In this case, the labeled data will cover small portions of the appearance space at a low density.

This group is designated as the “far pair” group in the experiments.

4.10 Experiment Details and Evaluation Metrics

The images used in these experiments were 320×240 pixels. The object occupied an area of approximately 175×110 pixels on average. It should be noted that, even though the object occupied a sizable portion of the image, the fact that object detections were permitted to occur anywhere in the image, even overlapping the images edges, made this a difficult detection problem.

Performance was evaluated on a set of test images distinct from the training set. In the test images, the objects appeared under similar lighting conditions and at a similar scale to the training set. Each test image contained exactly one instance of the object. To evaluate the performance of our algorithm, we computed the distance in pixels from the location of the true detection to the maximum likelihood location found by

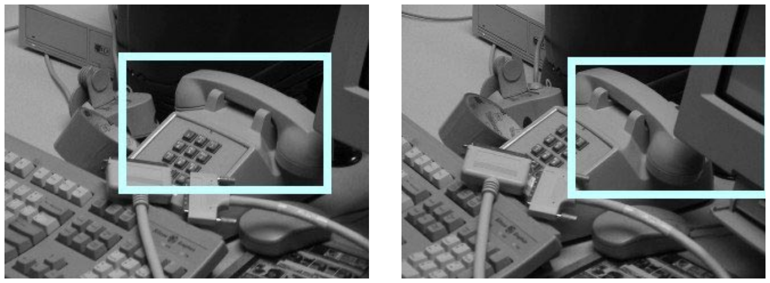


Figure 24: Examples of a correct and an incorrect detection. Detections that were within a distance of 40 pixels of the correct detection location were considered to be correct.

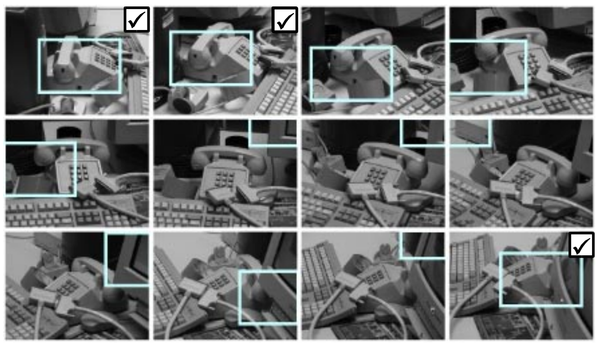
the trained model. Only detections that were within a distance of 40 pixels of the correct detection location were considered to be correct. Examples of a typical correct and an incorrect detection can be seen in Figure 24. The location of the ground truth detection was the centroid of the manually generated object mask for that specific image. Note that this evaluation metric is less stringent that other metrics which also require an accurate determination the presence and absence of an object, in addition to its location, as in [Schneiderman 03]. In contrast, our evaluation metric is more stringent than those that only require the correct determination of absence or presence, as in [Schiele 00].

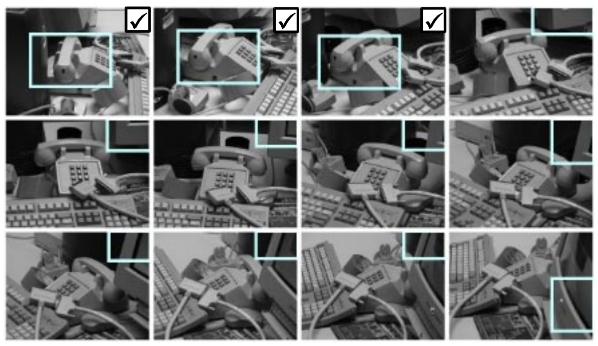
In terms of specific parameter values in these experiments, the probability of a pixel being generated from the background model for clutter images was set to be 0.99. Also, for simplicity, in experiments where examples were incrementally added to the training set, the mixture model was run for a fixed number of EM iterations, specifically 10, before adding the next weakly labeled example.

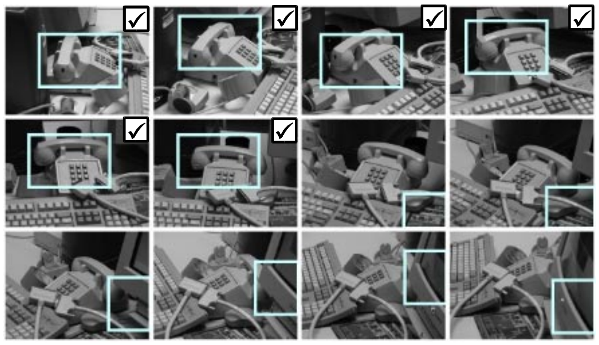
4.11 Experimental Results

4.11.1 Overview

It is useful to start an examination of the experimental results by qualitatively examining typical detection results on the test set which can be seen in Figure 25. For visualization purposes we assume that the object is present and plot a fixed size light colored rectangle centered at the location of the maximum odds ratio detection. Images which contain a correct detection are indicated with a “check” mark. The images in Figure 25a show the result on the test set for an experiment where the model was trained using only a single labeled example and the “Labeled Only” method. The detections in three out of twelve images are considered to be correct. The results in Figure 25b are for an experiment where the model was trained with a single labeled example and the “All At Once” method where weakly labeled data is added to the training set all at once. There are three correct detections in this case. The results in Figure 25c are for an experiment where the model was trained with a single labeled example and the “Incremental Odds Ratio” method where weakly labeled data is added to the training set incrementally according to the detection odds ratio. There are six correct detections in this case. The results in Figure 25d are for an experiment where the model was trained with a single labeled example and the “Incremental Reverse 1-NN” method, where weakly labeled data is incrementally added to the training set according to the method described in subsection 4.7.2. The weakly labeled training data now allows the model to correctly detect the object in eleven out of twelve images.


(a)


(b)


(c)

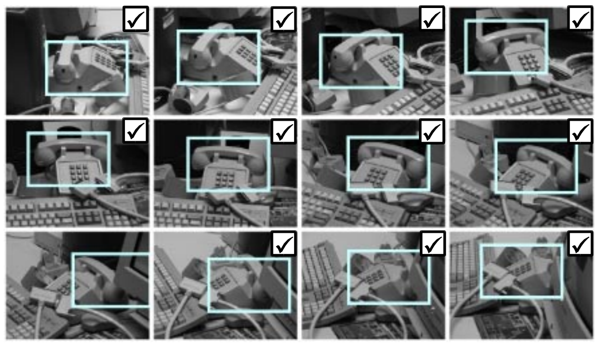

(d)

Figure 25: Detection results with one fully labeled example and different weakly labeled data algorithms. Detections are marked with a light colored rectangle and correct detections are indicated with a “check” mark. Set (a) contains the results for a detector trained on the fully labeled examples alone. Sets (b), (c), and (d) contain the results for detectors trained on the single fully labeled example and eleven weakly labeled examples. Set (b) used the “All At Once” method, set (c) used the “Incremental Odds Ratio” method and set (d) used the “Incremental Reverse 1-NN” method.

One of the goals of this set of experiments was to examine a wide range of parameters in our semi-supervised approach and empirically evaluate what effect those parameters have on performance. To that end the following sections detail the results of a large number of experiments to explore that space.

4.11.2 Detailed Results

In the sections that follow, we describe each experimental variation in detail. In this section we introduce Tables 6 and 7 which contain a full summary of the experimental results. In subsequent sections we refer back to these tables to present specific performance measurements. Each experimental condition is a separate row in the table and each pair of columns is a different data set. In Table 6 the performance is listed as the average number of correct detections on the test set for each of the 12 sub-experiments. Table 7 presents the same results as the average percentage correct, which can make it easier to understand and compare the performance measurements. Also presented are 95% significance intervals for the performance results, which are computed as 1.64 times the standard error of the mean.

It is important to note that when the percentage of correct detections is close to 50%, it does not imply chance detection results. Chance would result in much worse performance since many detection locations are being evaluated and only one is being selected. The chance detection level is approximately the pixel area of the detection circle divided by the pixel area of the test image, $40^2 \times \pi/(320 \times 240) = 0.065$, which is 6.5%. Notice that the detector achieves 100% accuracy when the full data set is fully labeled. So the detection problem is tractable with the given training data for this test set. (Note that in this case, only a single experiment is performed because there is only one possible partition of the data set.)

4.11.3 Establishing Upper and Lower Performance Bounds

The purpose of this set of experiments is to establish lower and upper bounds on the semi-supervised training performance.

To establish a lower performance bound, a set of models was trained with different training sets consisting of subsets of the fully labeled data only. The results in Figure 26a are for for 12 different data partitions and a single image in each partition. Figure 26b shows results for models trained with “close pair” images. The results in Figure 26c are for models trained with “near pair” images. The results in Figure 26d are for models trained with “far pair” images. The detailed results are reported in Tables 6 and 7 in the row labeled “Labeled Only”.

We first observe that, in all cases, using two images results in better average performance, 40.8%-52.5%, than using one, 26.7%. It also appears that the models trained with the “near pair” images perform better on average, 52.5%, than the images trained with the “close” set, 40.8%. The possible reason for this is that the information contained in the “close pair” images is redundant; much of the information contained in the second image was already present in the first image. In the case of the “far” results compared to the near results, the outcome is more mixed at 50.8%. The “far pair” models always perform better than the single image models, but when compared to the “near’ case, in some cases perform worse and in some cases perform better. One possible explanation for the cases with inferior performance is that when the appearance of two training images differs greatly, the algorithm may “interpolate” between two parts of the space where it is very uncertain. This generates erroneous results.

	Single		Close Pair		Near Pair		Far Pair	
Algorithm	# Corr	Conf L/H	# Corr	Conf L/H	# Corr	Conf L/H	# Corr	Conf L/H
Full Data Set	12	12 / 12	12	12 / 12	12	12 / 12	12	12 / 12
True Location	10.4	9.6 / 11.2	11.5	11.3 / 11.7	11.8	11.6 / 12.0	11.8	11.5 / 12.1
Labeled Only	3.2	2.6 / 3.8	4.9	4.3 / 5.5	6.3	5.7 / 5.9	6.1	5.2 / 7.0
All at Once	2.3	1.6 / 3.0	4.3	3.4 / 5.2	6.3	5.7 / 6.9	6.5	5.5 / 7.5
Incremental Odds Ratio	4.1	2.6 / 5.6	5.8	4.9 / 6.7	8.8	7.4 / 10.2	6.3	4.7 / 7.9
Incremental Reverse 1-NN	5.7	4.0 / 7.4	7.7	6.1 / 9.3	9.9	9.2 / 10.6	8.5	6.9 / 10.1
Incremental Reverse 2-NN	5.7	4.0 / 7.4	7.4	6.8 / 9.0	9.9	9.3 / 10.5	8.4	6.9 / 9.9
Incremental Reverse 3-NN	5.9	4.2 / 7.6	7.8	6.1 / 9.5	9.6	9.0 / 10.2	8.8	7.5 / 10.1
Incremental Reverse 4-NN	5.9	4.2 / 7.6	7.3	5.7 / 8.9	10.1	9.6 / 10.6	9.0	7.9 / 10.1
Inc. Rev. 1-NN 1-Gauss Mdl	3.8	2.5 / 5.1	4.9	3.4 / 6.4	6.8	4.9 / 8.7	6.9	5.2 / 8.6
Inc. Rev. 1-NN 2-Gauss Mdl	5.1	3.5 / 6.7	7.1	5.7 / 8.5	8.5	7.0 / 10.0	9.4	8.0 / 10.8
Inc. Rev. 1-NN 3-Gauss Mdl	5.2	3.4 / 7.0	6.6	5.3 / 7.9	9.6	8.6 / 10.6	9.3	7.9 / 10.7
Inc. Rev. 1-NN 40-Gauss Mdl	6.4	4.8 / 8.0	7.8	6.1 / 9.5	10.8	10.3 / 11.3	8.8	7.3 / 10.3
Inc. Rev. 4-NN 40-Gauss Mdl	6.4	4.7 / 8.1	8.3	6.7 / 9.9	10.3	9.6 / 11.0	9.2	8.3 / 10.1
All at Once Linear Sched.	2.4	1.6 / 3.2	4.9	4.0 / 5.8	6.4	5.9 / 6.9	6.9	5.8 / 8.0
All at Once Sq. Root Sched.	2.3	1.5 / 3.1	5.2	4.3 / 6.1	6.5	5.9 / 7.1	6.6	5.6 / 7.6
All at Once Squared Sched.	2.2	1.3 / 3.1	4.3	3.5 / 5.1	6.6	6.1 / 7.1	6.7	5.6 / 7.8

Table 6: Results of training with weakly labeled data. The column labeled “# Corr” is the average number correct over all 12 experiments for that algorithm. The columns labeled “Single Example” are for models trained with one example, “Close Pair” is for two images similar in pose, “Near Pair” is for two images with relatively different poses, “Far Pair” is for two images with very different poses.

	Single		Close Pair		Near Pair		Far Pair	
Algorithm	% Corr	Conf L/H	% Corr	Conf L/H	% Corr	Conf L/H	% Corr	Conf L/H
Full Data Set	100.0	100 / 100	100	100 / 100	100	100 / 100	100	100 / 100
True Location	86.7	80.0 / 93.4	95.8	93.8 / 97.8	98.3	96.8 / 99.8	98.3	96.0 / 100
Labeled Only	26.7	21.6 / 31.8	40.8	36.1 / 45.5	52.5	47.6 / 57.4	50.8	43.0 / 58.6
All at Once	19.2	13.1 / 25.3	35.8	27.9 / 43.7	52.5	47.3 / 57.7	54.2	45.5 / 62.9
Incremental Odds Ratio	34.2	21.5 / 46.9	48.3	40.9 / 55.7	73.3	62.0 / 84.6	52.5	39.0 / 66.0
Incremental Reverse 1-NN	47.5	33.3 / 61.7	64.2	51.0 / 77.4	82.5	76.6 / 88.4	70.8	57.6 / 84.1
Incremental Reverse 2-NN	47.5	33.1 / 61.9	61.7	48.2 / 75.2	82.5	77.3 / 87.7	70.0	57.8 / 82.2
Incremental Reverse 3-NN	49.2	35.4 / 63.0	65.0	51.1 / 78.9	80.0	75.0 / 85.0	73.3	62.2 / 84.4
Incremental Reverse 4-NN	49.2	35.4 / 63.0	60.8	47.7 / 73.9	84.2	79.8 / 88.6	75.0	65.9 / 84.1
Inc. Rev. 1-NN 1-Gauss Mdl	31.7	21.0 / 42.4	40.8	28.1 / 53.5	56.7	40.6 / 72.8	57.5	43.7 / 71.3
Inc. Rev. 1-NN 2-Gauss Mdl	42.5	28.8 / 56.2	59.2	47.6 / 70.8	70.8	58.5 / 83.1	78.3	66.9 / 89.7
Inc. Rev. 1-NN 3-Gauss Mdl	43.3	28.6 / 58.0	55.0	43.9 / 66.1	80.0	72.0 / 88.0	77.5	65.7 / 89.3
Inc. Rev. 1-NN 40-Gauss Mdl	53.3	40.1 / 66.5	65.0	50.9 / 79.1	90.0	85.5 / 94.5	73.3	60.8 / 85.8
Inc. Rev. 4-NN 40-Gauss Mdl	53.3	38.8 / 67.8	69.2	56.2 / 82.2	85.8	80.2 / 91.4	76.7	68.8 / 84.6
All at Once Linear Sched.	20.0	13.1 / 26.9	40.8	33.7 / 47.9	53.3	49.5 / 57.1	57.5	48.5 / 66.5
All at Once Sq. Root Sched.	19.2	12.7 / 25.7	47.3	36.1 / 50.5	54.2	49.2 / 59.2	55.0	46.6 / 63.4
All at Once Squared Sched.	18.3	11.1 / 25.5	35.8	28.9 / 42.7	55.0	50.9 / 59.1	55.8	46.8 / 64.8

Table 7: Results of training with weakly labeled data, expressed as percentages. The column labeled “% Corr” is the average percentage correct. The columns labeled “Single Example” are for models trained with one example, “Close Pair” is for two images similar in pose, “Near Pair” is for two images with relatively different poses, “Far Pair” is for two images with very different poses.

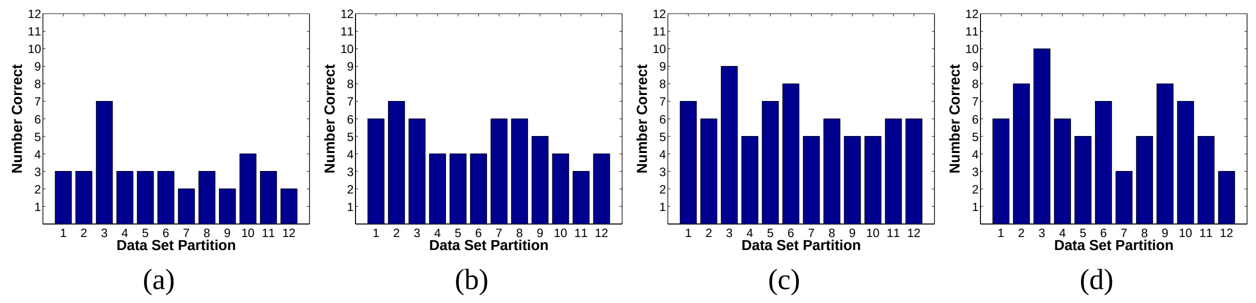


Figure 26: Plots of the performance of models *trained with fully labeled data only.* The number correct on a test set for twelve different training and test set partitions is displayed. Subfigure (a) is for single images, (b) is for “close pair” images, (c) is for “near pair” images, and (d) is for “far pair” images.

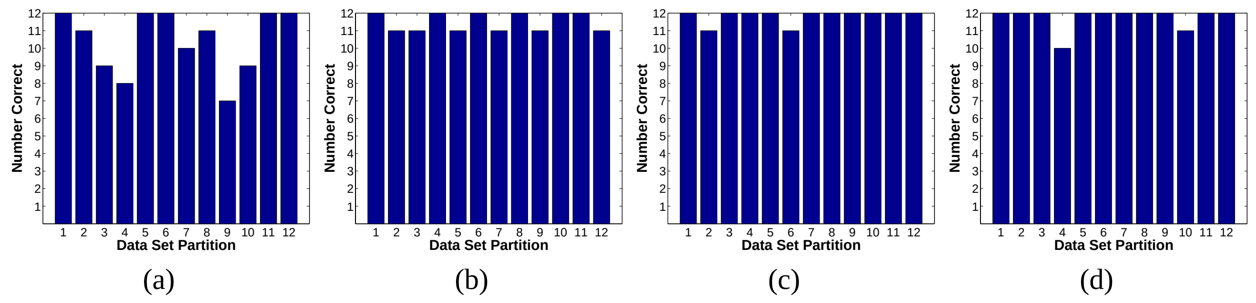
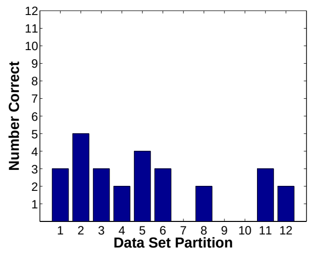


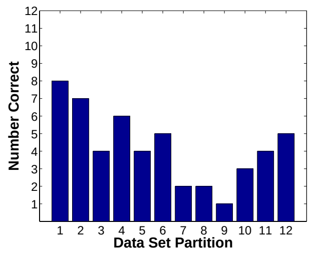
Figure 27: Plots of the performance of models *trained with fully labeled data and ground truth information for the weakly labeled data.* The number correct on a test set for twelve different training and test set partitions is displayed. Subfigure (a) is for single images, (b) is for “close pair” images, (c) is for “near pair” images and (d) is for “far pair” images.

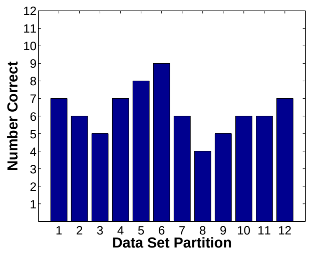
To establish an upper performance bound, the detector is trained using a mix of fully labeled data and weakly labeled data, but it is only given the ground truth object location for the weakly labeled data. No pixel-wise labeled image mask is given for the weakly labeled training data. Because the semi-supervised training algorithm only estimates the location of the object in the weakly labeled data and not the image mask, this establishes an an upper bound on what we would expect to see from semi-supervised training. The results for this experiment are plotted in Figure 27. The detailed results are reported in Tables 6 and 7 in the row labeled “True Location.” As one can see, the performance achieved from this scenario is quite good, reaching 86.7%, 95.8%, 98.3%, and 98.3% correct for the various experimental conditions. This means that our decision to only estimate the location of the object in our semi-supervised training approach, and not generate a pixel-wise labeling, will not greatly limit achievable performance.

4.11.4 Evaluating standard EM - all weakly labeled data at once

In this set of experiments, the standard EM approach was used, in which all weakly labeled images were added at once, and the latent variables, the object positions, were estimated at each iteration. Models were trained with different training sets consisting of subsets of fully labeled data, and the remaining data in the training set was weakly labeled. The results in Figure 28a are for 12 different data partitions and a single


(a)


(b)


(c)

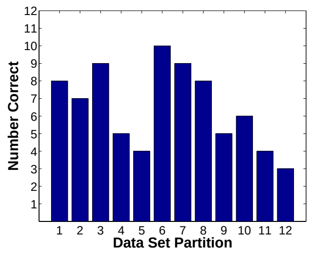

(d)

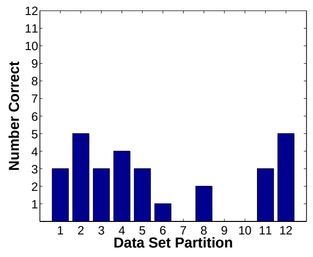
Figure 28: Plots of the performance of models *trained with all weakly labeled data at once.* The number correct on a test set for twelve different training and test set partitions is displayed. Subfigure (a) is for single images, (b) is for "close pair" images, (c) is for "near pair" images, and (d) is for "far pair" images.

labeled image, with 11 weakly labeled images in each partition. Figure 28b shows the results for models trained with fully labeled "close pair" images and 10 weakly labeled examples. The results in Figure 28c are for models trained with fully labeled "near pair" images and 10 weakly labeled examples. Figure 28d shows the results for models trained with fully labeled "far pair" images and 10 weakly labeled examples. The detailed results are reported in Tables 6 and 7 in the row labeled "All at Once".

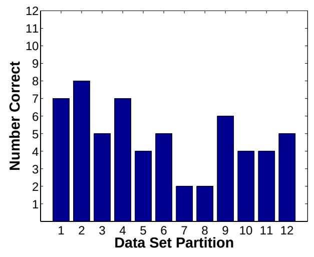
In the single image case, the results with weakly labeled data are always the same as, or worse than, the results without weakly labeled data. The one exception is the model trained with a single fully labeled image where the performance was 19.2% with weakly labeled data vs. 26.7% for fully labeled data alone. This may be explained by weakly labeled data swamping out fully labeled data. The result is that, once the algorithm makes a mistake when labeling the weakly labeled data, it is pulled in an incorrect direction and there is in effect "positive feedback" where further incorrect detections pull the model further and further in the wrong direction. In the case of image pairs, the results are more equivocal. Sometimes the weakly labeled data seems to help a little and sometimes it seems to hurt. The average performance when comparing models trained with weakly labeled data vs. models trained with fully labeled data alone is as follows, close: 35.8% vs. 40.8%, near: 52.5% vs. 52.5%, far: 54.2% vs. 50.8%.

4.11.5 Evaluating weakly labeled data weight schedule weighting with standard EM

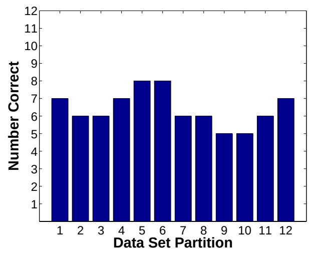
As noted in the previous section, one possible explanation for the poor performance of the standard EM approach is that, because the amount of weakly labeled data is large compared to the fully labeled data, it has a large influence on the training process and is swamping out the fully labeled data. This observation was made and briefly explored in earlier work with unlabeled data in [Nigam 98]. We adopt an approach similar to the one suggested by the authors and re-weight the contribution of the weakly labeled data in the EM formulation so it does not overwhelm the fully labeled data. In our implementation we train for a fixed number of iterations and use a scheduled approach, where the weight of all the weakly labeled data combined is equivalent to the weight of a single fully labeled example at the beginning of training. Over the course of training, the weight of the weakly labeled examples slowly changes until finally reaching the point where each weakly labeled example has the same weight as a fully labeled example. The rationalization is that, as the model becomes more certain about the labeling of the weakly labeled data, we can increase its contribution to the estimation of the model parameters, akin to an annealing approach. We performed experiments using different schedules and present their results:



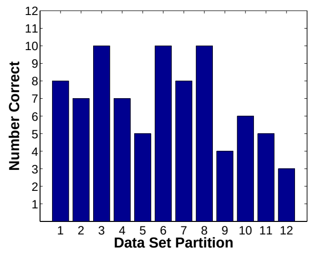
(a)



(b)

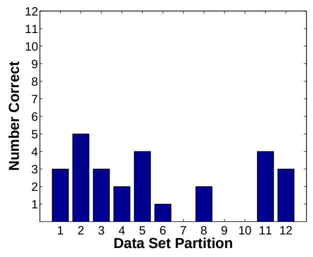


(c)

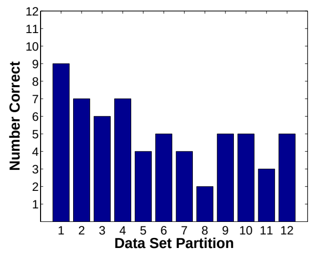


(d)

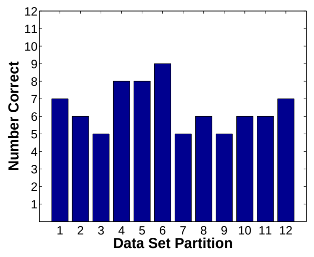
Figure 29: Plots of the performance of models *trained with all data at once and linear schedule weighting.* The number correct on a test set for twelve different training and test set partitions is displayed. Subfigure (a) is for single images, (b) is for “close pair” images, (c) is for “near pair” images, and (d) is for “far pair” images.



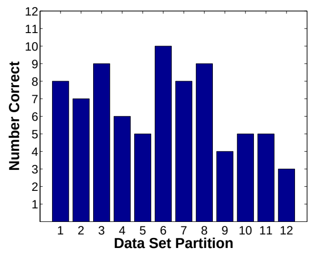
(a)



(b)



(c)



(d)

Figure 30: Plots of the performance of models *trained with all data at once and square root schedule weighting.* The number correct on a test set for twelve different training and test set partitions is displayed. Subfigure (a) is for single images, (b) is for “close pair” images, (c) is for “near pair” images, and (d) is for “far pair” images.

Linear schedule. The results can be seen in Figure 29. As one can see the results are equal to or worse than the all at once case. The detailed results are reported in Tables 6 and 7 in the row labeled “All at Once Linear Sched.” These results are promising; in all cases a small improvement is seen when comparing the average performance for scheduled training vs. non-scheduled training. Results as follows, single: 20.0% vs. 19.2%, close: 40.8% vs. 35.8% , near: 53.3% vs. 52.5%, far 57.5% vs. 54.2%. However, the results are not quite as promising when we compare scheduled training vs. fully labeled training as follows, single: 20.0% vs. 26.7%, close: 40.8% vs. 40.8% , near: 52.5% vs. 52.5%, far 57.5% vs. 50.8%.

Square root schedule. In this case, the weights grow more quickly than the linear schedule. The results are shown in Figure 30. The detailed results are reported in Tables 6 and 7 in the row labeled “All at Once Sq. Root Sched.” Again these results are promising, in all cases being the same as or showing a small improvement in average performance for scheduled training vs. non-scheduled training. The results are as follows, single: 19.2% vs. 19.2%, close: 47.3% vs. 35.8% , near: 54.2% vs. 52.5%, far 55.0% vs. 54.2%. In this case, they are also somewhat promising when we compare them to the fully labeled only case, especially for the two image training sets. Results are as follows, single: 19.2% vs. 26.7%, close: 47.3% vs. 40.8% , near: 54.2% vs. 52.5%, far 55.0% vs. 50.8%.

Squared schedule. In this case, the weights grow less quickly than the linear schedule. The results are plotted in Figure 31. The detailed results are reported in Tables 6 and 7 in the row labeled “All at Once

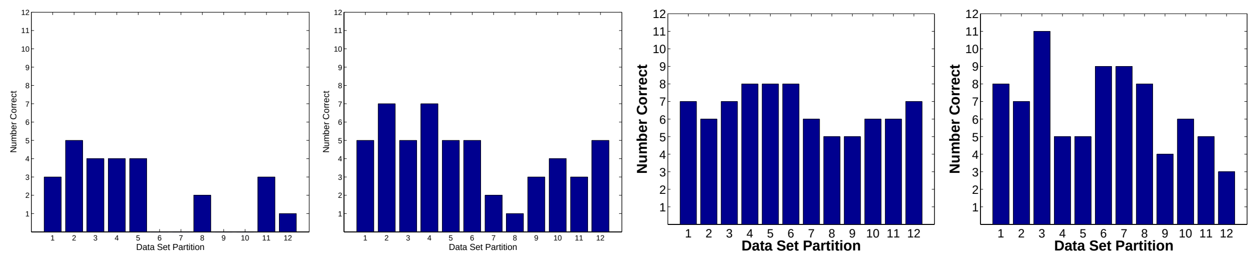


Figure 31: Plots of the performance of models *trained with all data at once and squared schedule weighting.* The number correct on a test set for twelve different training and test set partitions is displayed. Subfigure (a) is for single images, (b) is for "close pair" images, (c) is for "near pair" images, and (d) is for "far pair" images.

Squared Sched." The results are not quite as promising here, since in all cases the average performance for scheduled training is much closer to the non-scheduled case. The specific average performance values for scheduled training vs. non-scheduled training follows: single: 18.3% vs. 19.2%, close: 35.8% vs. 35.8% , near: 55.0% vs. 52.5%, far 55.8% vs. 54.2%. In this case, they are not particularly promising when we compare them to the fully labeled only case. The performance values for scheduled training vs. fully labeled training follows: single: 19.2% vs. 26.7%, close: 35.8% vs. 40.8% , near: 55.0% vs. 52.5%, far 54.2% vs. 50.8%.

In summary, a small gain was observed when a weighting schedule was used for the incorporation of weakly labeled data. In our experiments, this was only observed for the two image training sets and the largest advantage was seen with the square root schedule which increased the weight of the weakly labeled data more quickly than linear. However, it should be noted that the improvement in performance achieved with this semi-supervised approach over the performance achieved with the fully labeled data alone was very modest at best.

4.11.6 Evaluating incremental data addition based on the detect odds ratio

A self-training approach was used in this set of experiments, in which, as a variation on the standard EM approach, the weakly labeled images were added incrementally, one at a time. Also, the latent variables, for example the object position for those weakly labeled images added to the training set, were not re-estimated at each iteration. Images were selected to be added to the training set based on the odds ratio of the maximum likelihood detection for that image.

In this set of experiments, models were trained with different training sets consisting of subsets of fully labeled data and the remaining data in the training set was weakly labeled. The results in Figure 32a are for 12 different data partitions, with a single labeled image and 11 weakly labeled images in each partition. The results in Figure 32b are for models trained with fully labeled "close pair" images and 10 weakly labeled examples. The results in Figure 32c are for models trained with fully labeled "near pair" images and 10 weakly labeled examples. The results in Figure 32d are for models trained with fully labeled "far pair" images and 10 weakly labeled examples. The detailed results are reported in Tables 6 and 7 in the row labeled "Incremental Odds Ratio".

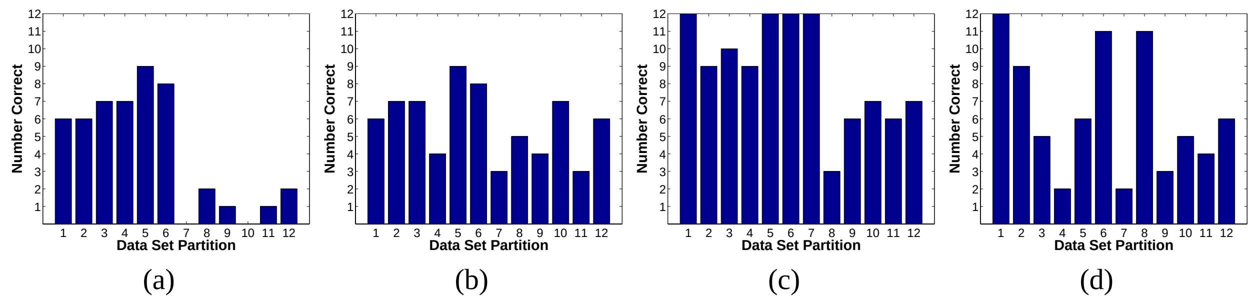


Figure 32: Plots of the performance of models *trained with incremental add based on detect odds ratio.* The number correct on a test set for twelve different training and test set partitions is displayed. Subfigure (a) is for single images, (b) is for “close pair” images, (c) is for “near pair” images, and (d) is for “far pair” images.

In the single image case, the results with weakly labeled data are sometimes much better and sometimes much worse. So it would seem that sometimes the model is still diverging from the correct answer. However, on average, an improvement is achieved over fully labeled data alone, 34.2% vs. 26.7%.

The results are more interesting in the case of image pairs. For the “close” and “far” cases, the results are similar to the single image case; sometimes the model is much better, and sometimes it is much worse. However, in the “near” case, the results always improve with the use of weakly labeled data and are often much better. It would seem that in this case the model is able to accurately generalize to nearby regions in space, and can use that generalization ability to incrementally extend the model and not diverge. The key aspect, which would seem to be important, is the ability to accurately predict the confidence of a detection in the weakly labeled data. Again, when we examine the average performance of the incremental method vs. the performance of fully labeled data alone, the incremental method is always better: close: 48.3% vs. 40.8%, near: 73.3% vs. 52.5%, far: 52.5% vs. 50.8%.

So in summary it appears that the incremental approach empirically exhibits a clear advantage over the standard EM “all at once” approach, by consistently extracting increased performance from the weakly labeled data.

4.11.7 Evaluating incremental data addition based on reverse odds ratio (1-NN)

In the experiments described in the previous section, self-training, an incremental variant of EM, was found to exhibit good performance. A confidence based selection metric, the odds ratio, was used to determine which image would be added to the pool of training images next. However, as was noted in section 2.5 of this document, a confidence based metric will often choose to select data points that are far from the decision boundary, and this skewed distribution of training examples can cause issues when training the detector.

In this section, we use a metric for deciding which images to add, which more closely models the behavior of a nearest neighbor approach. Images are selected based on the “reverse odds ratio” of each image as described in detail in section 4.7.2. The “reverse odds ratio” is computed by building a separate model of the maximum odds ratio detection for each weakly labeled candidate image. The “score” for a candidate is the maximum of the odds ratio over each example in the labeled training set. The candidate with the highest score is selected to be added to the training set. This scoring metric is designed to be an approximation to

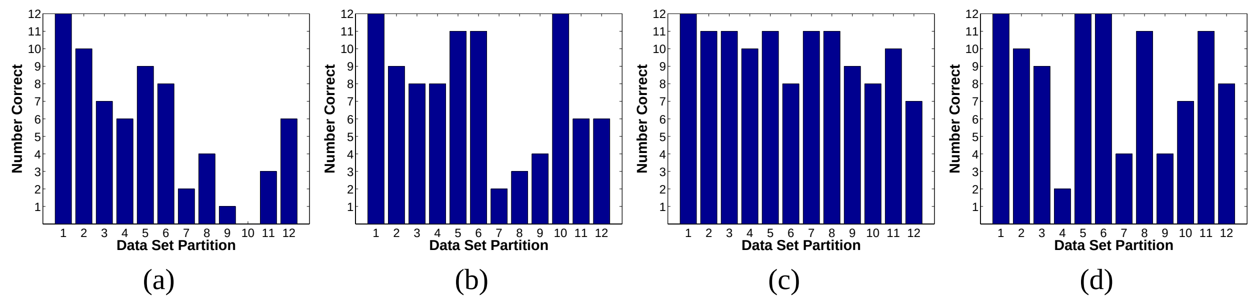


Figure 33: Plots of the performance of models *trained with incremental add based on reverse odds ratio (1-NN)*. The number correct on a test set for twelve different training and test set partitions is displayed. Subfigure (a) is for single images, (b) is for "close pair" images, (c) is for "near pair" images, and (d) is for "far pair" images.

selecting images that are the nearest neighbor of one of the training examples, and therefore highly likely to be an accurate detection.

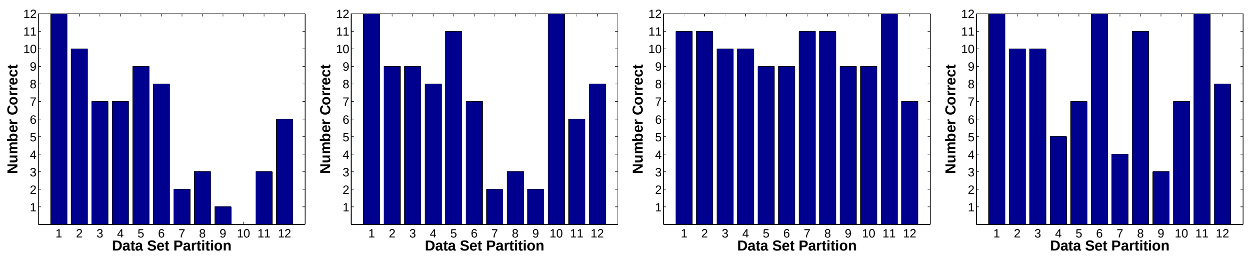
As in the previous section, in this set of experiments, a self-training approach was used where weakly labeled images were added incrementally, one at a time. Also, the latent variable, the object position, for those weakly labeled images added to the training set were not re-estimated at each iteration. Images were selected to be added to the training set based on the odds ratio of the maximum likelihood detection for that image.

In this set of experiments, models were trained with different training sets consisting of subsets of fully labeled data, and the remaining data in the training set was weakly labeled. The results in Figure 33a are for 12 different data partitions. In each partition there was a single labeled image and 11 weakly labeled images. Figure 33b displays the results for models trained with fully labeled "close pair" images and 10 weakly labeled examples. The results in Figure 33c are for models trained with fully labeled "near pair" images and 10 weakly labeled examples. Figure 33d displays the results for models trained with fully labeled "far pair" images and 10 weakly labeled examples. The detailed results are reported in Tables 6 and 7 in the row labeled "Incremental Reverse 1-NN".

The first thing to note is that, in the single image case, the results with weakly labeled data and the "reverse odds ratio" score are always better than the weakly labeled results for the odds ratio experiments even for the individual test / train splits.

In the case of image pairs, when comparing individual test / train splits, the "reverse odds ratio" results are almost always better than the "odds ratio" experiments. In the case of the "near" training set, the results are on average much better; they are also more consistent across training set splits and are starting to approach the fully labeled case. The result here again suggests that when a limited amount of labeled training is used, and selecting which data to label is possible, the specific examples chosen can have a very large effect on final performance.

When we look at the performance on average for the "reverse odds ratio" vs. the standard "odds ratio" metric, we see it always performs better: single: 47.5% vs. 34.2%, close: 64.2% vs. 48.3%, 82.5% vs. 72.3% 70.8% vs. 52.5%. In summary, for this data set, this approach extracts a large performance boost under almost all training set variations.



(a) (b) (c) (d)

Figure 34: Plots of the performance of models *trained with incremental add based on reverse odds ratio (2-NN)*. The number correct on a test set for twelve different training and test set partitions is displayed. Subfigure (a) is for single images, (b) is for “close pair” images, (c) is for “near pair” images, and (d) is for “far pair” images.

4.11.8 Evaluating incremental data addition based on reverse odds ratio (*m*-NN)

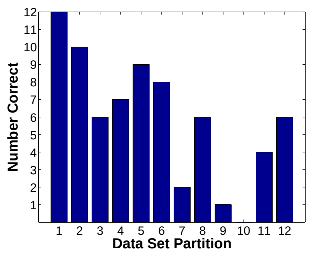
Given the success of the “reverse odds ratio” score-based selection metric introduced in the previous section, it is reasonable to explore whether there are ways of improving it. The metric is equivalent to a one nearest neighbor (1-NN) approach where only the nearest neighbor affects the selection metric for each weakly labeled example. It is reasonable to hypothesize that an *m*-NN approach might be more robust if the distance, in our case the reverse odds ratio, to the *m* nearest neighbors were combined in some way. We chose to use the mean of the *m* highest odds ratio (closest) training images. If the training set has less than *m* images, then the reverse odds ratio of just the highest scoring image was used.

In the next set of experiments, we used a two nearest neighbor (2-NN) approach. Just as in the previous section, an EM based approach with incremental addition is used with a reverse odds ratio. The individual experiment results are presented in Figure 34 and the average results are presented in Tables 6 and 7 in the row labeled “Incremental Reverse 2-NN”. When we examine the average performance for this 2-NN version vs. the 1-NN version, we see that it is almost identical or very slightly worse: single: 47.5% vs. 47.5%, close: 61.7% vs. 64.2%, near: 82.5% vs. 82.5%, far: 70.0% vs. 70.8%.

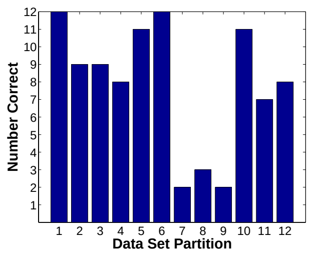
We then used a three nearest neighbor (3-NN) approach. The individual experiment results are presented in Figure 35 and the average results are presented in Tables 6 and 7 in the row labeled “Incremental Reverse 3-NN”. When we examine the average performance for this 3-NN version vs. the 1-NN version, we see that it is typically very slightly better, single: 49.2% vs. 47.5%, close: 65.0% vs. 64.2%, near: 80.0% vs. 82.5%, far: 73.3% vs. 70.8%.

Next, we used a four nearest neighbor (4-NN) approach. The individual experiment results are presented in Figure 36 and the average results are presented in Tables 6 and 7 in the row labeled “Incremental Reverse 4-NN.” When we examine the average performance for this 4-NN version vs. the 1-NN version, it is typically very slightly better, single: 49.2% vs. 47.5%, close: 60.8% vs. 64.2%, near: 84.2% vs. 82.5%, far: 75.0% vs. 70.8%.

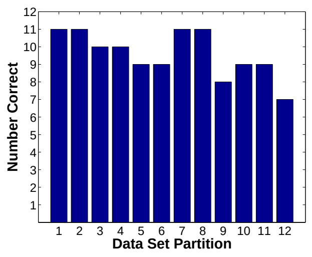
In summary, it appears that increasing the number of nearest neighbors typically had no effect or a very small effect on the final performance of the detector trained using weakly labeled data. The one case where a small effect can be seen is for the “far pair” case. Under that condition, the additional neighbors may help reject spurious associations that can occur because the fully labeled images are further from one another in terms of appearance.



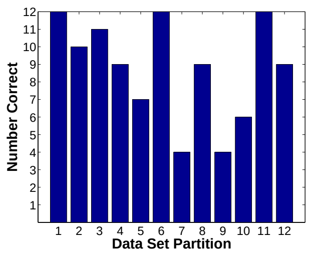
(a)



(b)

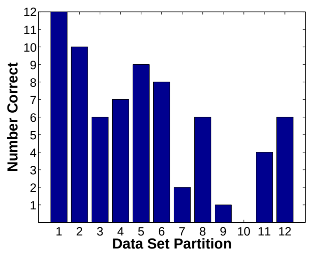


(c)

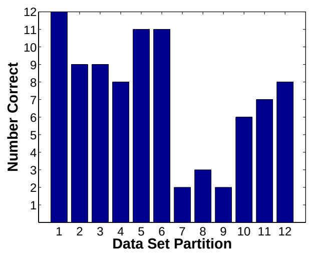


(d)

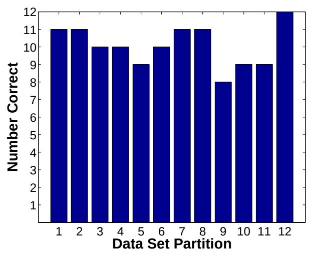
Figure 35: Plots of the performance of models *trained with incremental add based on reverse odds ratio (3-NN)*. The number correct on a test set for twelve different training and test set partitions is displayed. Subfigure (a) is for single images, (b) is for “close pair” images, (c) is for “near pair” images, and (d) is for “far pair” images.



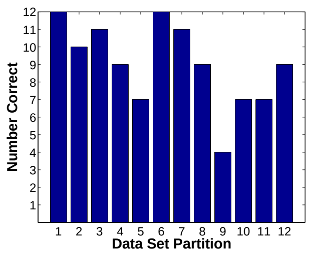
(a)



(b)

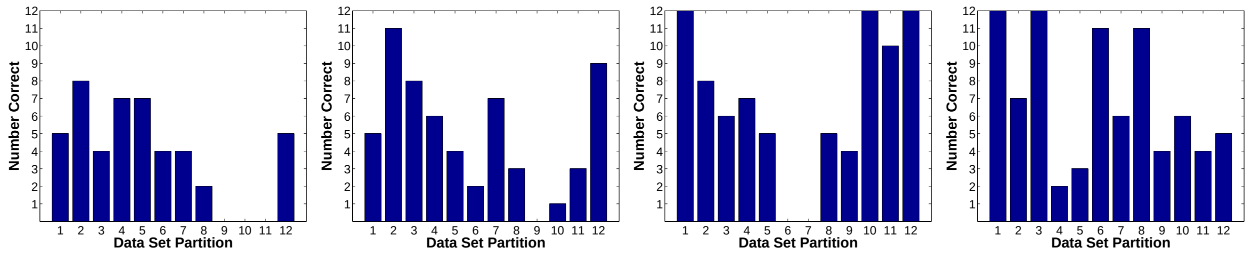


(c)



(d)

Figure 36: Plots of the performance of models *trained with incremental add based on reverse odds ratio (4-NN)*. The number correct on a test set for twelve different training and test set partitions is displayed. Subfigure (a) is for single images, (b) is for “close pair” images, (c) is for “near pair” images, and (d) is for “far pair” images.



(a) (b) (c) (d)

Figure 37: Plots of the performance of models *trained with incremental add based on reverse odds ratio and single Gaussian Model (1-NN)*. The number correct on a test set for twelve different training and test set partitions is displayed. Subfigure (a) is for single images, (b) is for “close pair” images, (c) is for “near pair” images, and (d) is for “far pair” images.

4.11.9 Evaluating incremental data addition based on reverse odds ratio and a varying number of Gaussian components for the metric (1-NN)

The purpose of this next set of experiments was to explore how the complexity of the model of each candidate image used to generate the reverse odds ratio of the training images, for the selection metric, affected final performance. The experiments here are identical to the “1-NN” neighbor experiment, but the number of components of the full covariance Gaussian mixture model that was used to model the candidate image is variable. In the previously described experiments, 20 components were used. An advantage could be gained if these models could be simplified with little performance change. The complexity of these models has a large impact on training time, since they need to be re-trained multiple times each time the detector model is updated and is re-run over the candidate images. We explore four different variations:

Single full covariance Gaussian model for the reverse odds ratio computation. The full results are presented in Figure 37 and as the average performance in Tables 6 and 7 in the row labeled “Inc. Rev. 1-NN 1-Gauss Mdl.” As one can see, the results with this simple model vs. the 20 component model are quite a bit worse: single: 31.7% vs. 47.5%, close: 40.8% vs. 64.2%, near: 56.7% vs. 82.5%, far: 57.5% vs. 70.8%. It would seem that it is important to model the feature distribution of the weakly labeled example at a certain level of fidelity for the purposes of the selection-metric, and one might expect that it is unlikely that a single Gaussian could do this.

A mixture model of two full covariance Gaussians for the reverse odds ratio computation. The full results are presented in Figure 38 and as the average performance in Tables 6 and 7 in the row labeled “Inc. Rev. 1-NN 2-Gauss Mdl.” As one can see, the results with this model vs. the 20 component model are much closer: single: 42.5% vs. 47.5%, close: 59.2% vs. 64.2%, near: 70.8% vs. 82.5%, far: 78.3% vs. 70.8%. These results suggest that the model could be significantly simplified if training time was an important issue in exchange for only a small loss in performance. It is interesting to note that performance is actually better with the simpler model in the “far” case. It may be that some smoothing helps the distance computation when the examples are further apart.

A mixture model of three full covariance Gaussians for the reverse odds ratio computation. The full results are presented in Figure 39 and as the average performance in Tables 6 and 7 in the row labeled “Inc. Rev. 1-NN 3-Gauss Mdl.” As one can see, the results with this model vs. the 20 component model are very close: single: 43.3% vs. 47.5%, close: 55.0% vs. 64.2%, near: 80.8% vs. 82.5%, far: 77.5%

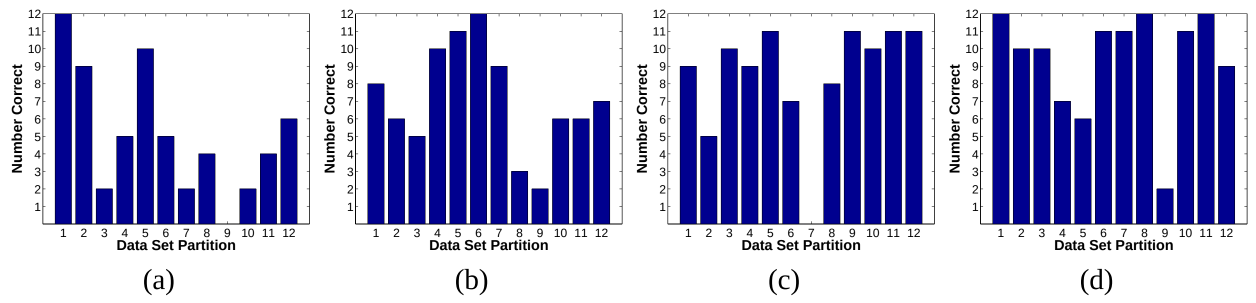


Figure 38: Plots of the performance of models *trained with incremental add based on reverse odds ratio and two Gaussian Model (1-NN)*. The number correct on a test set for twelve different training and test set partitions is displayed. Subfigure (a) is for single images, (b) is for “close pair” images, (c) is for “near pair” images, and (d) is for “far pair” images.

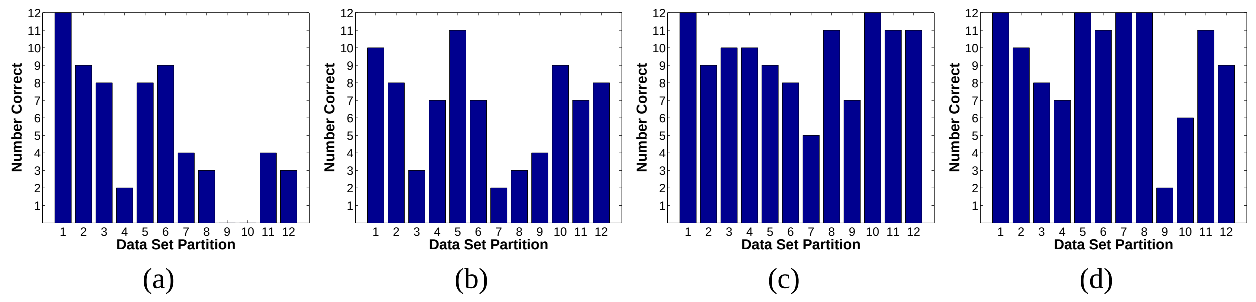


Figure 39: Plots of the performance of models *trained with incremental add based on reverse odds ratio and a three Gaussian model (1-NN)*. The number correct on a test set for twelve different training and test set partitions is displayed. Subfigure (a) is for single images, (b) is for “close pair” images, (c) is for “near pair” images, and (d) is for “far pair” images.

vs. 70.8%. These results are even better than the two component model, again suggesting that the model could be significantly simplified if training time was an important issue for only a small loss in performance. Again, performance is actually better with the three component model than the 20 model in the “far” case.

A mixture model of 40 full covariance Gaussians for the reverse odds ratio computation. The goal of this experiment was to look at the opposite end of the spectrum and see if any performance could be gained by increasing the complexity of the model used for the reverse odds ratio computation at the cost of increased training time. The full results are presented in Figure 40 and as the average performance in Tables 6 and 7 in the row labeled “Inc. Rev. 1-NN 40-Gauss Mdl.” As one can see, the results with this model vs. the 20 component model are typically better, as follows: single: 53.3% vs. 47.5%, close: 65.0% vs. 64.2%, near: 90.0% vs. 82.5%, far: 73.3% vs. 70.8%. The results here are better than the 20 component model, except in the “far pair” case. This suggests that if final detector performance is paramount, a more complex model is worth the additional training complexity.

In summary, the experiments in this section suggest that if reducing training complexity is important, then the model used to compute the reverse odds ratio metric can be greatly simplified and only a small performance penalty will be paid. On the other hand, maximum performance can be achieved by increasing the model complexity significantly at the cost of greatly increased training times. The one notable exception is in the case of “far pair” images, where a possible smoothing effect from a simpler model achieves greater

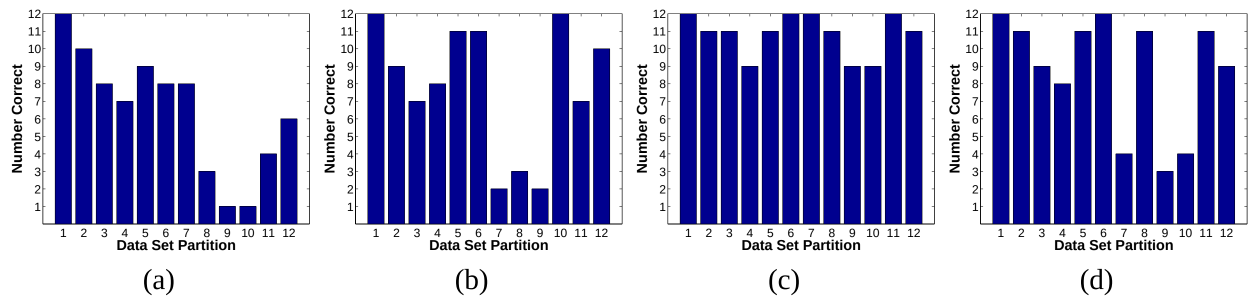


Figure 40: Plots of the performance of models *trained with incremental add based on reverse odds ratio and forty Gaussian Model (1-NN).* The number correct on a test set for twelve different training and test set partitions is displayed. Subfigure (a) is for single images, (b) is for "close pair" images, (c) is for "near pair" images, and (d) is for "far pair" images.

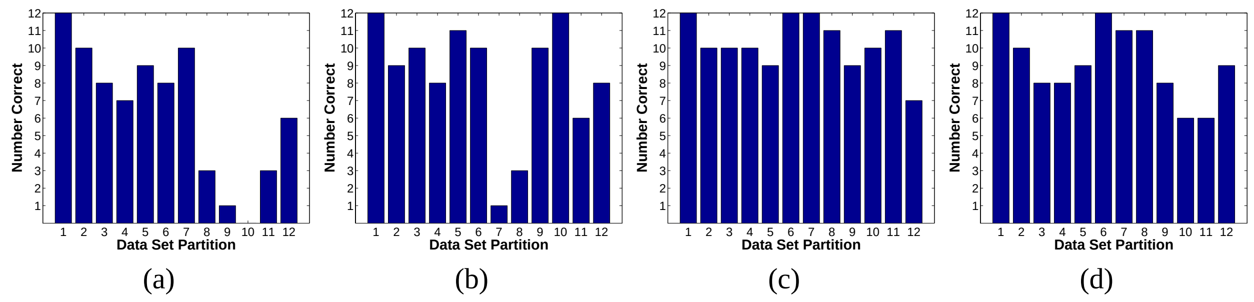


Figure 41: Plots of the performance of models *trained with incremental add based on reverse odds ratio and forty Gaussian Model (4-NN).* The number correct on a test set for twelve different training and test set partitions is displayed. Subfigure (a) is for single images, (b) is for "close pair" images, (c) is for "near pair" images, and (d) is for "far pair" images.

performance.

4.11.10 Evaluating incremental data addition based on reverse odds ratio and forty Gaussian Model (4-NN)

In this set of experiments, we took the maximum performance concept to its logical conclusion. In this case, the experiment is identical to the "4-NN" neighbor experiment but a 40 component mixture model of full covariance Gaussians was used to model the candidate image for the selection metric computation, instead of a 20 component mixture model. The results are presented in Figure 41 and in Tables 6 and 7 in the row labeled "Inc. Rev. 4-NN 40-Gauss Mdl." The average performance for the 4-NN 40 component model vs. the 1-NN 20 component model is better under all experimental conditions: single: 53.3% vs. 47.5%, close: 69.2% vs. 64.2%, near: 85.8% vs. 82.5%, far: 76.7% vs. 70.8%. Other models occasionally achieve very slightly better performance. One notable exception is the 90% accuracy achieved for the 1-NN 40-Gauss model under the "near pair" conditions. Overall, we believe that if final performance is most important, then this model would be a good choice.

4.12 Conclusions

There are a number of conclusions which can be drawn from the experimental results detailed in this chapter. The most important is that the standard EM approach of incorporating weakly labeled data performed poorly in our experiments when compared to the incremental approach used in self-training. This can be seen visually for the detection results presented in Figure 25. Figure 25a shows the test set results for an experiment where the model was trained using only a single fully labeled example and the “Labeled Only” training method. Here three out of twelve detections are considered to be correct. In Figure 25b the model was trained with the same fully labeled data and eleven weakly labeled examples using the EM method. The number of correct detections remains the same at three. In Figure 25d where the model was trained with the incremental reverse model method of incorporating weakly labeled data, eleven out of twelve detections are considered correct.

There are a number of key points can be gleaned from our experimental results:

Weakly labeled data overload. One hypothesis for the poor performance of EM was that the weakly labeled data swamped the information in the fully labeled data. To that end we performed a set of experiments which reduced the influence of the weakly labeled data during the training process. This was done by instituting a scheduled re-weighting of the weakly labeled data, so the influence started small and increased during the training process. However, we found that this did not improve performance.

Self-training outperforms EM and the sample selection metric is critical. One possible explanation for the strong performance of the incremental approach is that it only incorporates the weakly labeled examples in the training process of which it was more certain. Indeed, we observed a sizable performance benefit in terms of detector performance from the weakly labeled data when the incremental self-training approach was used. We also found that this benefit was greatly enhanced with the choice of a specific score-based selection metric to decide which weakly labeled image should be accepted next into the training set. In our case, we experimented with two different metrics: the standard measure of confidence, the odds ratio of the detection, and what we call a reverse odds ratio metric, which implements a nearest neighbor approach using a specialized local distance metric. We found that the reverse odds ratio provided a large performance boost. Under one set set of experimental conditions, the average accuracy achieved from using fully labeled data alone was boosted from 52.5% to 90.0% with the addition of weakly labeled data. We believe this is because this new distance metric better emulates a nearest neighbor approach to the selection of new weakly labeled examples. The standard confidence metric will select examples which are highly confident, but which are potentially far from the current examples. This can skew the distribution of training examples such that performance is degraded. We also explored various parameters of the underlying model used to compute the local distance metric and found that some trade-offs could be made between complexity and performance. However, the overall effect was small.

Initial training set importance. Another goal of these experiments was to try to understand how the distribution of fully labeled examples affected the final performance. To investigate this, we performed experiments with different fully labeled training sets. In all of our experiments, we consistently observed that the maximum gain from weakly labeled data was achieved for the “near pair” versus the “close pair” or “far pair.” One can hypothesize that this is because the fully labeled “close pair” images are too similar in appearance, so the information they provided is redundant. And in the case of the “far pair,” the images are too dissimilar in appearance so there may not be any single region in the feature space which the initial detector is certain about, so it is difficult for it to generalize to the weakly labeled data. This suggests that, in general, if it is possible to choose the specific images that will be labeled in the initial fully labeled training set, careful selection of these images, taking their similarity into consideration, has an important affect on performance.

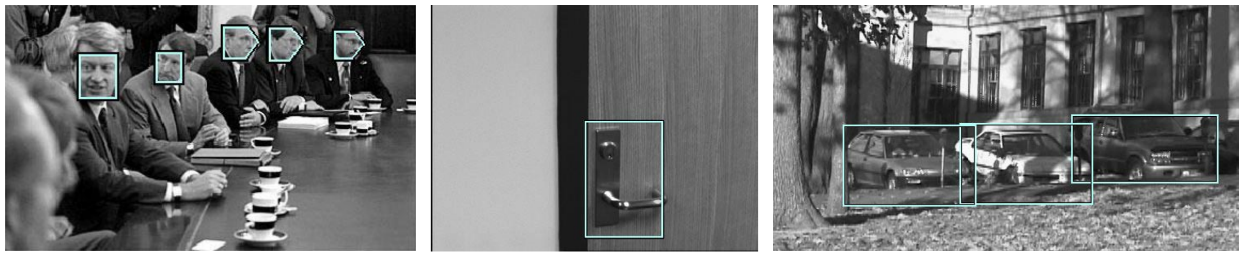


Figure 42: Example detections from the Schneiderman detector for models trained to detect frontal and profile faces, door handles, and cars respectively from left to right.

5 Schneiderman Detector Experiments

5.1 Overview

The goal of this set of experiments was to explore the behavior of the weakly labeled data approach on a more powerful, state of the art detector. We chose the Schneiderman detector, [Schneiderman 03, 04a, 04b, 04c]. It has been used successfully for face detection and other rigid objects, examples of which are presented in Figure 42, and has been demonstrated to be one of the most accurate in terms of face detectors available. The full details of the detector are provided in the next section.

The object chosen for the experiments with this detector was a frontal human eye. This object was felt to be a good target because, in a final application setting, the output of a face detector could provide many weakly labeled examples for training an eye detector. Again, in this work, weakly labeled images here are images that are known to contain the object, but for which specific hidden variables like size and location are missing.

5.2 Detector Details

An overview of the Schneiderman detector is presented here. For a more detailed description, please see [Schneiderman 03, 04a, 04b, 04c]. The Schneiderman detector is an appearance-based detector; that is, the detector builds a model of the appearance of the object to be detected using solely a set of registered training images for both the positive and negative classes. The basic detector is able to capture certain aspects of appearance variation such as intra-class variation, and a small amount of shift, scale, and rotation variation. (Lighting variation is typically handled by a pre-processing step.) Large appearance variation is handled in a different manner. To handle large changes in scale and translation, the detector is scanned over the image at different scales and locations, and each corresponding subwindow is run through the detector as is presented schematically in the first stage in Figure 43. Large changes in object pose are handled using separate models for each pose. In the work presented here, only a single pose and model are used.

The detector also consists of a cascade of detectors, whose architecture has been described frequently in recent literature. In this architecture, a number of detectors are placed in series; only image patches which are accepted by the first detector are passed on to the next. There are two potential advantages of this architecture. The first is speed. If the detectors early in the cascade are simpler and faster and tuned to generate very few false negatives, they can be used to greatly reduce the number of image patches which are passed onto later stages in the cascade. The second potential advantage is that the detector cascade

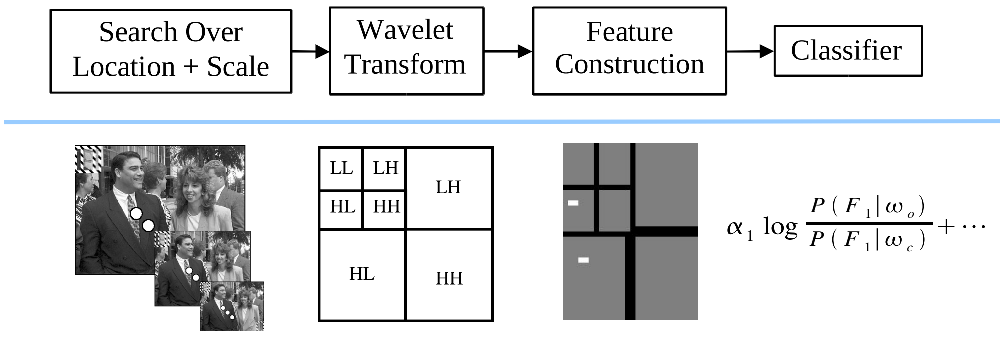

Search Over Location + Scale
Wavelet Transform
Feature Construction
Classifier
LL
LH
HL
HH
LH
HL
HH
$\alpha_1 \log \frac{P(F_1|\omega_o)}{P(F_1|\omega_c)} + \cdots$

Figure 43: A schematic representation of the detection process for a single stage of the Schneiderman detector. A search over location and scales generates subwindows which are processed via a wavelet transform, whose values are used to construct feature values which are finally passed through a classifier.

can form more complex decision boundaries than the individual detectors in the cascade. This can be thought of as similar to boosting. With each successive level of the cascade, some part of the original input space (examples) are removed from consideration. One could think of a circular decision boundary being approximated by the combination of a set of linear separators. In this work we only use a single stage of the cascade to simplify the training process. Accordingly, detection performance is lower than what is typically achieved for the detector.

5.2.1 Detection Process

A schematic representation of the detection process is presented in Figure 43. The specifics of the detection process for a single subwindow are as follows:

1. Pre-process the subwindow for lighting correction.
2. Perform a wavelet transform of the subwindow. (In practice this can be performed in a more efficient manner than transforming each subwindow independently.)
3. Extract features from current subwindow for cascade stage *n*.
4. Classify the subwindow by computing a linear function of the likelihood ratio of the features for cascade stage *n*.
5. Threshold the function output to either reject or accept subwindow.
6. If there are further stages in the cascade then go to step 2 and repeat for the next stage.

The feature extraction process is as follows. First, a two level wavelet transform of the image is performed. Then small groups of a subset of the wavelet coefficients across levels are concatenated to form a real valued vector. (The specific groups of wavelet coefficients that are combined are selected during the training process using cross validation.) The real valued vector is then vector quantized. One of the methods of simplifying the complexity of the detector is to use the same feature (arrangement of wavelet coefficients) throughout the subwindow; this is typically done in the first stage. (We only use the first stage in this work.)

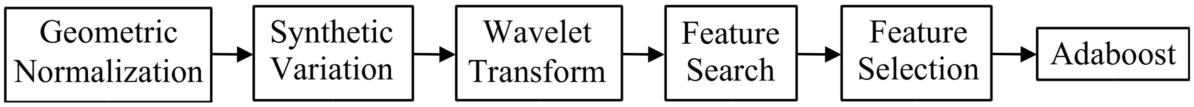


Figure 44: A schematic representation of the training process for a single stage of the Schneiderman detector. Each training image is first geometrically normalized and then synthetic variations of it are generated. The wavelet transform is then applied to each training image and then a feature search and feature selection are performed. Finally the weights of the classifier are trained with Adaboost.

However, even if the arrangement of wavelet coefficients is the same, the response from each shifted location in the subwindow is treated differently. And not all positions will necessarily be utilized as features in the discriminative process.

5.2.2 Training with Fully Labeled Data

A schematic representation of the training process is presented in Figure 44. Training the model with fully labeled data consists of the following steps:

1. Given the training data landmark locations, geometrically normalize the training example subimages for both the positive and negative examples.
2. Generate synthetic training examples. This consists of scaling, shifting, and rotating the images by small amounts. This will allow the final detector to be less sensitive to these types of changes in the input and allow scans over position and scale to be more coarse.
3. Apply lighting normalization to the subimages.
4. Compute the wavelet transform of the subimages.
5. Start with individual wavelet coefficients and greedily add them to each group. Quantize each group and build a naive Bayes model with respect to each group. Use this model to find highly correlated groups of wavelet coefficients that are more discriminative than the individuals. Use performance on a cross validation set to measure discrimination performance.
6. Build a naive Bayes model to discriminate between positive and negative examples.
7. Adjust the naive Bayes model using boosting, but maintaining a linear decision function, effectively performing gradient descent on the margin.
8. Compute an ROC curve for the detector using a cross validation set.
9. Choose a threshold for the linear function, based on the final performance desired. If this is an intermediate stage in the cascade, then a threshold is chosen which will generate a low false negative rate. If this is the final stage, then a threshold is chosen which will provide high accuracy.
10. If this is not the final stage of the cascade then a training set is generated for the next stage of the cascade. Positive examples are all those images in the current training set which passed the detection tests for the previous stages. Negative examples are generated in a bootstrapping process whereby

the current detector runs over a set of images known not to contain any faces. Any images which the current detector detects as faces are put into the new set of negative training examples.

11. If this is not the final stage of the cascade go to step 1.

5.2.3 Semi-Supervised Training

The goal of the experiments here is to train the Schneiderman detector with a combination of fully labeled and weakly labeled data and evaluate the resulting detector performance when trained with different combinations of labeled and unlabeled data. In this case, negative examples are assumed to be plentiful [Viola 01] and weakly labeled data are images which known to contain the object of interest, but for which the object location and size is not known.

A large number of experiments were performed which explored many variations of two broad approaches of "self-training": "all at once" and "incremental." The "all at once" version is trained as follows:

1. Train the detector using a limited amount of fully labeled positive examples and the full set of fully labeled negative examples.
2. Run the detector over the weakly labeled portion of the data set and find the maximum likelihood locations and scales of the object.
3. Use the output of the detector to label the unlabeled training examples and add them to the training set.
4. Iterate and go back to step 1. Stop after a fixed number of iterations.

The incremental version is trained as follows:

1. Train the detector using a limited amount of fully labeled positive examples and the full set of fully labeled negative examples.
2. Run the detector over the weakly labeled portion of the data set and find the maximum likelihood locations and scales of the object.
3. Use the output of the detector to label the unlabeled training examples and assign a selection score to each detection.
4. Select a subset of the newly labeled examples using the selection metric. Either select the top N or use a threshold based on the confidence metric.
5. Iterate and go back to step 1. Stop after a fixed number of iterations or after all of the training images have been added.

Typically, once an image has been added to the training set, it is not removed, and the values of the latent variables are fixed.

5.3 Detector Stages and Training Process

A one stage detector was employed in this set of experiments. Given the long training times, which are discussed in Section 5.8.1, a single stage detector was employed to facilitate many repetitions of the training process. Abstracting away the specific details already presented, there are two distinct steps in training the detector for a single stage. The first is feature selection, and the second is training the weights of the classifier. In all of the experiments, the first step consisted of training the *base model.* The *base model* was trained with a small set of fully labeled data.

This base model was then applied to the weakly labeled data set. The detector generated a set of detections for each image. Each detection consisted of position, scale, and confidence information. This information was used to select a set of weakly labeled images to add to the training set. Images were selected according to the confidence value output by the classifier of the detector. A single detection, the highest confidence detection, was selected from each weakly labeled training image. Then, typically, the twenty detections with the highest selection scores were chosen and added to the training set.

The next step was to train a new detector using this augmented data set consisting of the original fully labeled data and the newly labeled weakly labeled data. The process can then be repeated, and an additional set of weakly labeled images selected to be added to the training set. Note that once a weakly labeled image is "labeled," it is removed from consideration in subsequent iterations, and the values of the latent variables, the position and scale, are kept fixed. It should be noted that we only selected a single detection per training image, even though there were typically more than one object in each of our training images. An enhancement would be to mark a part of an image as "consumed" and then allow for detections outside of that region.

For the purpose of the experiments, a fixed number of iterations was performed, typically until all of the images in the weakly labeled data set were added, and reported performance was that of the best iteration. When used in an actual application, a small cross validation set consisting of a small number of labeled images can be used to determine at which iteration to stop. At the end of training, the labeled images in the cross validation set can be added to the training set for one final training iteration, so that labeled data is still utilized.

Two different training procedures were used. In the first experimental procedure, the detector was re-trained from the beginning at each iteration; both the feature selection process and the classifier training were run. In the second experimental procedure, the full training procedure was only completed for the base model. The feature set was fixed after that point. In subsequent iterations, which incorporate weakly labeled data, only the classifier was re-trained. This second approach was used for two reasons. The first was to reduce the variance of the result. As will be discussed in following sections, the feature selection process seemed to be very sensitive to the specific images in the training set when the training set size was small. This complicated the evaluation of the effect of weakly labeled data because the performance effects from the additional training images and from the selection of a different set of features could not be separated. A second advantage of only retraining the classifier was reduced training time, since the feature selection process only needed to be run for the base model.

It should also be noted that, when a weakly labeled image is added to the training set, it actually represents more than one training image. In the specific configuration of the Schneiderman detector used for the experiments here, 80 synthetic variations are used. The synthetic variations vary the position, scale, and rotation of the training images by small random amounts so that the detector will be less sensitive to small variations in appearance, as was also done in [Pomerleau 91]. The same synthetic variations are also applied to all of the fully labeled training data.

One important experimental detail is the partitioning of the data set during training. In the standard fully labeled training process of this detector, a separate cross validation set is used for two purposes. First the feature selection process uses the cross validation set. Here, wavelet coefficients are formed into features. Groups of features are then selected as input to the classifier. Multiple candidate groups are generated using the training data during this process. A second step uses cross validation to decide which of these groups has the most discriminative power. Because a limited set of fully labeled data is available, two versions of this cross validation process were explored. The first alternative is the same as the standard fully labeled version. The fully labeled data set is split into two pieces; one is used in the training process and the other is used for cross validation. The advantage of this approach is that true cross validation is being performed using a separate labeled test set. The disadvantage of this approach is that, with a limited amount of training data, the amount of data being used for the actual training process is reduced. Another approach used the fully labeled data set for training, and a small subset of the training set was designated to be the cross validation set. This has an advantage in that the labeled training data is not being partitioned, so that all of it is being utilized in the training process. The disadvantage is that, since the training set is being used for cross validation, true cross validation is not taking place. However, in preliminary experiments it was found that the final performance of the detector was not adversely affected by performing the feature group selection in this manner.

A cross validation set is also used in the training process during the training of the classifier. It is used here to select the iteration at which the Adaboost training process stops. The same advantages and disadvantages apply in this situation. In this case, the full data set was used for training Adaboost, and a subset of the training set was used for cross validation. Again, it was found in preliminary experiments that the final performance of the detector was not adversely affected by performing the feature group selection in this manner.

It should also be noted that, in both cases, these pseudo cross validation sets consisted only of the fully labeled data; none of the weakly labeled data was utilized.

To summarize, we will explore the following issues and how they affect performance through a series of experiments:

- sensitivity of the detector, trained with only fully labeled data, to training set size and number of features
- weakly labeled data performance for the confidence-based selection metric
- weakly labeled data performance for the MSE-based selection metric
- weakly labeled data performance in relation to rotation estimation and synthetic rotation variation
- weakly labeled data performance for a variety of other detector issues: feature count, Adaboost iterations

5.4 Performance Evaluation Metrics

One issue that always arises when comparing detection systems (or classifiers in general) is how to evaluate their performance in isolation, and specifically, how to compare the performance of different detectors. Of course, the detector needs to be evaluated on a separate test set so that an accurate measure of performance can be extracted, since it is often possible with a simple system to memorize the training set and achieve maximum performance. Also, a reasonable metric must be chosen.

In detection scenarios, a simple measure of accuracy is often not useful. This is because instances of the object are rare, so maximum accuracy can be achieved by a system which always answers that the object is not present. Another possible metric is detection rate: the ratio of the number of true detections detected to the number of instances of the object in the test set. The problem here is that a detector which returns all possible locations as object locations will achieve a 100% detection rate, but will have many false positives.

A metric is needed which balances between true and false positive rates. If the detector is meant to be used in a specific application, then a cost can be assigned to true positives and false positives. It is then possible to pick an operating point for the detector which minimizes the cost or risk. However, in the absence of a cost function, other metrics are often used.

One such measure is the area under the ROC curve. The ROC curve is the receiver-operator-characteristic curve, which typically plots detection rate on the vertical axis and false positive rate on the horizontal axis. The ideal detector has 100% detection rate at any false positive rate, including a 0% and 100% false positive rate, and hence has maximum area under the curve. Therefore any detector whose ROC curve is above the ROC curve for another is better; it achieves a better detection rate at every possible false positive rate. A single statistic which captures this is the area under the ROC curve (AUC). A detector which has greater area under the ROC curve will, on average have a higher detection rate, over the false positive rates examined. The AUC metric was also chosen because it is consistent with the approach taken internally by the Schneiderman detector. The Schneiderman detector uses AUC to judge the relative merit of different models during the cross validation phases.

Since we are interested in whether the addition of weakly labeled data has improved the performance of the detector over the base model, which utilizes only the fully labeled data, we choose to use the ratio of the AUC of the base model to the AUC of the model being evaluated as a measure of the model performance. This gives us a rough measure of whether the performance of the model is increasing or decreasing with the addition of weakly labeled data.

Another common way of comparing detector performance is to choose a specific detection rate and compare the number of false positives (or the false positive rate) for the detector under evaluation. This measure has merit in that it takes both the detection rate and the false positive rate into account. The problem with this approach is that a specific detector may perform better than others at a specific detection rate and the detection rate chosen as the basis for comparison may not be the one that is actually used.

5.5 Experiment Specifics

The object chosen for these experiments is a human eye as seen in a full or near frontal face. A single detector was trained to detect either the left or right eye. In each fully labeled example, four landmark locations on the eye were labeled: the left most point, the right most point, the middle upper most point, and the middle lower most point. An example of this is shown in Figure 45 where the landmark locations are marked with light colored diamonds. This information provides detailed information about the scale, position, and orientation information of each training example. The region (or regions) of the training image identified by the labeled training data were rotated to a canonical orientation, scaled and cropped to result in

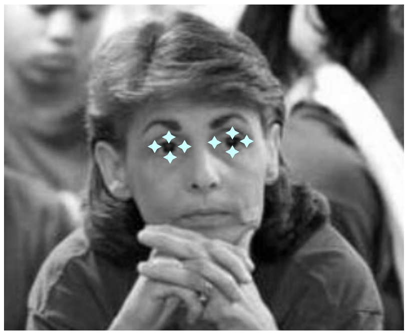


Figure 45: Training data labels consisted of marking four landmark locations on each eye, as illustrated by the light colored diamonds in this image.

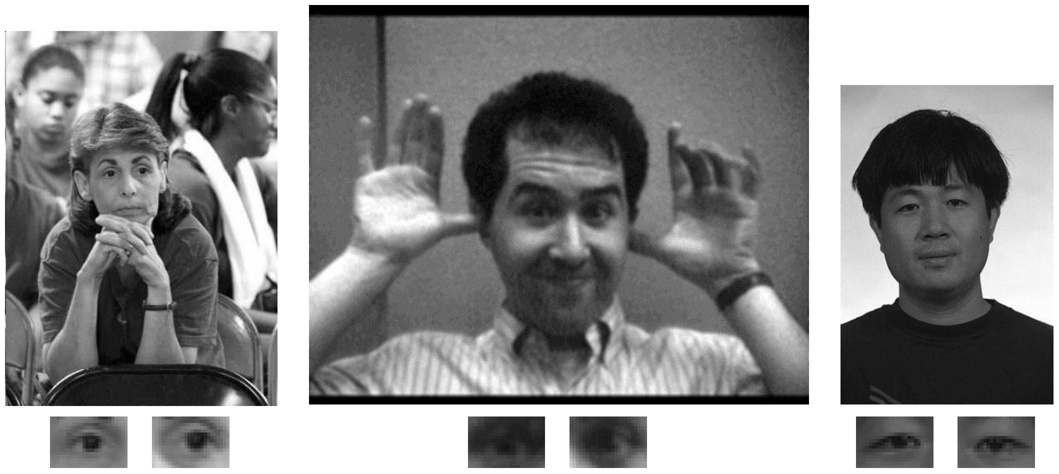


Figure 46: Sample training images and the training examples associated with them.

a training example image 16 pixels wide and 24 pixels high. Example training images and the corresponding training examples extracted from them can be seen in Figure 46.

The full set of labeled training images consists of 231 images. In each of these images, there were from two to six training examples per image for a total of 480 training examples. The independent test set consisted of 44 images. In each of these images, there were from two to 10 testing examples for a total of 102 test examples. In both the training set and the test set, some of the images contained only a single person against a blank background and, in some instances, one or more people were present against a cluttered background. Examples can be seen in Figure 46. Obviously, the nature and amount of "clutter" increases the "difficulty" for the detector. In the context of semi-supervised training, the "difficulty" of the training set is just as important as the test set in terms of the final measured performance, since the detector will be used to identify training instances in these images. Obviously "easier" images will make this an easier task for the detector. This suggests that the "labeling" of weakly labeled data might have additional aspects in addition to indicating the presence or absence of an object. One possibility is that only "easier" images could be chosen for training, images could be rated according to their difficulty, or some sort of coarse masking could be used. These issues were not explored here, but would be an interesting avenue for future research.

In the experiments described here, typically 25 fully labeled images were used for "cross validation" during the feature selection and/or the Adaboost training process. In some experiments those images were segregated, as is typically done with cross validation data sets. In another case, the up to 25 “cross validation” images used were a subset of the fully labeled data set used for training. In these cases it would be more appropriate to call the “cross validation” sets training sets. In the cases where the total labeled training set was smaller than 25 images, the entire data set was chosen. This number of images was chosen because a set of preliminary experiments showed little difference in final detector performance even when the cross validation set was shrunk down to this size.

The training and test images were typically in the range of 200-300 pixels high and 300-400 pixels wide. The eye itself ranged from being quite small, occupying 16×8 pixels, to quite large, occupying 72×48 pixels. Given the ground truth information, all eyes were scaled to 24×16 pixels. Scale invariance is achieved during detector use by scaling the image during the detection process.

During the course of these experiments, we found that there was quite a bit of variance in final performance and the behavior of the semi-supervised training process. Much of this variance arose from the specific set of images randomly selected in the initial training subset. To overcome this limitation, each experiment was repeated using a different initial random subset. We call a specific set of experimental conditions an *experiment*, and each repetition of that experiment we call a *run*. In most cases 5 runs were performed for each experiment. In cases where the result was ambiguous or especially promising, additional runs were performed to increase the certainty of the result.

Another set of parameters controlled the synthetic variation of training examples. Synthetic variation included random variation in position, size, and orientation. In all of the experiments, a total of 80 synthetic variations were created of each training example. This was done for both the fully labeled and weakly labeled data used in the training process. For all experiments the positional variation was ± 0.5 pixels and the size variation was in the fixed range of 0.945 to 1.055 of the original size. In most experiments the variation was ± 12 degrees; however, a small number of experiments also utilized ± 6 degrees or ± 3 degrees.

It is also important to note that the performance of this detector is not as high as that of the full Schneiderman detector because it only uses one stage and a limited number of features.

One of the challenges in performing this set of experiments is that training this detector is time consuming, taking on the order of twelve hours on 3.0 GHz level machines. This is an issue because the training of the detector is the inner loop of the algorithm. If the detector is trained during 10 iterations and 5 repetitions of an experiment are performed, then each experiment takes $12 \times 10 \times 5 = 600$ hours of compute time, or 25 days. Thankfully, the five repetitions can be performed in parallel, so wall time is more like $12 \times 10 = 120$ hours of compute time, or 5 days. As different parameters of the detector, the weakly labeled data approach, and their interactions needed to be explored, the magnitude of the amount of compute time necessary for this investigation becomes immense. Through the generosity of various research groups and individuals at CMU listed in the Acknowledgments, approximately 30-40 CPU’s were used simultaneously for this investigation. However, the logistics of managing a large number of long running experiments, given heterogeneous computer configurations and hardware failures, proved to be quite challenging.

Another parameter of the experiments is the number of images added at each iteration. Ideally, only a single image would be added at each iteration. However, because of the time it takes to train the detector and the goal of reducing the number of iterations, more than one image was added at each iteration. Adding more images reduces the average training time per weakly labeled image, but this increases the chance that there will be an incorrect detection included in the weakly labeled data set. Typically 20 weakly labeled images were added to the training set at each iteration of the experiment.

5.6 Experimental Variations

An attempt was made to use the detector in its original unmodified form for the purpose of these experiments. As much as possible, the addition of weakly labeled data was executed as a "wrapper" to the standard training process, the hope being that any detector could be used in the inner loop and be readily adapted to the semi-supervised training process. However, this was not as straightforward as it might seem because there can be subtle interactions between the detector and the unlabeled data process, and there are many parameter choices for both training and running the detector. The parameter settings for running the detector can become critical because in the semi-supervised training process, the output of the detector is now part of the training process.

Some of the parameters that were experimented with in the training of the detector follow: amount of scale invariance, number of synthetic variations, number of non-object examples, number of features in the classifier, number of cross validation images, partitioning of the training set into cross validation and training sets, keeping the features fixed or allowing them to vary during the semi-supervised training process.

One of the issues that arose was that the output of the detector, when used to label weakly labeled data, may need to be more accurate than is needed for other purposes; specifically scale and rotation accuracy can be important. In early experiments, we explored the resolution of the scale search during detection. One of the issues was that the set of training examples provided defined the concept of the object. When we provide carefully labeled training data, we specifically define that concept which can be extended by introducing controlled synthetic variation and specifically designing the detector to do a systematic search over the variation. If the parameters of the variation, e.g. rotation or scale or shift, are not estimated, or are only coarsely estimated, then the "concept" of the object will be extended. An example, which is the case in some of the experiments described here, is a detector which is able to detect eyes that are ± 12 degrees in orientation from horizontal. If the detector does not output orientation information, and the detected eyes are added to the training data, then the orientation range which the detector will need to handle will increase with each iteration. Typically one would want to have a detector which is as invariant as possible to the orientation of the object which is to be detected. However, many detectors also have a certain amount of representational or modeling capacity. That is, many detectors can handle up to a certain amount of appearance variation. The incorporation of unlabeled data should not push the detector past those limits.

Since the detector we are using does not explicitly output orientation information, we decided to implement this as a "wrapper" around the detector for the purpose of labeling weakly labeled data. To this end, we generated rotated versions of the weakly labeled data, specifically ± 4, ± 8, and ± 12. Including the original orientation, this is a total of seven variations of each image. This allows us to extract some orientation information from the detector. By selecting the "best" detection amongst the seven images, we hope to get a detection as close as possible to the "neutral" orientation of the object. Operationally, we ran the detector on each variation of the training image. We selected the highest confidence detection per image, and then the best detection per set, using either the highest confidence or "score," depending on which was the figure of merit for the specific experiment.

A subset of the experimental parameters have been assigned code names. These codes are of the form stsXX, where XX is a two digit value. These are detailed in Table 8.

Another important parameter is the number of fully labeled training images. A number of experiments were performed with varying percentages of the original fully labeled training set. The number of training images has an extremely large effect on the outcome of the experiments described here. If there is too much fully labeled data, the parameters of the detector can be accurately estimated and weakly labeled data will not

Name	Number of Features	Non-Object Examples	Object Variations	Angle Variation
sts00	2	15000	80	±12
sts04	5	15000	80	±12
sts05	10	15000	80	±12
sts06	15	15000	80	±12
sts07	20	15000	80	±12
sts08	15	15000	80	±6
sts09	15	15000	80	±3
sts10	15	15000	80	±20

Table 8: This table details the conditions used for various experiments. Each condition was designated by an "stsXX" name.

help. If there is too little labeled data, the detector will not be powerful enough to extract anything useful out of the weakly labeled data.

5.7 Experimental Protocol

A small number of labeled training examples will cause the initial performance of the detector trained with only the fully labeled data, and the final performance when weakly labeled data is incorporated, to be highly variable. Sometimes the random labeled data set is a "lucky" draw and performance can be as good as the full data set performance and sometimes the draw is very "unlucky" and performance is abysmal. Because we would like to assign a significance to the relative performance of different approaches, each experiment was repeated at least five times using the same parameters with different random initial fully labeled data subsets. As mentioned in a previous section, each "run" was evaluated utilizing either the area under the ROC curve (AUC) or the false positive count at a specific detection rate. Because different experimental conditions change performance, performance measures were normalized relative to the full data performance of that run. So a reported performance level of 1.0 would mean that the model being evaluated has the same performance as it would if all of the labeled data was utilized. A value of less than 1.0 would mean that the model has a lower performance than that achieved with the full data set. To compute the full data performance, each specific run is trained with the full data set and its performance recorded. The performance from all of the runs of a specific experiment are aggregated and we compute a single set of performance measures: the mean, the standard deviation, and the 95% significance interval. The mean and the standard deviation give us a measure of the expected performance level and its variation. The 95% significance interval is computed as the mean plus and minus 1.64 times the standard error of the mean. This is useful for determining whether the performance difference between two sets of experimental conditions is significant. One should note that the significance interval can be shrunk to zero size, given a large enough number of experimental runs, so it is not a good measure of the inherent variation of the performance. For this reason, when we present error bars on plots, we use them to present one of, or a combination of, the performance measures, depending on the point that we are trying to present in a specific plot. Standard deviation measures the inherent variation, irrespective of the number of experimental runs. The 95% significance interval determines whether the performance is significantly different between two sets of experimental conditions. Finally, the standard deviation and the 95% significance interval may both be presented. In this case, the two sets of error bars are super-imposed, where the outer set is the standard deviation and the inner set is the 95% significance interval. It is therefore important to look at the caption of each plot to understand what the errors bars in that specific plot represent.

The performance reported here is what we call "best iteration performance". This is the best performance on the test set over all of the iterations in which unlabeled data is added. It is optimistic because it uses the actual test set performance to decide which iteration to choose. Ideally, some function of training set performance could be used as a metric to decide which iteration to choose. A set of preliminary experiments, which examined functions of the detector confidence and MSE score on both the weakly labeled data and the fully labeled initial training set, did not result in any promising measures. An alternate possibility is to use a small cross validation set. We leave this possibility to future work. One issue, of course, is that the cross validation set must be small since we are operating under the scenario that fully labeled data are scarce and expensive. One possible way to get around "wasting" labeled data on a cross validation set would be to use an approach similar to N-fold cross validation. This approach would take advantage of the fact that, in the semi-supervised training process, the weakly labeled data are assigned labels. Different splits of the data into training and cross validation sets could be performed. The semi-supervised training procedure could be repeated for each of these splits. In a final round of training, labelings of the weakly labeled data from all of these splits could be combined, and a final model could be trained using this data. Again, we leave this to future work.

It is useful to list the steps involved in a typical experiment run:

1. A "base model" with a random subset of fully labeled data is trained.
2. A "full data model" is trained for normalization purposes to compute the best performance.
3. A "weakly labeled model" is trained, and its performance is compared relative to the "full data model".

5.8 Analysis of Experimental Results

5.8.1 List of Experiments

Table 9 contains an overview of the experiments performed under various conditions using the Schneiderman detector. CPU Days was computed assuming 12 hours per detector training execution and assuming 10 iterations per semi-supervised training run. It is interesting to note that a total of 1072 estimated CPU days was used, which is 3 years. (It should be noted that this is a significant underestimate because it does not include the many debugging runs during code development and other early runs to characterize detector behavior.) Under ideal conditions, using 36 machines simultaneously with 100% utilization, this would only take 1 month of calendar time. Of course, 100% utilization is next to impossible to achieve in practice. The bulk of the experiments described here were completed over a period of 5 months.

5.8.2 Sensitivity to Fully Labeled Data Set Size and Number of Features

The first set of experiments was performed under the "cv025" experimental conditions. Under these conditions, a random subset of 25 images from the randomly selected training set is utilized, where a cross validation set would typically be used in the training process. These images are still used in the training process. This means that, in effect, there is no cross validation set. This was done for a number of reasons.

Condition	Base Sample Rates : Run Count	Semi-Supervised Sample Rates				Approx. Train Count / CPU Days
		Conf.	MSE	Rot	MSE+Rot	
cv025 sts00	1,2,3,4,6,8:5 ; 5:10					40 / 20.0
cv025 sts04	1,5:5 ; 8:6					16 / 32.0
cv025 sts05	1,8,12:5	8				70 / 35.0
cv025 sts06	1-6:5 ; 8,10,12,14,15,16,20:10	8,10,12, 14,16,20	8,10,12, 14,16,20	12	12	1460 / 730.0
cv025 sts07	1,8,10,12:5	8				75 / 37.5
cv025 sts08	12:5	12	12	12	12	50 / 25.0
cv025 sts09	12:5	12	12	12	12	50/ 25.0
cv025 sts10	12:5	12	12	12	12	50 / 25.0
scva025 sts06	1,2,3,4,5:5 ; 6:10	3,4,5,6	4			355 / 177.5

Table 9: This table is an overview of all of the experiments performed under various conditions using the Schneiderman detector. The train count is approximately the total number of times the detector was trained for that set of experimental conditions, assuming 10 iterations per semi-supervised run. CPU Days was computed assuming 12 hours per detector training execution. The total estimated number of CPU days is 1072, which is 3 years. Of course, under ideal conditions using 36 machines simultaneously 100% of the time this would only take 1 month of calendar time. Of course 100% utilization is next to impossible to achieve in practice.

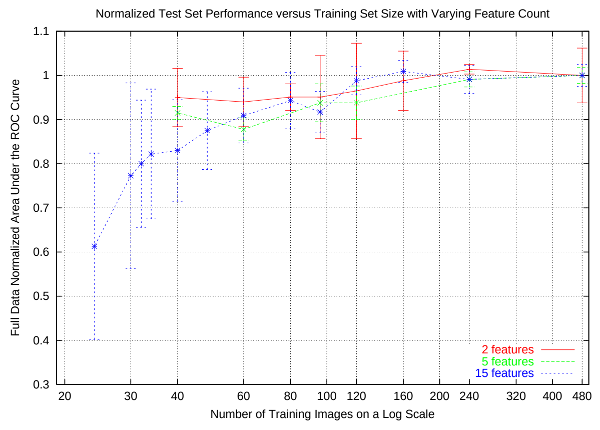
The first was that, in the semi-supervised training scenario where labeled data is limited, maintaining a cross validation set may not be the best use of the limited labeled data available. The second was that, in preliminary experiments with the full data set, training the detector in this way did not seem to have an effect on test set performance. Finally, a small set of images (25) was used to decrease training time. Again, preliminary experiments with the full data set indicated that 25 images was sufficient, and using a larger set of images did not yield improved test set performance.

The issues we will explore through the set of experiments detailed in this section are:

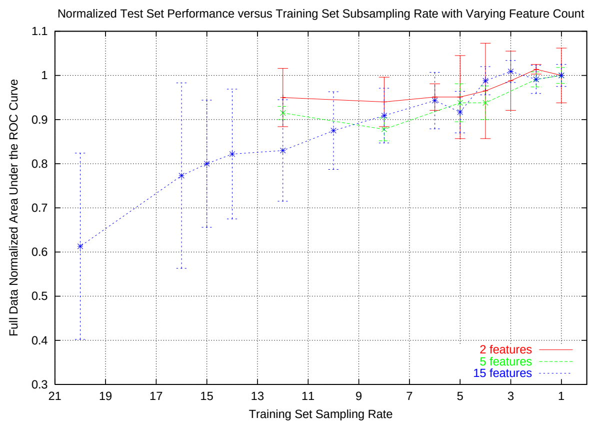
1. What is a reasonable number of features to select for the Schneiderman detector in our subsequent experiments? This is based on experiments which measure how feature count affects performance as fully labeled data set size varies.

2. What is a reasonable data set size to use for our weakly labeled experiments, based on experiments that measure how performance changes as the fully labeled data set size varies? We need to select a data set size small enough for additional data to improve performance and large enough for the semi-supervised training approach to extract useful information.

The first step in our experiments is to choose specific parameter values for the Schneiderman detector and the conditions under which we will use that detector. The set of experiments described in this section investigates sensitivity to training set size and to the number of features used in the detector. It is very important to characterize sensitivity to training set size because we want to perform our experiments under conditions where the addition of weakly labeled data will make a difference. If the performance of the



(a)



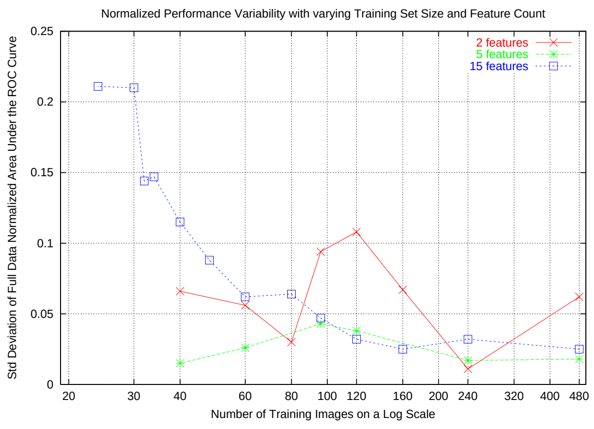
(b)

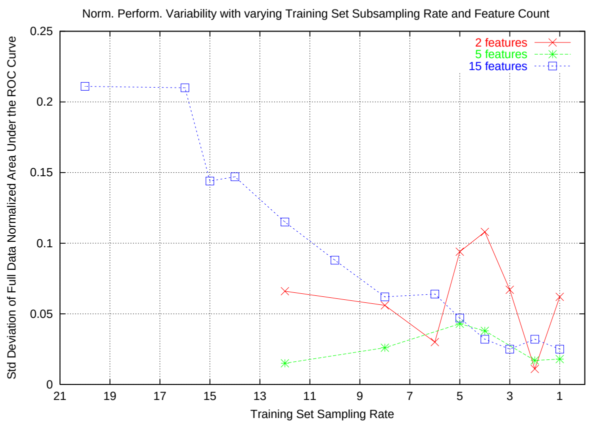
Figure 47: These figures plot the normalized AUC performance of the detector as fully labeled training set size varies using detectors with different numbers of features. Subfigure (a) plots normalized AUC performance versus training set size on a log scale and subfigure (b) plots normalized AUC performance versus training set sampling rate. In both figures the inner error bars indicate the 95% significance interval, and the outer error bars indicate the standard deviation of the mean.

detector is already at its maximum, given a labeled training set of a specific size, then we cannot expect weakly labeled data to help. The second issue is the number of features. We would like to choose a sufficiently large number of features such that we obtain a minimum level of stable performance, but too many would make training time and space requirements unmanageable.

To characterize the number of features, we trained a number of base models using 2 (code sts=00), 5 (code sts=04), and 15 (code sts=06) features. The results of these experiments are plotted in Figure 47 (a) and (b). The data presented is the full data normalized AUC performance. Subfigures (a) and (b) both plot the same performance data, but in the case of (a), the x-axis is presented as the number of training images on a log scale. In (b) the x-axis is presented as the training set sampling rate, where higher sampling rates correspond to smaller training set sizes. The error bars plotted are the standard deviation of the mean performance. Each data point and its associated error bars is computed from 5-10 runs under the same experimental conditions. As can be seen in these plots, using only two features resulted in a large variance in performance. Moving to a large number of features, like 5 or 15, greatly reduced this variance. This effect can be seen more clearly in Figure 48, which isolates the standard deviation as a performance measure and plots it on a separate set of graphs. Also, although the difference was not significant in these experiments, it appeared that performance was better when using 15 features versus 5 features. For these reasons, reduced performance variance and higher performance, we chose to perform the majority of our experiments using 15 features.

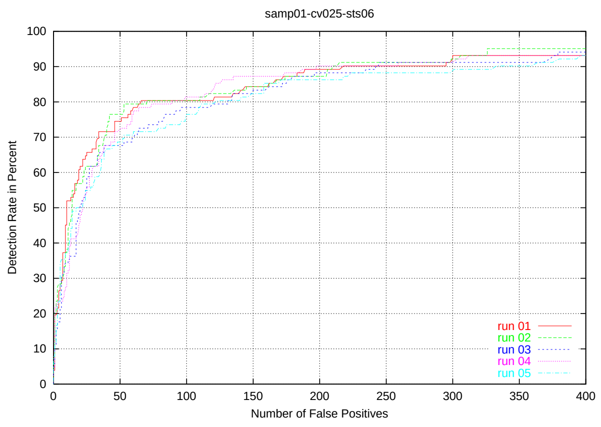
Given a specific feature count, we can now examine how performance varies with labeled training set size more closely. The details of the performance from the detector trained with labeled data with 15 features under the “cv025 sts06” conditions are presented in Figure 50 and Table 10. In subfigure (a) of Figure 50 the results are presented as the full data normalized AUC on the y-axis vs. training set size on a log scale on the x-axis. Subfigure (b) reports the same performance results with sampling rate on the x-axis. In both cases the inner error bars plot the 95% significance interval, and the outer error bars plot the standard deviation of the mean.

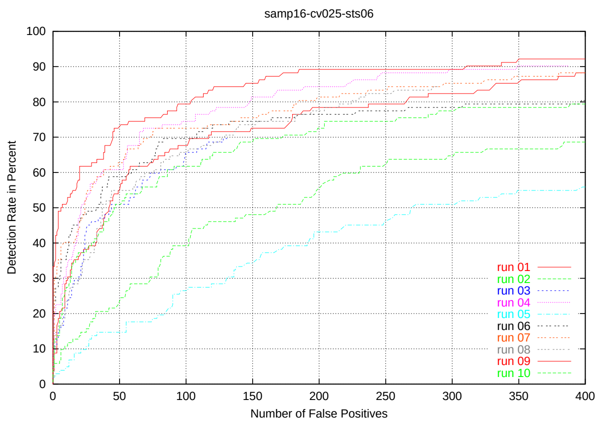




(a) (b)

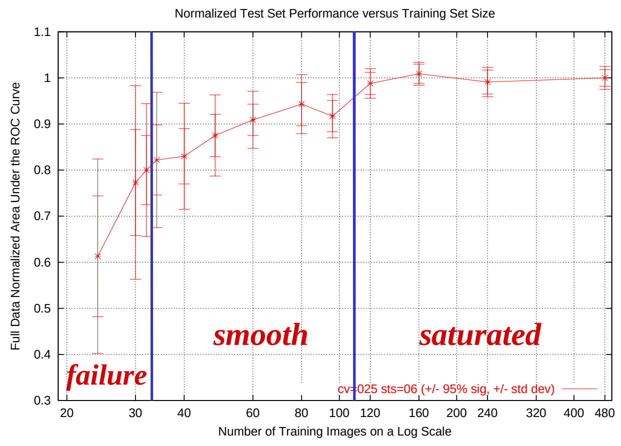
Figure 48: These figures plot the variation in the normalized AUC performance of the detector, measured as the standard deviation, as fully labeled training set size varies using detectors with different numbers of features. Subfigure (a) plots the standard deviation normalized AUC performance versus training set size on a log scale and subfigure (b) plots normalized AUC performance variation versus training set sampling rate.



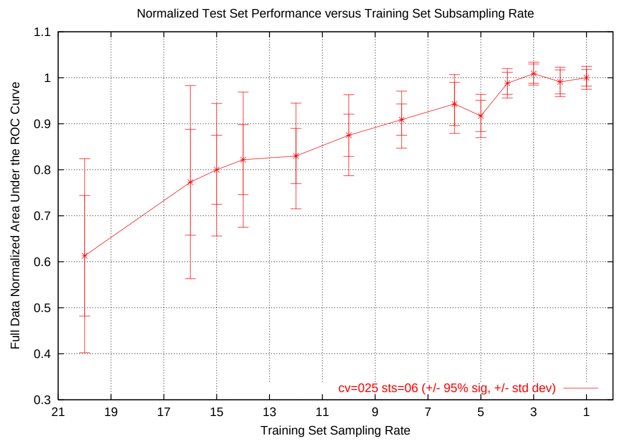


(a) (b)

Figure 49: Examples of low and high variance runs under the “cv025 sts06” condition. Both subfigures contain plots of multiple ROC curves for different experimental runs consisting of different fully / weakly labeled data splits. Subfigure (a) is for five runs of the full data set and (b) is for ten runs of 1/16 of the data set. The high variation in performance introduced by small data set size can be seen in figure (b).



(a)



(b)

Figure 50: These figures plot the normalized AUC performance of the detector as fully labeled training set size varies under the “cv025 sts06” condition, where fifteen features are used. Subfigure (a) plots AUC performance versus training set size on a log scale with the three regimes of operation labeled. Subfigure (b) plots AUC performance versus training set sampling rate. In both figures the inner error bars indicate the 95% significance interval, and the outer error bars indicate the standard deviation of the mean.

The first thing to notice is that the variability of the detector performance grows as training set size decreases. This can be seen more clearly in Figure 48 which plots the standard deviation separately. In Figure 48a we can observe that the standard deviation grows extremely quickly as the training set size decreases. This effect can also be seen in Figure 49 which plots the ROC curves for individual runs for a sampling rate of 1 in Figure 49a and a sampling rate of 16 in Figure 49b. Because of the random nature of selecting the cross validation set and other parameters, there is still some performance variation even when the full data set is used. However, the figure makes it clear that the variation is much higher when the training set size is small. It is interesting to observe the range of performance achieved with the smaller training set. The best, “lucky,” runs can exhibit performance as good as the runs trained with the full data set.

Figure 50a plots the normalized AUC performance versus training set size on a log scale. This demonstrates how the expected performance varies with training set size. Our interpretation of this data is that is there are three regimes in which the training process operates. We call the first the “saturated” regime, which in this case appears to be from approximately 160 to 480 training examples. In this regime, 160 images are sufficient for the detector to learn the requisite parameters; more data does not result in a boost in performance. Similarly, variation in performance also seems to be relatively constant and small in this range, which can be seen in Figure 48. We call the second regime the “smooth” regime which appears in this case to be between 35 and 160 training examples. In this regime, performance seems to decrease relatively smoothly as training set size decreases and appears, in these experiments, to be relatively linear when plotted on a log scale as in Figure 50a. Similarly, the standard deviation of the performance seems to increase smoothly in this regime. In Figure 48a, it appears that the standard deviation grows less quickly than log linearly. In Figure 48b it appears to grow linearly with the sampling rate, which is the inverse of the training set size. The third regime we call the “failure” regime. In this regime there is both a precipitous drop in performance and a very large increase in performance variation. In these experiment this happens when the training set size starts to drop below 35 examples. We assume that this third regime is caused by a

Sample Rate	Data Count	Mean AUC	Norm AUC	Std Dev	Std Err * 1.64	Run Count
1	480	150.42	1.000	0.025	0.018	5
2	240	149.07	0.991	0.032	0.026	4
3	160	151.83	1.009	0.025	0.021	4
4	120	148.54	0.988	0.032	0.024	5
5	96	137.95	0.917	0.047	0.034	5
6	80	141.90	0.943	0.064	0.047	5
8	60	136.75	0.909	0.062	0.034	9
10	48	131.56	0.875	0.088	0.046	10
12	40	124.87	0.830	0.115	0.060	10
14	34	123.59	0.822	0.147	0.076	10
15	32	120.33	0.800	0.144	0.075	10
16	30	116.25	0.773	0.210	0.115	9
20	24	92.20	0.613	0.211	0.131	7

Table 10: This table summarizes the AUC performance of the fifteen feature detector under the "cv025 sts06" condition as fully labeled training set size was varied.

Sample Rate	Data Count	Mean AUC	Norm AUC	Std Dev	Std Err * 1.64	Run Count
1	480	325.96	1.000	0.014	0.010	5
2	240	322.75	0.990	0.024	0.018	5
3	160	317.75	0.975	0.031	0.023	5
4	120	304.20	0.933	0.053	0.039	5
5	96	309.21	0.949	0.037	0.027	5
6	80	312.18	0.958	0.038	0.020	10
8	60	284.66	0.873	0.074	0.049	6

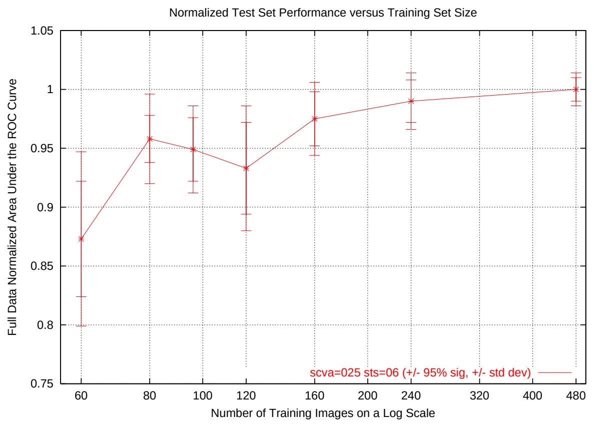
Table 11: This table summarizes the AUC performance of the fifteen feature detector under the "scva025 sts06" condition as fully labeled training set size was varied.

situation where the training algorithm does not have sufficient data to estimate some set of parameters. An extreme case of this would be when the parameter estimation problem is ill conditioned.

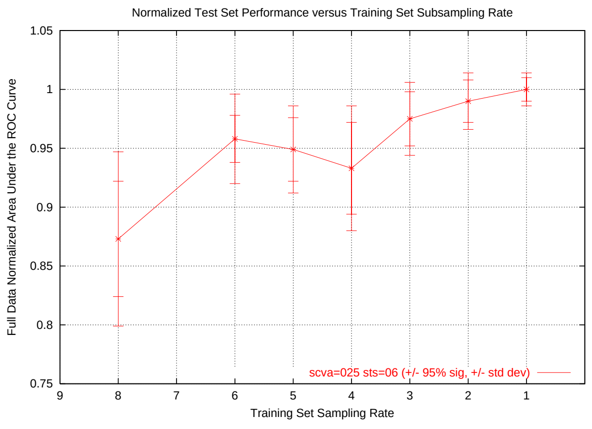
Another set of experiments was also performed under the "scva025 sts06" conditions where the training set is partitioned so that both the feature set and the classifier can be retrained when weakly labeled data is used. The results of these experiments and are presented in Figure 51 and Table 11. Because the training set is partitioned, a higher variation in performance is seen for the same amount of data. This large amount of variation makes it difficult to clearly distinguish the three regimes of training set size seen in prior experiments.

5.8.3 Weakly Labeled Data Performance

In this section we perform a series of experiments to explore the performance of self-training when applied to the Schneiderman detector. In these experiments, weakly labeled data is utilized and the confidence of the detector is used to decide which new examples to add at each iteration. The details of these results are

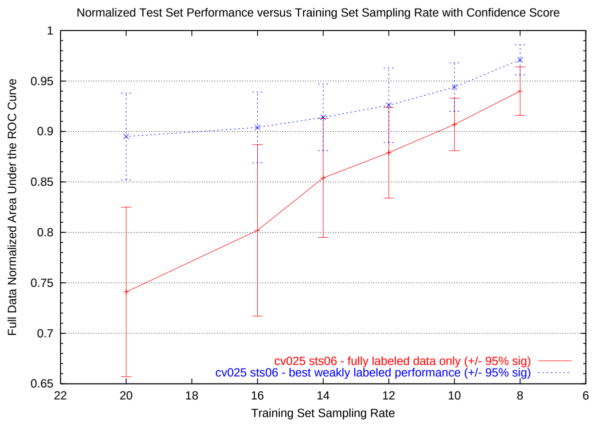


(a)

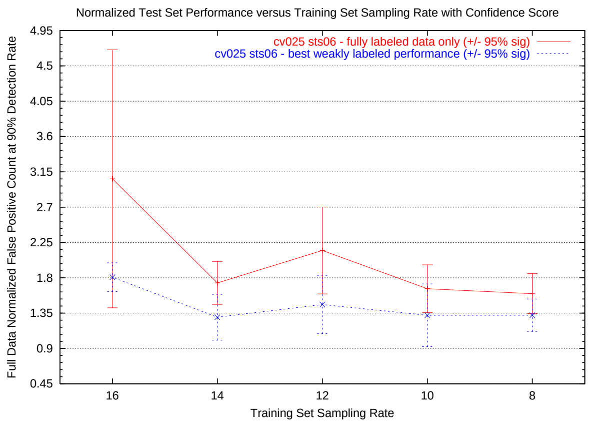


(b)

Figure 51: These figures plot the normalized AUC performance of the detector as fully labeled training set size varies under the “scva025 sts06” condition, where fifteen features are used. Subfigure (a) plots AUC performance versus training set size on a log scale and subfigure (b) plots AUC performance versus training set sampling rate. In both figures the inner error bars indicate the 95% significance interval, and the outer error bars indicate the standard deviation of the mean.



(a)



(b)

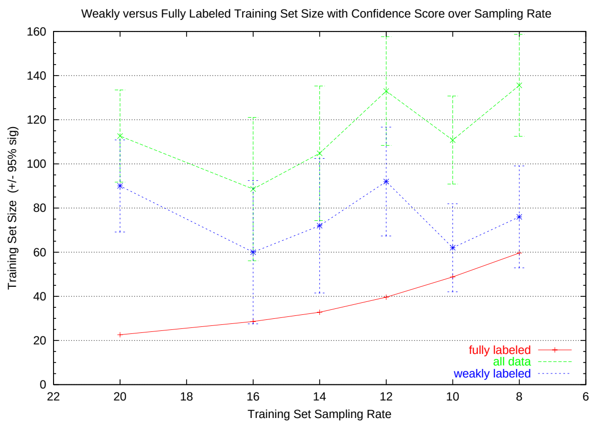
Figure 52: This figure plots the normalized performance of the detector, incorporating weakly labeled data and using the confidence metric, as the fully labeled training set size varies under the “cv025 sts06” condition, where fifteen features are used. Subfigure (a) plots the normalized AUC performance. The bottom plot line is the performance with labeled data only, and the top plot line is the performance with the addition of weakly labeled data. Subfigure (b) plots the normalized false positive count performance at a detection rate of 90%. The top plot line is the performance with labeled data only and the bottom plot line is the performance with the addition of weakly labeled data. The error bars indicate the 95% significance interval of the mean value.

Sample Rate	Data Count	Full Norm AUC Start		Full Norm AUC Best				Run Count
		Mean	95% Interval	Mean	95% Interval	Std Dev	Std Err	
8	60	0.940	0.916-0.964	0.971	0.956-0.986	0.028	0.009	10
10	48	0.907	0.881-0.932	0.944	0.920-0.968	0.046	0.015	10
12	40	0.879	0.835-0.924	0.926	0.890-0.963	0.071	0.022	10
14	34	0.854	0.794-0.913	0.914	0.880-0.947	0.064	0.020	10
16	30	0.802	0.718-0.887	0.904	0.869-0.939	0.068	0.021	10
20	24	0.741	0.657-0.825	0.895	0.852-0.938	0.083	0.026	10

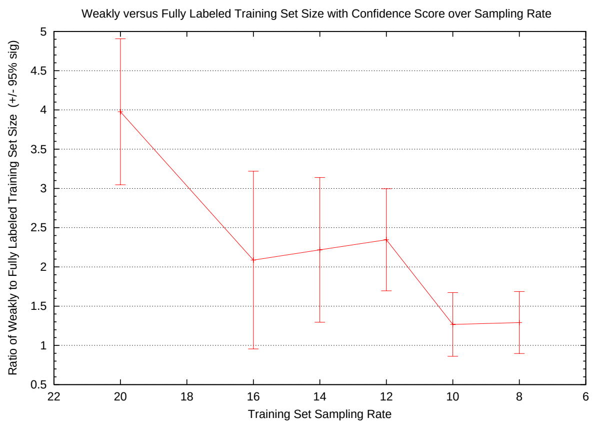
Table 12: This table summarizes the normalized AUC performance of the fifteen feature detector under the “cv025 sts06” condition. Weakly labeled data was incorporated using the confidence metric and fully labeled training set size was varied. The 95% significance interval is computed as 1.64 times the standard error.

presented in Figure 52 and Table 12. Figure 52a plots performance as measured by full data normalized AUC on the y-axis and sampling rate on the x-axis. In this plot, larger values (a performance value of 1.0 is ideal) indicate better performance, indicating that the detector has reached the performance level that was possible with full data. The error bars on this plot indicate the 95% significance interval. Figure 52b plots “point performance,” performance as measured by the full data normalized false positive count at a detection rate of 90% on the y-axis and sampling rate on the x-axis. In this plot, smaller values indicate better performance and a performance value of 1.0 is ideal, indicating that the detector has reached the performance level that was possible with full data. The AUC performance measure is less noisy, thus providing a measure of performance over the entire possible range of operating points. The “point performance” measure is more noisy, but provides an idea of performance at a specific operating point. From the plots of both measures in Figure 52, it can be seen that the mean performance with the addition of weakly labeled data seems to always outperform the performance with fully labeled data alone. However, this difference is not significant at the 95% level given the inherent variance in the performance and the number of experiments performed. It is also useful to note that, in the case of the normalized AUC performance, performance does decrease as initial fully labeled training set size decreases, so that has an effect on performance. However, the slope of that decrease is smaller than the decrease in performance from the fully labeled data alone. This is important because the final solution with the inclusion of weakly labeled data reduces the reliance on the fully labeled data. Also, the variance of the performance has been reduced when compared to the results from fully labeled data alone.

Another useful way to analyze the incorporation of weakly labeled data is to examine the relative proportions of fully labeled and weakly labeled data in the best performing solutions. This data is presented in Figure 53 and Table 13. In Figure 53a the raw data counts for the fully labeled, weakly labeled, and total training set size for different initial fully labeled data set sizes are plotted. Also, error bars are presented which plot the corresponding 95% significance intervals. One can see that the amount of fully labeled data and the total data seems to be relatively constant over the base training set size. What this means, as plotted in Figure 53b, is that the ratio of weakly labeled to fully labeled data in the best performing solution increases as the fully labeled data set size decreases and the change in this ratio seems to be significant. This means that as the fully labeled base model training set gets smaller, the semi-supervised training approach is better able to utilize the weakly labeled data. A final interesting point is that the “saturated” regime as plotted in Figure 50 appears to end at training set size of about 120 images which is close to the mean of the total training set size in Figure 53a.



(a)



(b)

Figure 53: This figure plots the amount of fully labeled and weakly labeled data utilized in the best performing solutions using the confidence metric as the fully labeled training set size varies under the “cv025 sts06” condition, where fifteen features are used. Subfigure (a) plots the actual data counts. Subfigure (b) plots the ratio of weakly labeled to fully labeled data. The error bars indicate the 95% significance interval of the mean value.

Sample Rate	Data Count	Weak Count	Weak Ratio	Total Count	Run Count
8	60	76.0	1.291	135.6	10
10	48	62.0	1.267	110.8	10
12	40	92.0	2.346	133.0	10
14	34	72.0	2.217	104.8	10
16	30	60.0	2.087	88.6	10
20	24	90.0	3.977	112.6	10

Table 13: This table summarizes the normalized AUC performance of the fifteen feature detector under the “cv025 sts06” condition. Weakly labeled data was incorporated using the confidence metric and fully labeled training set size was varied. The 95% significance interval is computed as 1.64 times the standard error.

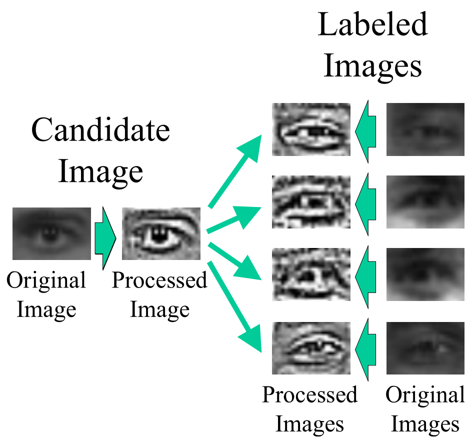


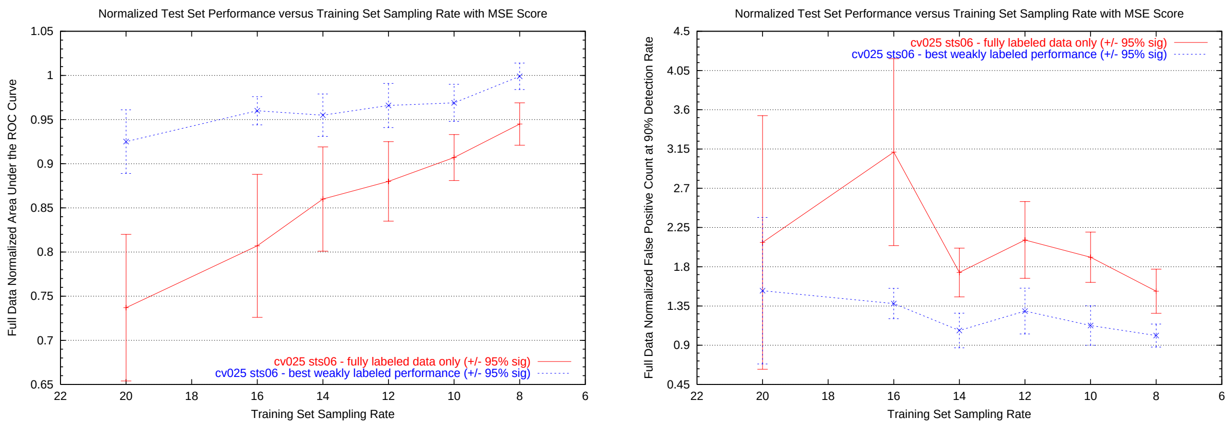
Figure 54: A schematic representation of the computation of the MSE score metric. The candidate image and the labeled images are first normalized with a specific set of processing steps before the MSE based score metric is computed.

5.8.4 MSE Scoring Metric

In this section we perform a series of experiments to explore the effect of using an alternate selection metric, the MSE metric, in the context of self-training when applied to the Schneiderman detector. As detailed in the results in previous sections of this document, we found that the selection metric used to decide which of the weakly labeled examples to add to the training set had a large effect on final performance. Typically, a measure which approximated a nearest neighbor approach seemed to be most beneficial. In the previous section, the confidence of the labels assigned to a weak example, as reported by the detector, was utilized to decide when to add an image. However, as reported earlier in this document, such a measure often has issues in that it can be inaccurate when the training set size is small. Also, such a measure may select examples that skew the distribution of labeled examples and the final detector decision boundary.

In the following experiment, we use a distance metric as the scoring function based on the mean squared error. (The distance metric was developed via experimentation on a small set of images to judge if images that appeared to be visually similar were scored by the metric.) The details of the distance computation are as follows:

1. Both images were preprocessed by high-pass filtering using a simple 3x3 kernel. This reduced the effect of local lighting variation over the images.
2. Each of the two images was separately processed such that the overall mean and variance of all the pixel values, regardless of location in the image, was normalized to zero mean and unit variance. The goal here was to reduce the effect of global lighting variation over the images.
3. The Mahalanobis distance was then computed between the preprocessed images. (The result of preprocessing steps 1 and 2 can be seen in Figure 54.) In this distance computation, a separate weight was used for each pixel position in the image. The value of the weight itself was based on the variance of the pixels at that position in the image using preprocessed images in the initial fully labeled training set.



(a) (b)

Figure 55: This figure plots the normalized performance of the detector as the fully labeled training set size varies under the “cv025 sts06” condition, where 15 features are used. Weakly labeled data is incorporated using the MSE selection metric. Subfigure (a) plots the normalized AUC performance. The bottom plot line is the performance with labeled data only, and the top plot line is the performance with the addition of weakly labeled data. Subfigure (b) plots the normalized false positive count performance at a detection rate of 90%. The top plot line is the performance with labeled data only, and the bottom plot line is the performance with the addition of weakly labeled data. The error bars indicate the 95% significance interval of the mean value.

This distance metric was used to compute a nearest neighbor score. The score assigned to a particular weakly labeled candidate was calculated by computing the distance between the candidate and all of the images in the current labeled training set, which includes both the originally fully labeled images and the weakly labeled images added to the training set in previous iterations. The candidate image is assigned a score value, which is the minimum over these distances. All of the candidate images are then sorted according their respective scores, and the m images with the minimum scores are added to the training set in that particular iteration. A schematic representation of this computation can be found in Figure 54. If we define W_i to be the weakly labeled image under consideration, j to be the index over labeled images, $L_j$to be a specific image from the set of labeled images, $g(X)$ to be the transformation performed by the image preprocessing step, and Σ to be the weights for computing the Mahalanobis distance, then the overall computation can be written as:

$$\text{Score}(W_i) = \min_j \text{Mahalanobis}\left(g(W_i), g(L_j), \Sigma\right)$$

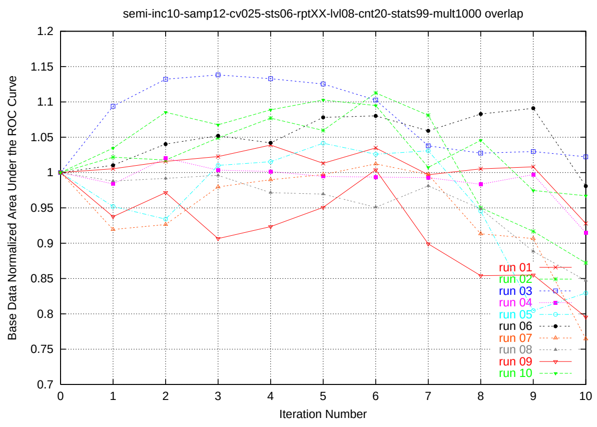
Figure 55 and Table 14 show the performance results for these experiments. Figure 55a presents the full data normalized AUC results and Figure 55b presents the full data normalized false positive count at at detection rate of 90%. Again, the error bars indicate the 95% significance interval. The first thing to notice is that, for both performance measures, the performance is significantly better than the results with the fully labeled data alone over the range of fully labeled data sets explored. It is also interesting to note that normalized AUC performance is relatively flat over initial fully labeled training set size, with a mean of about 0.96 of the full data performance. Also, the standard deviation around the mean is relatively small and does not increase with a decrease in training set size. This also seems to apply to the data set at a sampling rate of 16,

Sample Rate	Data Count	Full Norm AUC Start		Full Norm AUC Best				Run Count
		Mean	95% Interval	Mean	95% Interval	Std Dev	Std Err	
8	60	0.945	0.921-0.969	0.999	0.985-1.014	0.027	0.009	9
10	48	0.907	0.881-0.932	0.969	0.949-0.990	0.040	0.013	10
12	40	0.880	0.835-0.924	0.966	0.941-0.991	0.049	0.015	10
14	34	0.860	0.800-0.919	0.955	0.932-0.979	0.046	0.014	10
16	30	0.807	0.726-0.888	0.960	0.945-0.976	0.030	0.009	10
20	24	0.737	0.654-0.820	0.925	0.889-0.962	0.070	0.022	10

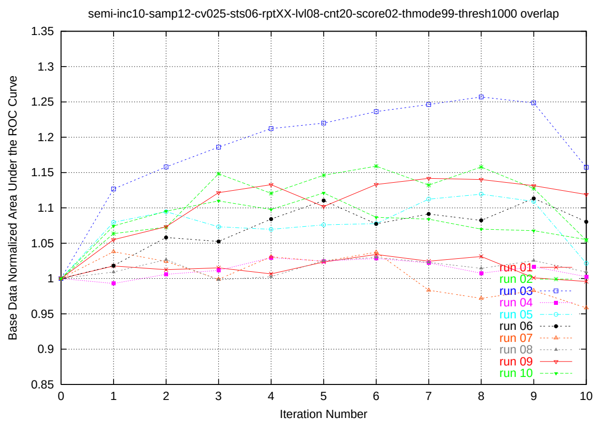
Table 14: This table summarizes the normalized AUC performance of the fifteen feature detector under the "cv025 sts06" condition. Weakly labeled data was incorporated using the MSE selection metric and the fully labeled training set size was varied. The 95% significance interval is computed as 1.64 times the standard error.

which we observed to be toward the beginning of the "failure" regime in Figure 50. This can be contrasted with the performance when the confidence metric is used, as presented in Figure 52 and Table 12, where the performance with weakly labeled data is not significantly better than the initial training set, and final performance is not level over training set size. It should be noted, however that, given the experimental results, the difference between the two selection metrics are not significantly different at the 95% level. It is also informative to present the results comparing these two selection metrics in a slightly different way. Figure 56 plots the base model normalized AUC values over iteration for each of the runs for the confidence metric in (a). The MSE metric is plotted in (b) at a sample rate of 12. The base model normalized AUC performance normalizes the AUC value by the initial performance level from the model, which uses fully labeled data only. This type of plot is useful in that it shows the dynamic behavior as weakly labeled images are added to the training set. Here it can be seen that, when performance increases, it will often peak and then start to decrease at some point. These plots are also useful in comparing the behavior of the two metrics. It is important to note that while the vertical scale is the same, the location of the 1.0 value on the y-axis is different in the two subfigures. Because of the normalization, all of the runs start at 1.0. In the case of the confidence-based runs plotted in (a), performance immediately increases for some runs and decreases for other runs. This is in contrast to the MSE based runs plotted in (b), where all of the runs seem to mostly stay above or at the 1.0 level of performance.

It is also interesting to visualize the actual differences in the detections made by the detector before and after the addition of weakly labeled data. Example detections on the test set are shown in Figure 57. The results shown are from run 2 of the "sts06" experiment with 1/12 of the fully labeled data set size. Results are presented for the initial model, containing fully labeled data only, and for the best AUC performing iteration, iteration 6. Detections are plotted as green boxes scaled according to the scale of the detection and centered at the detection location. The detections plotted are the two highest confidence detections for each image. Rows 1 and 3 contain example detections with the fully labeled data subset only. Rows 2 and 4 contain example detections with the addition of weakly labeled data. Row 1 and row 2 can be compared to examine some of the cases where the use of weakly labeled data has yielded an improvement in detection performance. Row 3 and row 4 can be compared to examine some of the cases where the use of weakly labeled data has not yielded any improvement in detection performance or, in the case of the final image on row 4, a small degradation in performance. Figure 57 presents only a subset of the test set results. But in almost all cases, the detections output by the model after the semi-supervised training process were as good



(a)



(b)

Figure 56: These figures plot the initial iteration normalized AUC performance of the detector .Weakly labeled data is incorporated using the MSE and the confidence selection metrics for a sample rate of 12 under the “cv025 sts06” condition. Subfigure (a) plots AUC performance versus iteration for the confidence-based measure and subfigure (b) plots AUC performance versus iteration for the MSE score based selection metric. Note that in (a), the performance both increases and decreases with each iteration as weakly labeled data is added, whereas in (b), performance generally increases or stays the same.

as or better than the base model.

Given the performance measurements, it seems that the MSE-based selection metric performs better than the confidence-based selection metric. However, it would be useful to better understand why this is the case. One possible explanation, which we have already presented, is that the MSE-based metric is not affected by the small training set size. It is similar to a nearest neighbor approach and does not suffer from the distribution change that the confidence-based measure can have. Another possible explanation is related to the rotation of the weakly labeled training examples and the rotation invariance of the detector itself, as generated through the training set examples and their interaction with synthetic rotation. We describe and examine this issue in more detail in the next section.

We may also understand the differences between the two metrics by looking at the images selected at each iteration by each metric. Figure 58 contains montages of the weakly labeled training images selected at each iteration for a single run (run 9) of semi-supervised training using the confidence metric and the corresponding run using the MSE-based selection metric. In both cases the initial training set, labeled iteration 0, is the same. The performance reported is the full data normalized AUC at the end of that specific iteration, including the new images on that row of the table. In this specific run, the performance of the detector trained with the MSE-based selection metric improves with each iteration, whereas the performance of the confidence-based one decreases. At iteration 1 both methods have chosen new training examples, which appear to be valid; however, there appears to be less variation in the training examples chosen by the confidence-based metric. Yet performance has already decreased at the end of this iteration for the confidence-based metric. From there, things seem to continue to be problematic for the confidence-based training. In iterations 2-4, there are clearly incorrect detections included in the training set and some detections which seem to be wildly off in terms of scale. The MSE-based training seems to behave very differently. All of the images that it selects seem to be “good”. There does seem to be one bad selection made in iteration 4. It is a mouth that looks like an eye in the first row, toward the center of the training images selected for that iteration. It is

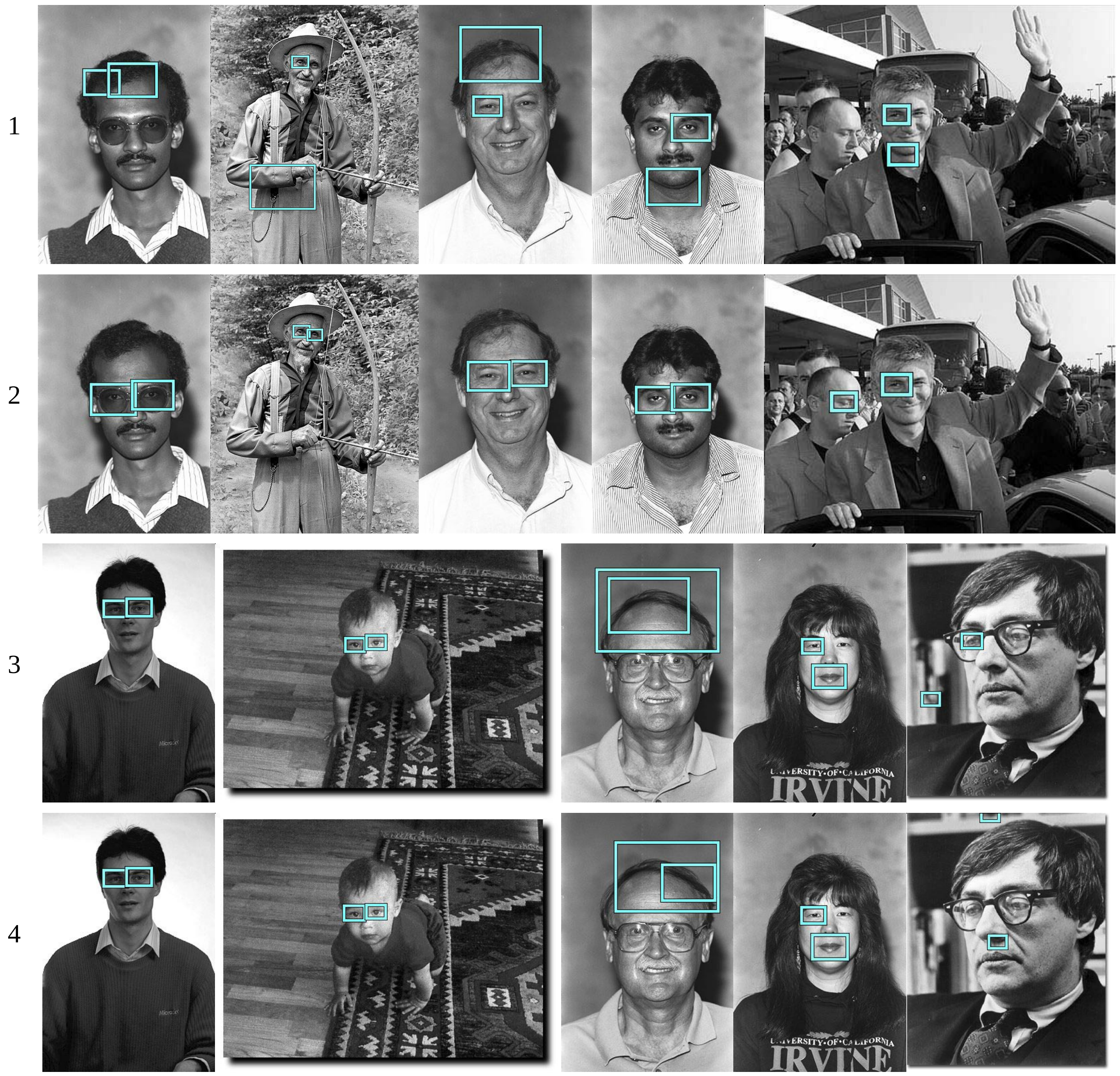


Figure 57: These are examples of detections generated by the detector on the test set before and after the use of weakly labeled data. The two detections with the highest confidence values are plotted with a green box scaled according to the detection size. Rows 1 and 3 contains example detections with the fully labeled data subset only. Rows 2 and 4 contain example detections with the addition of weakly labeled. The examples in row 2 show improvements over those in row 1 with the use of weakly labeled data. The examples in row 4 show no improvement or some degradation with the use of weakly labeled data.

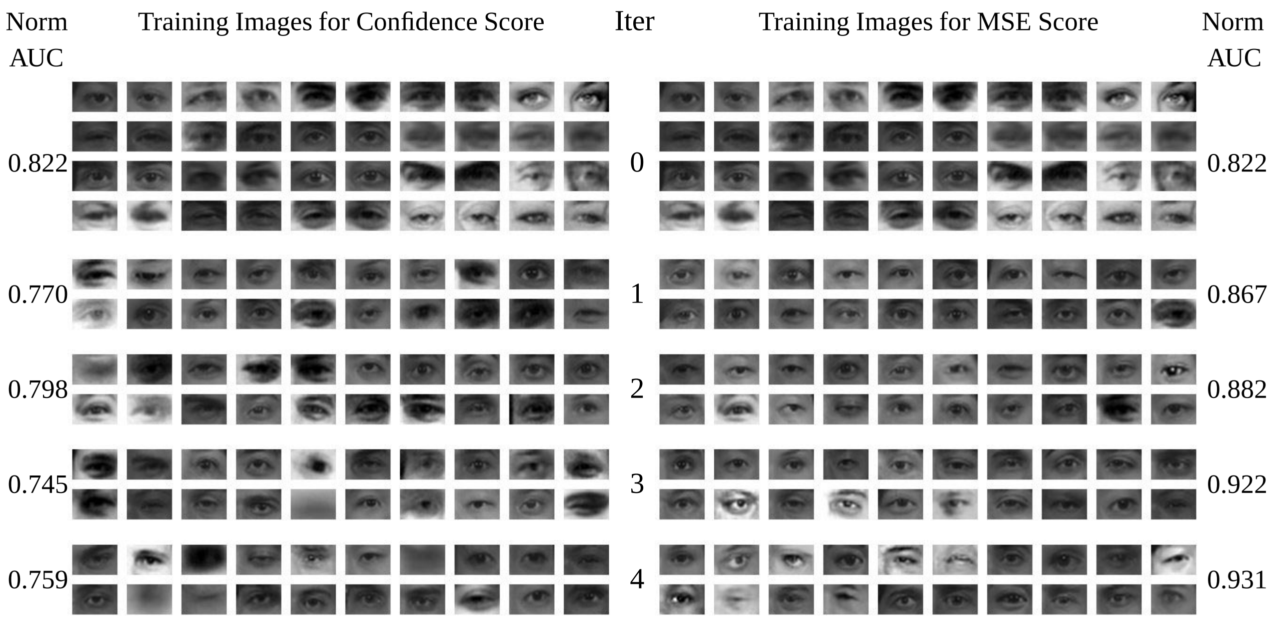
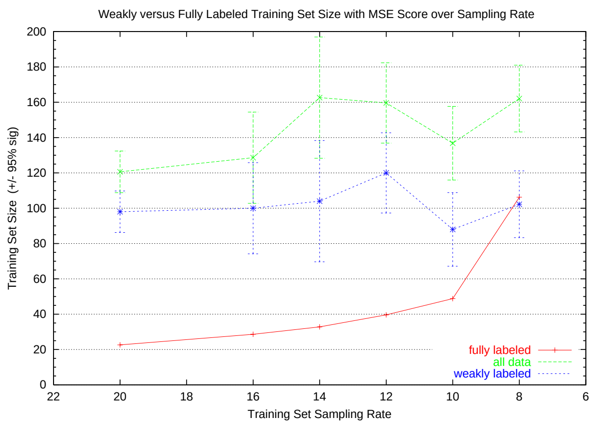


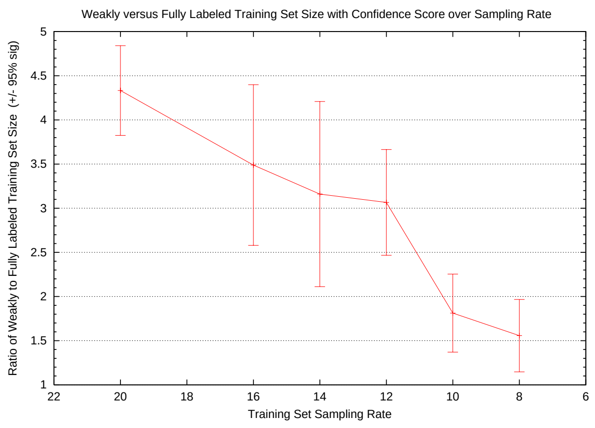
Figure 58: This figure compares the training images selected at each iteration for the confidence-based and the MSE-based selection metrics. The initial training set of 40 images is the same for both metrics and is 1/12 of the initial training set. Performance at each iteration is reported as full data normalized AUC and is increasing with each iteration for the MSE based score and decreasing for the confidence based score.

important to remember that, even when the MSE-based selection metric is used to select new weakly labeled images, the detector uses its confidence-based score to select the best detection in the image. It is those best detections which are displayed in this figure and used as input to the MSE-based algorithm. One hypothesis as to the failure of the confidence-based metric is that it selects high-confidence examples and focuses on those, causing it to learn a skewed distribution. High confidence examples might be eyes that are dark, since the brow ridge often casts a shadow over the eye. When comparing the training images selected by both methods, this does seem to be a major difference. Another hypothesis is that the MSE-based measure is tolerant of variations (such as scale, orientation, and location) and therefore does not cause the detector to expand the concept of an “eye” beyond its inherent capacity. Again, we will examine this in terms of invariance to rotation in the next section.

We also analyzed the MSE score metric in terms of the relative proportions of fully labeled and weakly labeled data in the best performing solutions. This data is presented in Figure 59 and Table 15. In Figure 59a the raw data counts for the fully labeled, weakly labeled, and total training set size for different initial fully labeled data set sizes. Also, error bars are presented, which plot the corresponding 95% significance intervals. The results here seem to be very similar to those found for the confidence metric in that the amount of fully labeled data and the total data seems to be relatively constant over the base training set size. Also, the ratio of weakly labeled to fully labeled data in the best performing solution increases as the fully labeled data set size decreases. The only difference appears to be that the total data set size is now slightly related to the base training set size, in that the larger base training sets will result in larger total final training sets. Again, the total training set size appears to lie in the “saturated” regime as plotted in Figure 50 at around 120 images.



(a)



(b)

Figure 59: This figure plots and compares the amount of fully labeled and weakly labeled data utilized in the best performing solutions using the MSE metric as the fully labeled training set size varies under the "cv025 sts06" condition, where fifteen features are used. Subfigure (a) plots the actual data counts. Subfigure (b) plots the ratio of weakly labeled to fully labeled data. The error bars indicate the 95% significance interval of the mean value.

Sample Rate	Data Count	Weak Count	Weak Ratio	Total Count	Run Count
8	60	76.0	1.291	135.6	10
10	48	62.0	1.267	110.8	10
12	40	92.0	2.346	133.0	10
14	34	72.0	2.217	104.8	10
16	30	60.0	2.087	88.6	10
20	24	90.0	3.977	112.6	10

Table 15: This table summarizes the normalized AUC performance of the fifteen feature detector under the "cv025 sts06" condition. Weakly labeled data was incorporated using the MSE metric and fully labeled training set size was varied. The 95% significance interval is computed as 1.64 times the standard error.

5.8.5 Rotation Estimation and Synthetic Rotation Variation

During the semi-supervised training process, we use the detector as trained up to that point to estimate the values of the latent variables in the weakly labeled examples. The latent variables, which the Schneiderman detector estimates, are scale and position of the object in the weakly labeled image. In preliminary experiments, we found that the entire semi-supervised process was quite sensitive to accurate estimation of these values. However, one value which the detector does not estimate is the orientation of the object in the image. In the case of the fully labeled data points, orientation information is provided. As part of the pre-processing of the training data, all of the training examples are scaled to a canonical size and rotated to a canonical orientation. Rotation invariance is achieved by generating synthetic versions of the training examples once they have been rotated into the canonical orientation. In all of the experiments presented in this section up to this point, the amount of this synthetic variation is ± 12 degrees. However, in the case the semi-supervised experiments described thus far, rotation was not estimated. Thus, when a weakly labeled example was added to the training set, it is added with its inherent rotation and an additional rotation will be added on top of that, given the synthetic variation. What this means is that, given that the detector has some amount of invariance to orientation, it is possible that the semi-supervised training process might attempt to increase the invariance captured by the detector with each training iteration, which labels new weakly labeled data. As this process repeats with each iteration, it is possible that the detector will be forced to try to capture more and more rotation variation. This can be problematic in that most detection systems have some limit to the amount of appearance (and hence orientation) variation which they can capture. It is possible that performance might drop with the addition of weakly labeled data, not because of the choice of the examples which are being added, but because the detector is being required to model too large a variation. This also is a possible explanation for why the MSE score metric is so successful. Because of its simplicity, it has no inherent orientation invariance, and therefore does not increase the modeling demands of the detector. In this section, we describe a series of experiments which attempts to explore this issue.

The issues we will explore through the set of experiments detailed in this section are:

1. Is the reason for the performance of the MSE score metric its rotation invariance?
2. How much rotation invariance can the detector capture, and what is its effect on semi-supervised training performance?
3. Is estimating rotation helpful?
4. What is the interaction between the MSE score metric and rotation estimation?

We first explore the sensitivity of the detector's performance with a limited fully labeled data set, the base detector, to the amount of synthetic angle variation. We compare this to the performance with the full data model for the same amount of synthetic variation. Figure 60 plots the results of a set of experiments, which evaluate four different amounts of synthetic variation, ± 3, ± 6, ± 12, and ± 20, for a data set at 1/12 sampling rate with fully labeled data only. Performance is reported as the full data normalized AUC with 95% significance intervals. It is interesting to note that there seems to be little difference between the performance of these models. It would appear that changing the amount of synthetic variation does not have a major impact on the sensitivity to training set size for this detector.

We now look at the performance of the detector and the interaction between synthetic angle variation, score metric, and rotation estimation. Since the detector does not inherently estimate orientation information, we

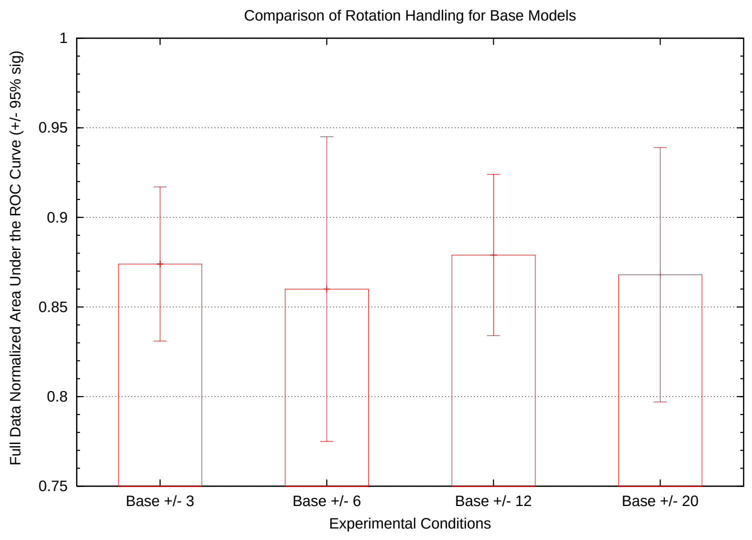


Figure 60: This figure plots the normalized AUC performance of the detector under various conditions under the "cv025" condition with fully labeled data only at a sampling rate of 1/12. The purpose of this plot is to evaluate the effect and interaction between the use of rotation estimation and synthetic rotation variation. The error bars indicate the 95% significance interval of the mean value.

wrote wrapper code which estimates orientation in a similar fashion to the method used by the detector to estimate scale and location. This is done by scanning the detector over scale and position and merging these detections together, taking detector confidence into account. Our approach was to generate rotated versions of the images to be detected. Specifically we used five versions of each image: 0, ±4, ±8, and ±12 degrees. Operationally, the detector was run on each of these images, and the detection with the highest confidence (or highest MSE score when that metric was used) over these images was selected as the "best" detection for that image. This detection process was used as part of the inner loop of semi-supervised training process. To understand the interaction, we performed a set of experiments that looked at how the performance of detectors trained with synthetic variations of ±6, ±12, and ±20 degrees, as well as weakly labeled data, varied with the use of the MSE based score metric and the confidence-based score metric with and without rotation estimation. Figure 61 and Table 16 detail the results of these experiments where performance is reported as the full data normalized AUC and error bars indicate 95% significance intervals. The first thing to note is that, unfortunately, the performance differences between the models is not significant at the 95% level. However, it appears that rotation estimation does not provide a large benefit and therefore, the improvement realized by the MSE score metric is probably not achieved solely by rejecting highly rotated potential training examples.

This issue is explored in further detail in Figure 62. These plots include results for models with synthetic variations of ±6, ±12, and ±20 degrees, as well as weakly labeled data, varied with the use of the confidence-based score metric and the MSE based score metric, both with and without rotation estimation. Again full data normalized AUC performance is plotted with a 95% significance interval. And again, the differences in most cases are not statistically significant. However, some observations can be noted. The first is that the only condition under which the weakly labeled data performance is significantly better than the base model is when the MSE score metric is used. Also, although the differences are not significant, the MSE-based model seems to perform best when compared to the other experimental conditions; rota-

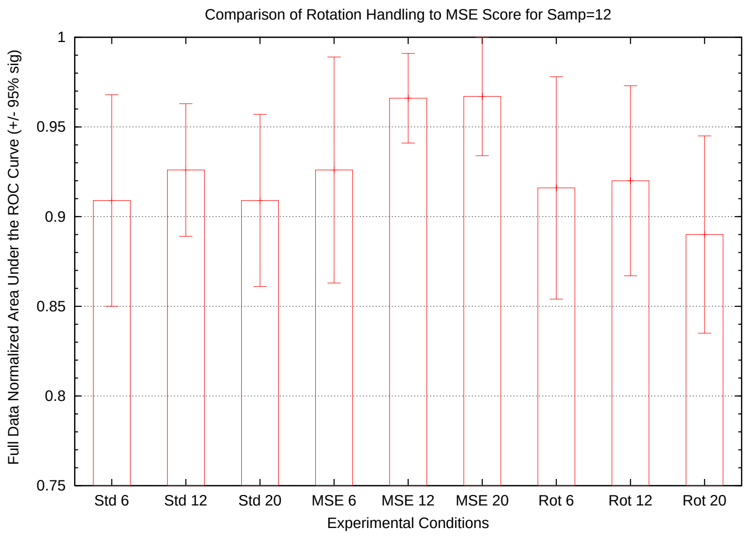
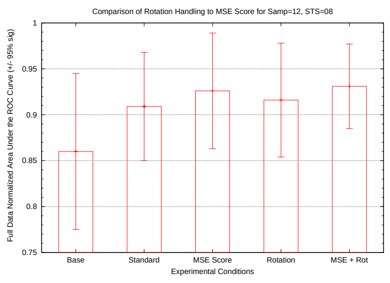


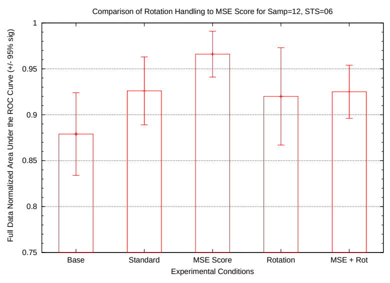
Figure 61: This figure plots the normalized AUC performance of the detector under the “cv025” condition with weakly labeled data. The fully labeled data set sampling rate was 1/12. The purpose of this plot is to evaluate the effect and interaction between the use of rotation estimation and synthetic rotation variation. The error bars indicate the 95% significance interval of the mean value.

Condition	Full Norm AUC Best				Run
	Mean	95% Interval	Std Dev	Std Err	Count
Base +/- 6	0.860	0.775-0.945	0.116	0.052	5
Base +/- 12	0.879	0.835-0.924	0.086	0.027	10
Base +/- 20	0.868	0.797-0.939	0.097	0.043	5
Std +/- 6	0.909	0.849-0.968	0.081	0.036	5
Std +/- 12	0.926	0.890-0.963	0.071	0.022	10
Std +/- 20	0.909	0.861-0.957	0.066	0.029	5
MSE +/- 6	0.926	0.863-0.989	0.086	0.038	5
MSE +/- 12	0.966	0.941-0.991	0.049	0.015	10
MSE +/- 20	0.967	0.935-1.000	0.045	0.020	5
Rot +/- 6	0.916	0.855-0.978	0.084	0.038	5
Rot +/- 12	0.920	0.868-0.973	0.072	0.032	5
Rot +/- 20	0.890	0.836-0.945	0.074	0.033	5
MSE+Rot +/- 6	0.931	0.885-0.977	0.063	0.028	5
MSE+Rot +/- 12	0.925	0.896-0.955	0.040	0.018	5
MSE+Rot +/- 20	0.929	0.887-0.971	0.058	0.026	5

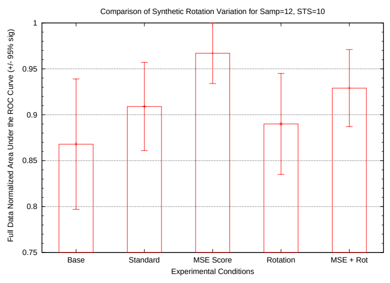
Table 16: This table summarizes the normalized AUC performance of the detector with the incorporation of weakly labeled data under various conditions under the “cv025 sts10” condition and a data set sampling rate of 1/12. The purpose of this table is to evaluate the effect and interaction between the use of rotation estimation and synthetic rotation variation. The 95% significance interval is computed as 1.64 times the standard error.



(a)



(b)



(c)

Figure 62: This figure plots the normalized AUC performance of the detector under the "cv025" condition with weakly labeled data. The fully labeled data set sampling rate was 1/12. The effect of various amounts of synthetic rotation is presented. The purpose of this plot is to evaluate the effect of the use of different score metrics as well as rotation estimation when used in isolation and in combination. The error bars indicate the 95% significance interval of the mean value. Subfigure (a) plots the AUC performance for ±6 degrees of angle variation, (b) ±12 degrees, and (c) ±20 degrees.

Condition	Full Norm AUC Best				Run
	Mean	95% Interval	Std Dev	Std Err	Count
cnt=15 / samp=08	0.971	0.956-0.986	0.028	0.009	10
cnt=20 / samp=08	0.942	0.916-0.967	0.034	0.015	5
cnt=15 / samp=12	0.926	0.890-0.963	0.071	0.022	10
cnt=20 / samp=12	0.928	0.882-0.974	0.063	0.028	5

Table 17: This table compares the normalized AUC performance of the detector with the incorporation of weakly labeled data utilizing different numbers of features and data set sampling rates of 1/8 and 1/12. The 95% significance interval is computed as 1.64 times the standard error.

tion estimation in combination with the confidence-based metric performs the worst. The performance of the MSE score version is also better than when combined with rotation estimation, although the difference again is not statistically significant. One might hypothesize that rotation estimation does not help because there is either not significant orientation variation in this data set or, similar to the observation in [Rowley 98b], rotation estimation can increase the false positive rate. This can happen when it causes every possible detection window to be rotated to look more like the object of interest. Another interesting trend observed in Figure 62 is that the performance of the method that uses rotation estimation seems to decrease as the amount of synthetic variation increases. Again, a possible explanation for this is that rotation estimation actually increases the likelihood of a false positive by making detection windows that were rejected because their rotation was too great or atypical by bringing them into the range of invariance of the detector, which becomes larger as synthetic variation is increased.

5.8.6 Varying Feature Count

In a previous section, we explored the effect of detector feature count on the performance of the detector when trained with fully labeled data lone. However, we also explored this issue in the context of weakly labeled data. To that end we performed a small set of experiments to explore whether using more than

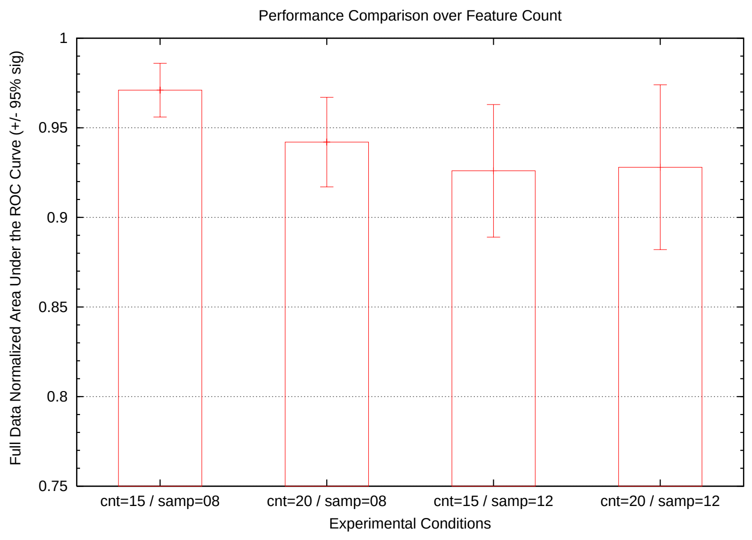


Figure 63: This figure plots and compares the normalized AUC performance of the detector utilizing different numbers of features with weakly labeled data. Fully labeled data set sampling rates were 1/8 and 1/12. The error bars indicate the 95% significance interval of the mean value.

Condition	Full Norm AUC Best				Run
	Mean	95% Interval	Std Dev	Std Err	Count
Base	0.940	0.901-0.980	0.054	0.024	5
Standard	0.984	0.972-0.996	0.016	0.007	5
MSE Score	0.967	0.951-0.983	0.021	0.010	5

Table 18: This table compares the normalized AUC performance of the detector with the incorporation of weakly labeled data under the "scva025 sts06" condition and a data set sampling rate of 1/4. The 95% significance interval is computed as 1.64 times the standard error.

15 features might improve performance. The results of these experiments are detailed in Figure 63 and Table 17. In these experiments, we examined using 15 and 20 features at sample rates of 8 and 12 with the confidence-based metric. The performance was measured in terms of full data normalized AUC. The results seem to be very similar; it appears that at a sample rate of 8, 15 features might perform better than 20 features. If this difference is genuine, then a possible explanation would be that as the number of features increases, the likelihood of overfitting increases.

5.8.7 Feature and Classifier Training

An alternate means of semi-supervised training of the Schneiderman detector is to perform both feature selection and re-train the Adaboost classifier during semi-supervised training. In all of the semi-supervised training experiments performed thus far, only the Adaboost classifier was re-trained. In this set of experiments, we performed a limited evaluation of a system which re-trained both the features and the classifier. As described in a previous section, in this approach, the training set is partitioned during the feature selection stage. This partitioning was performed because it was necessary to use a separate cross validation set during

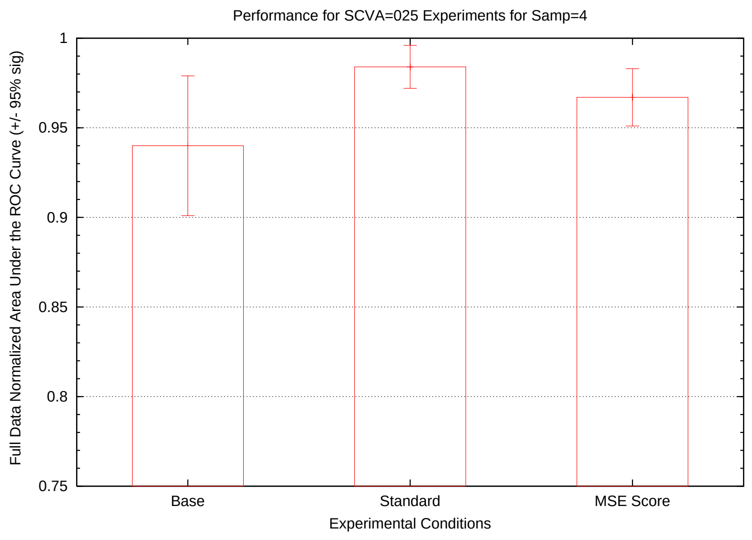


Figure 64: This figure plots and compares the normalized AUC performance of the detector under the "scva025 sts06" condition with the incorporation of weakly labeled data. The fully labeled data set sampling rate was 1/4. The error bars indicate the 95% significance interval of the mean value.

Condition	Full Norm AUC Best				Run
	Mean	95% Interval	Std Dev	Std Err	Count
adab=cv / samp=12	0.926	0.890-0.963	0.071	0.022	10
adab=08 / samp=12	0.889	0.846-0.933	0.059	0.027	5
adab=cv / samp=14	0.914	0.880-0.947	0.064	0.020	10
adab=08 / samp=14	0.846	0.794-0.898	0.071	0.032	5

Table 19: This table compares the normalized AUC performance of the detector with the incorporation of weakly labeled data utilizing either cross validation or a fixed number of iterations for Adaboost and data set sampling rates of 1/12 and 1/14. The 95% significance interval is computed as 1.64 times the standard error.

the feature selection step. (All of the training data is used for the Adaboost step.) The experiments compared the base model at 1/4 of the data set size to the confidence-based metric and and the MSE-based metric. The results of these experiments are presented in Figure 64 and Table 18. The results are not statistically significantly different for the different experimental conditions, but it does seem like the confidence-based metric might have a small advantage. However, many more experiments would need to be performed to resolve this issue.

5.8.8 Adaboost Cross Validation

For the experiments in this section of the thesis, we use the same data set both for setting the weights of and cross validating Adaboost during the training of the detector. The experiment conducted in this section evaluates that choice. The experiments compare selecting the number of boosting rounds from 1-8 using cross validation to always select the last, eighth boosting round. This comparison was conducted for sam-

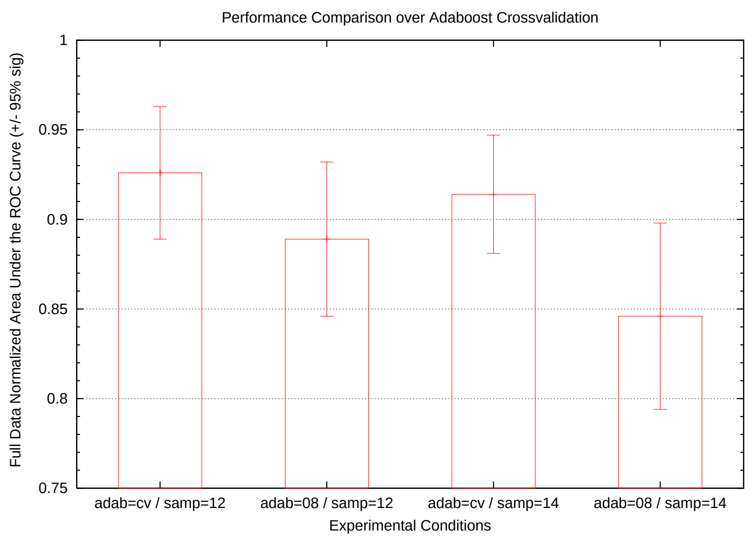


Figure 65: This figure plots and compares the normalized AUC performance of the detector with the incorporation of weakly labeled data utilizing either cross validation or a fixed number of iterations for Adaboost and data set sampling rates of 1/12 and 1/14. The error bars indicate the 95% significance interval of the mean value.

pling rates of 12 and 14 under the "cv025" set of experimental conditions. The results of these experiments are presented in Figure 65 and Table 19. Performance is plotted as full data normalized AUC and the 95% significance intervals are displayed. Given the number of experiments performed, the differences between the performance under different conditions is not statistically significant. However, we do believe that the data shows that the selection of our cross validation strategy was justified because it results in performance which is at least as good as or better than selecting a fixed Adaboost iteration.

5.9 Conclusions

The goal of the work described in this section was to further explore the issues which we are the focus of this thesis:

- detector performance in relation to fully labeled training set size
- performance of the self-training approach to semi-supervised training
- impact of the selection metric for self-training

To that end, we chose to work with a state of the art object detection system, the Schneiderman detector [Schneiderman 03, 04a, 04b, 04c]. We selected a human eye on a frontal face as the object to detect. Our semi-supervised training approach was implemented as a wrapper around the basic detector. One of the challenges of these experiments is that the Schneiderman detector, like other state of the art detectors [Viola 01], takes quite a long time to train. Since detector training was part of the inner loop of our system, a single experiment would often take 12 hours $\times$ 10 iterations = 120 hours of CPU time on a 3.0 GHz machine. The

5 runs of a typical experiment totaled 600 CPU hours or 25 CPU days. All of the experiments presented in this section consumed about a total of three CPU years. This makes explicit the trade-off in the semi-supervised approach between CPU time and human labeling time. Obviously, as CPU's become faster with time, this will be less of a consideration in the future.

Sensitivity to Fully Labeled Training Set Size

We first explored the sensitivity of the detection system itself to the amount of labeled training data. We found three regimes of operation, as presented in Figure 50a: the "saturated" regime where an increase in the number of fully labeled training examples would not help, the "smooth" regime where there is a smooth decline in performance as the number of training examples decreases, and the "failure" regime where performance quickly declines. We chose fully labeled training set sizes in the "smooth" regime and applied self-training to the training of this detector.

Final Performance

The final performance with weakly labeled data was very good and had relatively small variance over a range of initial fully labeled training set sizes. Sample results can be visually inspected in Figure 57. In our experiments, we compared a confidence-based selection metric to a MSE-based selection metric in terms of final performance. We found that the best performance was achieved using the MSE-based method as reported in Figure 55 and Table 14.

Selection Metric Importance

The most important factor which we believe was crucial to the performance obtained, was the use of the MSE score as the data selection metric. We evaluated both a confidence-based metric and the MSE metric. The performance of these two metrics are plotted in Figures 52 and 55. A comparison of the reduction in the variance of the final performance to the in initial training set is presented in Figure 56. We believe that the improved performance is mostly because of the nearest neighbor effect as described previously in this thesis; however, we cannot rule out other effects, like improved rejection of false positives and highly rotated training images. We explored these issues in our experiments detailed in this section, but the differences observed were not significant.

Other Factors Affecting Performance

There was also a conjunction other factors, system parameters, that made the overall approach work. One important factor was an accurate estimate of the location and scale of the object as reported by the detector during the semi-supervised training process. In a set of preliminary experiments that were performed, but not reported here, we estimated the location of the weakly labeled object to an accuracy of 4 pixels at the native object scale (24×16) and the scale was estimated in steps of $2^{0.25} \approx 1.19$. We found that this was insufficient to achieve good performance in our semi-supervised approach. When we moved to a positional accuracy of 2 pixels and a scale step $2^{0.125} \approx 1.09$, performance greatly improved. The disadvantage of moving to finer granularity in position and scale is an increase in processing time. An improvement by a factor of two on positional accuracy in the x and y dimensions and scale increases detection time by a factor of eight. This may not seem like a big issue, since detectors are designed to operate quickly. However, if a large weakly labeled data set needs to be scanned, this increase may be significant. Of course, this is only an issue during training time and is another example where the weakly labeled data approach greatly increases training time in exchange for a smaller number of labels which need to be generated.

A second issue faced in the experiments was to configure the detector so that a large enough pool of features was selected, and a good classifier could result. Because of the small data set size, feature selection might not always correctly rank and in turn select the best features. By increasing the feature pool, some of these mis-rankings will have a small effect. However, the feature pool must not be made too large or overfitting may occur.

Future Work

There are still many tasks left to future work. Among them are further experiments that could be used to reduce the size of the significance intervals and better resolve performance differences between experiments conducted under different conditions. Another task for future consideration is the development of a technique based on cross validation that could select the best semi-supervised iteration for the final model.

6 Discussion and Conclusions

6.1 Summary and Conclusions

The goal of this work was to explore and evaluate various approaches to semi-supervised training using weakly labeled data in the context of appearance-based object detection. To that end we devised and implemented a number of semi-supervised training approaches and evaluated their performance on various detectors using real world data sets. In our experiments, weakly labeled images were images that were known to contain the object of interest, but certain specifics about the object were not known. This typically included things like location, scale, and orientation. Without this information, a weakly labeled image cannot be used in the training process. The semi-supervised training process attempts to estimate these hidden values for all, or a subset of the weakly labeled data. It is also important to note that these latent variables make the semi-supervised training of image detectors somewhat different than the standard semi-supervised or unlabeled data case. The difference is that each weakly labeled image can generate one or more training examples. And if an iterative process is used, these training examples change with each iteration. So instead of a fixed set of weakly labeled examples, as is the typical case in image detection, the set of weakly labeled examples changes with each iteration.

The two semi-supervised approaches we focused on were expectation maximization and self-training. Expectation maximization (EM) is an iterative algorithm, which can be used to jointly estimate latent variables and model parameters. In the context of unlabeled data, it has been used to jointly estimate class labels and model parameters. In the context of weakly labeled data, it has been used to jointly estimate the values of the latent variables and the model parameters. The iterative process alternates between one phase that estimates the expected values of the latent variables and a second phase that maximizes the likelihood of the model parameters. This process is typically repeated until convergence.

Self-training is also an iterative process. It is similar to EM in that it alternates between one phase that estimates the values of the latent variables and another phase that estimates the model parameters. There are two main differences. The first is that not all of the weakly labeled data is typically used immediately in the training process. Usually, small amounts of data, or even a single example, are incorporated into the training process. This has two potential benefits; first, that bad training examples with incorrectly estimated latent variable values can be excluded from the training process. Secondly, the amount of weakly labeled data can be controlled so it does not overrule the fully labeled data. This effect is somewhat controlled in EM by the weights assigned during the training process. However, a large amount of unlabeled data can swamp out the labeled data even if the label values are uncertain. (A similar effect can be implemented by annealing the weight of the weakly labeled data in EM.) A second benefit of self-training is that we typically freeze the values of the latent variables once a weakly labeled example has been incorporated into the training set. (These values need not be frozen, but in general we found performance was improved when we did so.) Freezing these values has both its disadvantages and advantages. If an error is made in estimating a latent variable value, it cannot be corrected. On the other hand, freezing the variables prevents model drift. If the values are allowed to change, then newly labeled examples with potential errors may change the model in a direction that increases the likelihood of those errors, and the entire model may become worse with each training iteration.

We performed three sets of experiments with three different detectors using a real world data set to examine these semi-supervised training approaches. The first set of experiments, described in Section 3, used a simple color-based detector with a generative model. In the second set of experiments, described in Section 4, the detector used a filter bank to generate a pixel-based feature vector. The model utilized a generative mixture of Gaussians to model both the object and the clutter. The third set of experiments, described in Section 5, utilized the Schneiderman detector, where features are automatically constructed from groups of wavelet coefficients. These are fed to a discriminative classifier, which is trained using a variant of Adaboost. In all three cases, we found that training these detectors in a semi-supervised manner using weakly labeled data improved performance over the use of a small set of fully labeled data alone.

In the experiments with the filter-based detector we found that the self-training approach outperformed the standard EM approach. Given these results we performed experiments to further our understanding of the self-training approach. Our experimental results lead us to believe that there are two keys to gaining good performance when using the self-training approach. The first is accurate estimation of the latent variables. What we mean here is accurate position, scale, or orientation estimation. If these quantities are not accurately estimated and some error is then introduced, as the weakly labeled data is added to the training set, the amount of appearance variation of the object that the detector is asked to model will increase and may exceed its capacity. The trade-off is between final performance, which benefits from more accurate estimation, and training time, which can be decreased with a coarser search over possible latent variable values. The second key to good performance is the selection metric, which is used to decide which weakly labeled images to add to the training set. Our experiments showed that a local distance metric, such as a mean-squared-error metric, performed better than a confidence-based metric. We present a hypothesis for why this might be the case. First, the distance metric more closely emulates the nearest neighbor algorithm that we introduced and analyzed. A confidence-based metric, on the other hand, may add highly confident detections which are not close to existing training examples. Because these detections are high-confidence detections, they are very likely to be correct, so the introduction of bad training data is not necessarily the problem. What we suggest is that these detections skew the distribution of the training data to be different from its true distribution. The model may change to accommodate the skew, and in the process, move further away from the true distribution. Once this occurs, the inherent feedback loop may cause the model to drift further and further away from the desired model.

As a result of the work presented here we make the following conclusions related to semi-supervised training and object detection:

- **Semi-supervised training can be successfully applied to the object detection task.** In our experiments, significant performance improvements were observed.
- **Self-training can outperform EM for semi-supervised training of object detection systems.** In our experiments, we found that self-training, when paired with the correct selection metric, outperformed EM.
- **The selection metric is crucial to self-training performance.** In our experiments, variants of nearest neighbor worked much better than a confidence-based metric.

In conclusion, we feel that we have demonstrated that self-training is a viable means for incorporating weakly labeled data in the semi-supervised training process. Obviously, there are many additional questions to be answered and algorithms refinements that will be needed in the future.

6.2 Open Questions / Future Work

There are a number of questions which result from this work and were outside the scope of this work that would be interesting to evaluate in the future:

6.2.1 Methods for Selecting the Final Self-training Iteration

One key issue which this work did not address is when to stop adding weakly labeled data to the training set in the self-training process. As the results in Figure 56 demonstrate, performance peaks at a "best" iteration. The results reported in this thesis are the results for the best iteration. This "early termination" of training leaves the detector itself to determine the boundary between the two classes instead of the semi-supervised training technique. So, instead of trying to label points near the decision boundary, we stop the labeling process early and let the detector do it. Obviously it is important to be able to select the iteration in order to end training automatically. The most convenient way to do this would be to use some function of the selection metric or the detection confidence of the weakly labeled data. Although in this work we have found a metric that is a useful means of ordering which examples to select first, preliminary experiments, which are not reported on here, lead us to believe that these metrics are not a useful means of determining when training should be terminated. We believe that the detector will need to use an approach based on cross validation. Obviously, the problem with such an approach is that the amount of fully labeled data is limited and we would like to utilize all of our data for training. There are two issues which may make this less of an issue than it may at first seem. The first is that we believe that a relatively small cross validation set can be used for this purpose because often large performance drops are observed when "bad" data is added to the training set and that may be easily detectable with a relatively small cross validation set. The second is that we believe some form of n-fold cross validation can be used. The key here is that the semi-supervised training process generates labels for the weakly labeled data. This means that we can divide the fully labeled data into different training and cross validation data splits and train the model in a semi-supervised fashion in each case. We can then combine the labels of the weakly labeled data generated from these different splits and train a "final" model using the combined labels in a fully supervised fashion. We felt that exploring these issues was felt to be outside the scope of this thesis, and we leave them to future work.

6.2.2 Detector Retraining

One interesting aspect, which we did not explore in this work, is the decoupling between the detector used for semi-supervised training and the final detector used in operation. During semi-supervised training we found that it was important to have a detector that can more accurately estimate the latent variables in the weakly labeled data, like location and scale. This usually requires reducing the invariance of the detector. However, when the detector is used in an application, the accuracy of these estimates is often less important, and invariance is more important. A more invariant detector can be achieved using the labels generated for the semi-supervised data during the training process. It would be interesting to see how well this approach works.

6.2.3 Relation to Co-training

Another possible explanation for the success of the nearest neighbor metric in self-training is that it is performing a type of co-training, as originally described in [Blum 98] and further explored in [Nigam 00a,00b] and [Levin 03]. In co-training, two learners are typically used. At each iteration of the co-training approach, each learner labels the unlabeled data. The most confident labels are exchanged between the learners and used to augment their training sets. Typically each learner uses a different feature set or algorithm. The hope is that each learner will be more confident in different parts of the feature space and can pass its unique knowledge onto the other learner so they can co-operatively improve their performance. Our self-training approach is potentially similar in that the detector uses its confidence measure to find good detections, and the selection metric uses nearest neighbor to select good weakly labeled examples. One could use either measure alone for both steps. In fact, that experiment was performed when the confidence metric was used to select weakly labeled examples. Likewise, the nearest neighbor metric could be used as the basis of a detector and a selection metric. It would be interesting to perform a set of experiments to determine whether the self-training results in this thesis could be related to co-training.

6.2.4 Initial Training Set Selection

In our experiments we found that the initial training set had a large effect on performance. In some of our experiments, we explored the issue of how best to select an initial training set. However, these results were somewhat limited. It would be interesting to explore this issue further. One possible avenue would be to perform experiments that result in a set of guidelines, which a human might use to select a small set of images to label. Another direction would be to devise an active-learning approach, which could be used to generate an initial fully labeled training set, and then use a semi-supervised approach to label more data. In fact, there is no reason why this would need to be two distinct stages. The two processes might also be interleaved.

6.2.5 Training with Different Types of Information

In this work we utilize information provided in the form of fully labeled and weakly labeled data. However, one can imagine data which has been labeled in a variety of ways. For example, some images might be provided with scale information for the object and nothing else. Or we might be provided with pose information. Additional side information might also be provided, such as the rough shape of the object or perhaps a prior distribution over location in the image. It would be very interesting to further explore ways of incorporating these different types of data in the training process, as many real world applications have a variety of information sources available to them.

6.2.6 Utilizing Image Context

In real world situations, objects occur in a context. The advantage of training with images that are not imaged on blank backgrounds is that the context in which the objects occur is present. This context may take the form of global image statistics which characterize an environment type, like an indoor office scene or an outdoor ocean scene. For example, a horse is more likely to be present outside in a green field, whereas a television set is usually found indoors. Recent work in [Torralba 01,03a,03b] introduces a model for incorporating contextual information into the object detection process.

Smaller scale local context may also be important. For example a standard desk style telephone is usually found sitting on a desk. Another example is that a horse is usually standing up, so the blue of the sky is usually found near the horse's back and the green of the grass near the horse's feet.

Although atypical, this information can be made available in a supervised training scenario when the objects occur in images with context, and the portions of the image that belong to the objects and those that do not are labeled.

In a weakly labeled training scenario, this information can also be utilized. One way to do this is to infer two classes for image pixels, the object and non-object class. Certain background pixels might be incorrectly classified as being part of the object if some other object always occurs in the training data with the object of interest. In a completely weakly labeled scenario, the algorithm would not be able to distinguish this second object from the desired object. This may be handled in two ways. First, some fully labeled data might be introduced to help determine what is the “true” object. Second, some bias could be introduced in the object model.

A question here is whether a separate model or mechanism is needed for the conditioning context or whether the standard object model could accommodate it. If the conditioning pixels are not considered to be part of the object, then small scale models could be accommodated by allowing the context of the spatial model of object features to extend outside of the object itself. Large scale context might be modeled by more global statistics of the images. These might be similar to measures that are used in CBIR.

A staged approach might be used for image retrieval. If we were looking for an image of a horse we might first examine the images whose global statistics indicate that an image is likely to contain a horse (an outdoor scene) and then further process those high likelihood images to determine if a horse is present and where it is.

However, we do not want to rely on the context to detect an object. That is, if the object itself is readily detectable, we do not want the presence or absence of any particular type of context (such as other objects in the scene) to result in the failure of that object being detected. We envision that context will be utilized to help in ambiguous cases. For example if we see a flesh colored blob with no oblong shape (a body) nearby, we might think that it is very unlikely to be a face. However, if we see a nearby body-like object, we would more likely think it is a face. However, if we saw an object that clearly had two eyes, a nose, and a mouth, even if no body was attached, we would also clearly believe it to be a face - in effect ignoring the context.

6.2.7 Learning from Data Mined from the Web

The goal of this future work direction is to explore the possibility of using existing images on the web to train an object detection system. There are a large number of images on the web. These images are often closely associated with text on a web page. In certain cases, on certain news sites, the captions for an image are labeled in the HTML source. If weakly labeled data can successfully be employed for training, then these images on the web become a potential source of training data. In one scenario, the words labeled on a web page associated with an image are assumed to be the names of objects which are present in the image. More complicated analysis of the associated text could include semantic analysis to try to identify which objects are likely to be present in the image. Recent work by [Barnard 01,03] has explored this area, modeling the mapping between image features and words as a statistical machine translation process.

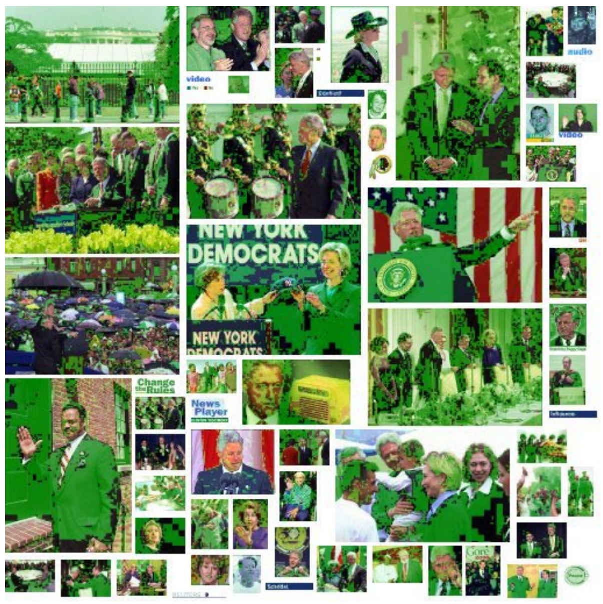
We implemented a web spider and collected images and related web pages from the web sites for CNN, ABC News, and Yahoo News. We ran the spider for 60 days and collected over 100,000 images, totaling about 3 GB of data. As a preliminary experiment, we used color histogram-based detection to see if anything useful could be learned from this data. We ran experiments to see if particular images colors were associated with specific key words. Two of the words we tried were “clinton” and “flag”. Only an empirical examination of the results was performed by visually examining the output. Figures 66a,b,c show the results of this experiment. It includes plots of colors which were more likely to be in the “clinton” or the “non-clinton” class; it also shows the performance on a number of images in a test set which contained the word “clinton” in their associated text. Figures 66c,d,e are the corresponding plots for the flag experiment. We believe the preliminary results to be quite promising.

6.2.8 Fully labeled data can hurt performance

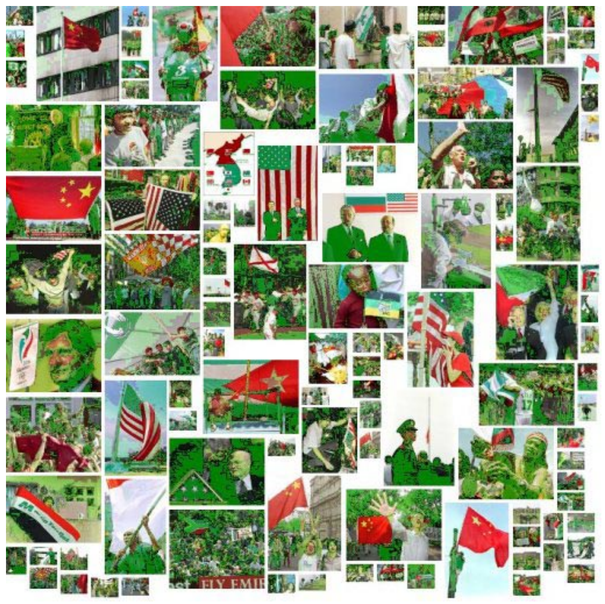
One finding which is applicable to machine learning in general, and needs to explored in future work, is that it is possible to add error-free, fully-labeled data and decrease accuracy. In our experiments we found situations where adding correctly labeled data during self-training resulted in a performance reduction. For example at iteration 1 in Figure 58 in Section 5 we see that all of the training examples selected by the confidence metric and by the MSE metric are valid training examples, yet performance decreases for the confidence metric and increases for the MSE based metric.

At first glance, it may seem counter-intuitive that more data can result in lower performance, but this happens because the learner is training on a data set which is not drawn from the underlying generative distribution. (We first presented this notion with synthetic data simulations in Section 2.5.) Imagine the scenario where a classifier is trained with a limited amount of fully labeled data. An “adversary” then selects a new set of fully labeled data points to augment the existing data set in such a way that our learner is “fooled” into selecting a worse hypothesis that it had with less data. This can happen even if the examples have the correct class labels. This may not be seem like a common situation, but we believe that we encountered this during our self-training experiments and it most probably would occur in active learning scenarios as well. Our hypothesis is that this effect offers one possible credible explanation of why the nearest neighbor selection metric for self-training works better, because it chooses a good distribution of points to label. So in self-training, it may not be sufficient to select examples which have been assigned the correct class labels, but it may be important to select points that are not harmful to the performance of the learner. One could imagine an approach that uses information about the underlying distribution, which is gleaned from unlabeled data to re-weight the training examples to ameliorate this effect. However, when very small training sets are used that do not cover the feature space well, this may not be feasible.

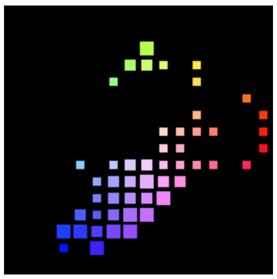
In conclusion, this seems like a novel phenomena which we have not seen previously discussed in the literature and merits further study.



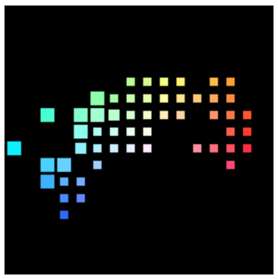
(a)



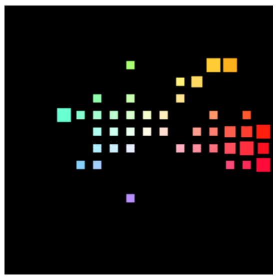
(d)



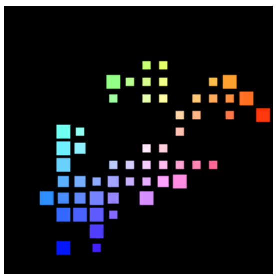
(b)



(c)



(e)



(f)

Figure 66: The color space subfigures plot the HSV color space in polar coordinates, where saturation increases with distance from the center of the plot and hue is plotted as angle. The color space is discretized to visualize the distribution value for that region of the color space, where the area of each square is in proportion to the magnitude of the value. This figure illustrates examples of color-based detection of images with captions containing a specific keyword. For the keyword “clinton”, subfigure (b) shows a plot of colors more likely to be in the “clinton” class, subfigure (c) is a plot of colors more likely to be in the “not clinton” class, and subfigure (a) is the results of classifying image blocks according to these color distributions. Blocks not considered to be members of the clinton class were colored green. For the keyword “flag”, subfigure (e) is a plot of colors more likely to be in the class, subfigure (f) is a plot of colors more likely not to be in the class, and subfigure (d) shows the results of classifying image blocks according to these color distributions, where members of the class colored green. (Note that this is a color figure.)

7 References

7.1 Machine Learning and Statistics

7.1.1 General Machine Learning

[Bishop 95] C. M. Bishop. *Neural Networks for pattern recognition.* New York: Oxford University Press, 1995, pp. 58-59.

[DeGroot 89] M. H. DeGroot. *Probability and statisti*cs *(second edition)*. Menlo Park: Addison-Wesley Publishing Company, 1989, pp. 158-169.

[Duda 73] R. O. Duda and P. E. Hart. *Pattern classification and scene analysis*. New York: John Wiley & Sons, 1973.

[Ling 03] C.X. Ling, J. Huang, and H. Zhang. AUC: a Statistically Consistent and more Discriminating Measure than Accuracy. Proceedings of IJCAI 2003.

[Mitchell 97] T. M. Mitchell. *Machine Learning*. New York: McGraw Hill, 1997.

[Pomerleau 91] D. Pomerleau. Efficient Training of Artificial Neural Networks for Autonomous Navigation. Neural Computation, Vol. 3, No. 1, 1991, pp. 88-97.

[Raghavan 89] Vijay Raghavan, Peter Bollmann, and Gwang S. Jung. A critical investigation of recall and precision as measures of retrieval system performance. ACM Trans. Inf. Syst. 7, 3 (July 1989), Pages 205-229.

[Yan 03] Lian Yan, Robert Dodier, Michael Mozer, and Richard Wolniewicz. Optimizing Classifier Performance via an Approximation to the Wilcoxon-Mann-Whitney Statistic. Proceedings of ICML 2003.

7.1.2 Learning with Multiple Instance Data

[Blum 98] Avrim Blum and Adam Kalai. A Note on Learning from Multiple-Instance Examples. Machine Learning, 30:23–29, 1998.

[Dietterich 97] Thomas G. Dietterich, Richard H. Lathrop, and Tomas Lozano-Perez. Solving the multiple-instance problem with axis-parallel rectangles. Artificial Intelligence , 89(1-2), pp. 31-71, 1997.

[Maron 98a] Oded Maron and Tomás Lozano-Pérez. A Framework for Multiple-Instance Learning. Neural Information Processing Systems 10, 1998.

[Maron 98b] Oded Maron and Aparna Lakshmi Ratan. Multiple-Instance Learning for Natural Scene Classification, ICML 1998.

[Maron 98c] Oded Maron. Learning from Ambiguity. Ph.D. Thesis, MIT, May 1998.

[Zhang 01] Qi Zhang and Sally A. Goldman. EM-DD: An Improved Multiple-Instance Learning Technique. NIPS 2001.

[Zhang 02] Qi Zhang, Sally A. Goldman, Wei Yu, Jason E. Fritts. "Content-Based Image Retrieval Using Multiple-Instance Learning." Proceedings of the Nineteenth International Conference on Machine Learning (ICML 2002).

7.1.3 General Learning with Unlabeled Data / Semi-supervised Training / Active Learning and Non-Vision Applications

[Andrews 02] Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. Support Vector Machines for Multiple-Instance Learning. NIPS 2002.

[Basu 02] Sugato Basu, Arindam Banerjee, and Raymond J. Mooney. Semi-supervised Clustering by Seeding. Proceedings of the Nineteenth International Conference on Machine Learning (ICML 2002), pp. 19-26.

[Blum 98] Avrim Blum and Tom Mitchell. Combining Labeled and Unlabeled Data with Co-Training. Proceedings of the 11th Annual Conference on Computational Learning Theory, pages 92–100, 1998.

[Blum 01] Avrim Blum and Shuchi Chawla. Learning from Labeled and Unlabeled Data using Graph Mincuts. ICML 2001, pp. 19-26.

[Corduneanu 01] A. Corduneanu and T. Jaakkola. Stable mixing of complete and incomplete information. MIT AI Memo AIM-2001-030, 2001.

[Corduneanu 03] A. Corduneanu and T. Jaakkola. On information regularization. In Proceedings of the Ninth Annual Conference on Uncertainty in Artificial Intelligence, 2003.

[Castelli 94] V. Castelli. The relative value of labeled and unlabeled samples in pattern recognition. PhD Thesis, Stanford, 1994.

[Castelli 95] Castelli and Cover. On the exponential value of labeled samples. Pattern Recognition Letters, 16, 105-111. 1995.

[Cover 67] T. Cover and P. Hart. Nearest Neighbor Pattern Classification. IEEE Trans. on Information Theory, IT-13(1):21–27, January 1967.

[Cover 68] T. Cover. Estimation by the Nearest Neighbor Rule. IEEE Trans. on Information Theory, IT-14(1):50–55, January 1968.

[Cozman 03a] Fabio G. Cozman, Ira Cohen, and Marcelo C. Cirelo. Semi-Supervised Learning of Mixture Models and Bayesian Networks. International conference of Machine Learning (ICML) 2003.

[Cozman 03b] Fabio Cozman, Ira Cohen, and Marcelo Cirelo. Semi-supervised Learning and Model Search. ICML 2003 Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining, pp. 111-112.

[Dempster 77] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum Likelihood from Incomplete Data via the EM algorithm". Journal of the Royal statistical Society, Series B, 39(1): 1-38, 1977.

[Ghahramani 94a] Z. Ghahramani and M. I. Jordan. Learning from incomplete data. MIT Center for Biological and Computational Learning Technical Report 108, 16 pages, 1994.

[Ghahramani 94b] Z. Ghahramani and M. I. Jordan. Supervised learning from incomplete data via an EM approach In Cowan, J.D., Tesauro, G., and Alspector, J. (eds.). NIPS 6, 8 pages, 1994.

[Ghani 02] Rayid Ghani and Rosie Jones. A Comparison of Efficacy and Assumptions of Bootstrapping Algorithms for Training Information Extraction Systems. Workshop on Linguistic Knowledge Acquisition and Representation at the Third International Conference on Language Resources and Evaluation (LREC 2002).

[Joachims 99] Thorsten Joachims. Transductive Inference for Text Classification using Support Vector Machines, in Proceedings of the 16th International Conf. on Machine Learning, pp. 200-209, 1999.

[Joachims 03] Thorsten Joachims. Transductive Learning via Spectral Graph Partitioning. ICML 2003.

[Jones 99] Rosie Jones, Andrew McCallum, Kamal Nigam, and Ellen Riloff. Bootstrapping for Text Learning Tasks. In IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications, pp. 52-63. 1999.

[Li 01] C. H. Li. Constrained minimum cuts for integrated classification of labeled and unlabeled data. CVPR 2001.

[Lin 03] Winston Lin, Roman Yangarber, and Ralph Grishman. Bootstrapped Leaning of Semantics Classes using Positive and Negative Examples. ICML 2003 Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining, pp. 103-110.

[Liu 02] Bing Liu, Wee Sun Lee, Philip S. Yu, and Xiaoli Li. "Partially Supervised Classification of Text Documents." Proceedings of the Nineteenth International Conference on Machine Learning (ICML-2002).

[McCallum 99] Andrew McCallum and Kamal Nigam. Text Classification by Bootstrapping with Keywords, EM and Shrinkage. In ACL '99 Workshop for Unsupervised Learning in Natural Language Processing, pp. 52-58, 1999.

[McCallum 00] Andrew McCallum, Kamal Nigam, and Lyle Ungar. Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching. In Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (KDD 2000.)

[Miller 97] D. Miller and H.S. Uyar. "A mixture of experts classifier with learning based on both labeled and unlabeled data." NIPS 1997.

[Moreno 03] Pedro Moreno and Shivani Agarwal. An Experimental Study of EM-based Algorithms for Semi-Supervised Learning in Audio Classifcation. ICML 2003 Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining, pp. 19-25.

[Nigam 98] Kamal Nigam, Andrew McCallum, Sebastian Thrun, and Tom Mitchell. Learning to Classify Text from Labeled and Unlabeled Documents. In Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98), pp. 792-799. 1998.

[Nigam 99] K. Nigam, Andrew McCallum, Sebastian Thrun and Tom Mitchell. Text Classification from Labeled and Unlabeled Documents using EM," Machine Learning, Kluwer Academic Press, 1999.

[Nigam 00a] Kamal Nigam and Rayid Ghani. Analyzing the Effectiveness and Applicability of Co-Training. Proceedings of the Ninth International Conference on Information and Knowledge Management (CIKM 2000).

[Nigam 00b] Kamal Nigam and Rayid Ghani. Understanding the Behavior of Co-Training. Proceedings of the Workshop on Text Mining at the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2000).

[Nigam 01] Kamal Nigam. Using Unlabeled Data to Improve Text Classification. Doctoral Dissertation, Computer Science Department, Carnegie Mellon University. Technical Report CMU-CS-01-126. 2001.

[Pavlopoulou 03] Christona Pavlopoulou, Avi Kak, and Carla Brodley. Applications of Semi-supervised and Active Learning to Interactive Contour Delineation. ICML 2003 Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining, pp. 26-33.

[Rachlin 02] Y. Rachlin. A General Algorithmic Framework for Discovering Discriminative and Generative Structure in Data. M.S. Thesis, Department of Electrical & Computer Engineering, Carnegie Mellon University, 2002.

[Rastaby 95] J. Rastaby and S. S. Venkatesh. Learning from a mixture of labeled and unlabled examples with parametric side information. In COLT, pages 412-417, 1995.

[Schwaighofer 02] Anton Schwaighofer and Volker Tresp. Transductive and Inductive Methods for Approximate Gaussian Process Regression. NIPS 2002.

[Seeger 01] M. Seeger. Learning with labeled and unlabeled data. Technical Report, Institute for Adaptive and Neural Computation, University of Edinburgh, 2001.

[Stauffer 03] Chris Stauffer. Minimally-supervised Classification using Multiple Observation Sets. ICCV 2003. Pages 297-304.

[Szummer 01] Martin Szummer and Tommi Jaakkola Partially labeled classification with Markov random walks. Neural Information Processing Systems (NIPS) 2001, vol 14. 8 pages.

[Szummer 02] Martin Szummer and Tommi Jaakkola. Information Regularization with Partially Labeled Data. NIPS 2002.

[Welling 02] Max Welling, Richard Zemel, and Geoffrey Hinton. Self Supervised Boosting. NIPS 2002.

[Yan 03] Rong Yan, Jie Yang, and Alex G. Hauptmann. Automatically Labeling Data Using Multi-class Active Learning. ICCV 2003. Pages 516-523.

[Zhang 00] T. Zhang and F. Oles. A probability analysis on the value of unlabeled data for classification problems. ICML 2000.

[Zhu 03a] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. ICML 2003.

[Zhu 03b] Jerry Zhu, John Lafferty, and Zoubin Ghaharmani. Combining Active Learning and Semi-Supervised Learning using Gaussian Fields and Harmonic Functions. ICML 2003 Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining, pp. 58-65.

7.2 Computer Vision

7.2.1 General Object Detection / Recognition / Content Based Image Retrieval

[Burl 95] M.C. Burl, T.K. Leung, and P. Perona. Face Localization via Shape Statistics. Int. Workshop Face and Gesture Recognition, 1995, Zurich, Switzerland.

[Burl 96] M.C. Burl and P. Perona. Recognition of Planar Object Classes. IEEE Comp. Society Conf. on Computer Vision and Pattern Recognition, CVPR 96, San Francisco, CA, June 1996

[Burl 97] M.C. Burl, M. Weber, T.K. Leung, and P. Perona. Recognition of Visual Object Classes. Chapter to appear in: From Segmentation to Interpretation and Back: Mathematical Methods in Computer Vision, Springer Verlag, 1997.

[Burl 98] M.C. Burl, M. Weber, and P. Perona. A Probabilistic Approach to Object Recognition Using Local Photometry and Global Geometry. Proc. of the 5th European Conf. on Computer Vision, ECCV 98.

[De Bonet 97] Jeremy S. De Bonet and Paul Viola. Structure Driven Image Database Retrieval. Neural Information Processing 10 (1997).

[LeCun 95] Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time-series. In M. A. Arbib, editor, The Handbook of Brain Theory and Neural Networks. MIT Press, 1995.

[Lowe 99] David G. Lowe. Object Recognition from Local Scale-Invariant Features. Proceedings of the International Conference on Computer Vision, Corfu, Greece (September 1999), pp. 1150-1157.

[Matan 92] Ofer Matan, Christopher J. C. Burges, Yann LeCun, and John S. Denker. Multi-digit recognition using a space displacement neural network. In J. M. Moody, S. J. Hanson, and R. P. Lippman, editors, Neural Information Processing Systems, volume 4. Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[Moghaddam 97] Moghaddam and Pentland. Probabilistic Visual Learning for Object Representation. PAMI 1997.

[Rikert 99] Thomas D. Rikert, Michael J. Jones, and Paul Viola. A Cluster-Based Statistical Model for Object Detection. ICCV 1999.

[Rowley 98a] H. A. Rowley and S. Baluja and T. Kanade. "Neural network-based face detection,", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp, 23-28, 1998.

[Rowley 98b] Henry A. Rowley, Shumeet Baluja and Takeo Kanade, Rotation Invariant Neural Network-Based Face Detection, Computer Vision and Pattern Recognition, 1998, pages 38-44.

[Rowley 99] Henry A. Rowley. Neual network-based face detection. Carnegie Mellon University PhD Thesis, CMU-CS-99-117, May 1999.

[Schiele 00] Bernt Schiele and James L. Crowley. Recognition Without Correspondence using Multidimensional Receptive Field Histograms IJCV 36(1), pp. 31-52, 2000.

[Schneiderman 98] H. Schneiderman and T. Kanade. "Probabilistic Modeling of Local Appearance and Spatial Relationships for Object Recognition." IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 45-51. 1998. Santa Barbara, CA.

[Schneiderman 00a] H. Schneiderman and T. Kanade. "A Statistical Method for 3D Object Detection Applied to Faces and Cars". To appear in IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2000)

[Schneiderman 00b] H. Schneiderman. "A Statistical Approach to 3D Object Detection Applied to Faces and Cars." Ph.D. Thesis. CMU-RI-TR-00-06. 2000.

[Schneiderman 03] H.Schneiderman. "Learning Statistical Structure for Object Detection." Computer Analysis of Images and Patterns (CAIP), 2003

[Schneiderman 04a] Henry Schneiderman and Takeo Kanade. “Object Detection Using the Statistics of Parts.” International Journal of Computer Vision 56 (3): 151-177, February - March, 2004.

[Schneiderman 04b] H. Schneiderman. "Feature-Centric Evaluation for Cascaded Object Detection." IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2004.

[Schneiderman 04c] H. Schneiderman. "Learning an Restricted Bayesian Network for Object Detection." IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2004.

[Tieu 00] Kinh Tieu and Paul Viola. Boosting Image Retrieval. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2000.

[Valliant 94] R. Vaillant, C. Monrocq, and Y. LeCun. Original approach for the localisation of objects in images. IEE Proc on Vision, Image, and Signal Processing, 141(4):245-250, August 1994.

[Viola 01] Paul Viola and Michael J. Jones. Robust Real-time Object Detection. Compaq Cambridge Research Laboratory Technical Report, Febtruary 2001.

[Wu 03] J. Wu, J. M. Rehg, and M. D. Mullin. Learning a Rare Event Detection Cascade by Direct Feature Selection. NIPS 2003.

7.2.2 Semi-Supervised Training for Object Detection / Recognition / Content Based Image Retrieval

[Baluja 98] Shumeet Baluja. Probabilistic Modeling for Face Orientation Discrimination: Learning from Labeled and Unlabeled Data. NIPS 1998.

[Barnard 01] Kobus Barnard and David Forsyth. "Learning the Semantics of Words and Pictures". International Conference on Computer Vision, vol 2, pp. 408-415, 2001.

[Barnard 03] Kobus Barnard, Pinar Duygulu, Nando de Freitas, David Forsyth, David Blei, and Michael I. Jordan, "Matching Words and Pictures," Journal of Machine Learning Research, Vol 3, pp 1107-1135, 2003.

[Cohen 03] I. Cohen, N. Sebe, F. Cozman, M. Cirelo, and T. Huang. Learning Bayesian network classifiers for facial expression recognition using both labeled and unlabeled data. CVPR 2003.

[Dong 03] A. Dong and B. Bhanu. A New Semi-Supervised EM Algorithm for Image Retrieval. CVPR 2003.

[Fei-Fei 03] Li Fei-Fei, Rob Fergus, and Pietro Perona. A Bayesian Approach to Unsupervised One-shot Learning of Object Categories. ICCV 2003. Pages 1134-1141.

[Fergus 03] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. CVPR 2003.

[Levin 03] Anat Levin, Paul Viola, and Yoav Freund. Unsupervised Improvement of Visual Detectors using Co-Training. ICCV 2003. Pages 626-633.

[Miller 00] E. Miller, N. Matsakis, P.Viola. "Learning from One Example Through Shared Densities on Transforms." CVPR 2000.

[Miller 03] E. Miller and C. Chefd'hotel. "Practical Non-parametric Density Estimation on a Transformation Group for Vision." CVPR 2003.

[Rosenberg 02] Charles Rosenberg and Martial Hebert. "Training Object Detection Models with Weakly Labeled Data". BMVC 2002.

[Rousson 03] M. Rousson, T. Brox, R. and Deriche. Active Unsupervised Texture Segmentation on a Diffusion Based Feature Space. CVPR 2003.

[Schmid 01] Cordelia Schmid. Constructing models for content-based image retrieval. CVPR 2001.

[Selinger 01] Andrea Selinger. Minimally Supervised Acquisition of 3D Recognition Models from Cluttered Images. CVPR 2001.

[Weber 99] M. Weber, M. Welling, and P. Perona. Unsupervised learning of Models for Visual Object Class Recognition. 6th Annual Joint Symposium on Neural Computation, JNSC99, Pasadena, CA, May 1999.

[Weber 00a] M. Weber, M. Welling, and P. Perona. Towards Automatic Discovery of Object Categories. Proc. IEEE Comp. Soc. Conf. Comp. Vis. and Pat. Rec., CVPR20000, June 2000.

[Weber 00b] M. Weber, M. Welling, and P. Perona. Unsupervised Learning of Models for Recognition. Proc. 6th Europ. Conf. Comp. Vis., ECCV2000, Dublin, Ireland, June 2000.

[Wu 01] Ying Wu, Thomas Huang, and Kentaro Toyama. Self-Supervised Learning for Object Recognition based on Kernel Discriminant-EM Algorithm. ICCV 2001.

7.2.3 Related Vision Topics

[Freeman 91] William T. Freeman and Edward H. Adelson. The Design and Use of Steerable Filters. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 13, No. 9, pp. 891-906, September 1991.

[Rosenberg 00] Charles Rosenberg. "Image Color Constancy Using EM and Cached Statistics", ICML-2000, pp. 799-806, July 2000.

[Rosenberg 01] Charles Rosenberg, Martial Hebert, and Sebastian Thrun. "Image Color Constancy Using KL-Divergence." ICCV 2001.

[Rosenberg 03] Charles Rosenberg, Thomas Minka, and Alok Ladsariya. "Bayesian Color Constancy with Non-Gaussian Models." NIPS 2003.

[Torralba 01] A. Torralba, P. Sinha. Statistical context priming for object detection. Proceedings of the IEEE International Conference on Computer Vision, ICCV01. (pp. 763-770), Vancouver, Canada. 2001.

[Torralba 03a] A. Torralba. Contextual priming for object detection. International Journal of Computer Vision, 53 (2): 153-167, July 2003.

[Torralba 03b] Antonio Torralba, Kevin Murphy, William Freeman, Mark Rubin. Context-based vision system for place and object recognition. ICCV'03.