UNIVERSITY OF CALIFORNIA, SAN DIEGO

Training Discriminative Computer Vision Models with Weak Supervision

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Boris Babenko

Committee in charge:

Professor Serge Belongie, Chair
Professor Gert Lanckriet
Professor David Kriegman
Professor Virginia de Sa
Professor Lawrence Saul

2012

The dissertation of Boris Babenko is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California, San Diego

2012

DEDICATION

To my parents.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

This dissertation represents a culmination of my formal education. I feel it is appropriate, then, to use this space to acknowledge and thank everyone who has played an important pedagogical role in my life.

First and foremost I express my gratitude to my family, whose continued support, sacrifices and love over the years have enabled me to pursue my dreams. My parents, Aleksandr and Galina, have encouraged curiosity and creativity throughout my childhood, from painting with watercolors to playing guitar. My older sister, Katya, has been a life-long role model, and is responsible for initiating my interest in computer science.

Next I would like to acknowledge a group of people who rarely get the credit they deserve: my elementary, middle and high school teachers. John Farley, Janet Eichsteadt, and Cathleen Zeleski had a particular influence on me and helped develop my love for math and science.

I owe gratitude to a number of people at UC San Diego, where I completed both my undergraduate and graduate studies. Serge Belongie has been an invaluable mentor, colleague and friend throughout my years in college and graduate school, and I could not have asked more of a research advisor. I know of no professor who is more dedicated to the well-being and enrichment of his students (both undergraduate and graduate). Oftentimes being surrounded by smart people has a higher educational value than any set of lectures; I thank the following people who I have collaborated with and spent many hours chatting with: Shiaokai Wang, Steve Branson, Carolina Galleguillos, Nakul Verma, Catherine Wah, Vincent Rabaud, Andrew Rabinovich, and Ming-Hsuan Yang. In particular, I am indebted to Piotr Dollar who took me under his wing when I joined the Belongie lab; it has been a pleasure collaborating with him over the years. I thank David Kriegman, Lawrence Saul, Gert Lanckriet and Virginia de Sa, not only for being on my thesis committee, but for teaching the courses that formed the foundation of my graduate work.

Last but not least I would like to thank the close friends I have made and kept over the years: Rita Beransky, Mark Kraz, Bryan Wasserman, Sawako

Sonoyama, Siavosh Aghassy. Each of these individuals has taught me valuable life lessons; I hope and expect to learn much more in the years to come.

Portions of this dissertation are based on papers that I have co-authored with others. Listed below are my contributions to each of these papers.

- Parts of Chapter 2 and Chapter 4 are based on the paper "Simultaneous Learning and Alignment: Multi-Instance and Multi-Pose Learning" by B. Babenko, P. Dollár, Z. Tu, and S. Belongie [13]. The dissertation author contributed to algorithm development, implemented code and experiments, and contributed to writing of the paper.

- Parts of Chapter 3 are based on the paper "Multiple Component Learning for Object Detection" by P. Dollár, B. Babenko, S. Belongie, P. Perona, and Z. Tu [42]. The dissertation author contributed to algorithm development, implemented parts of the code and experiments, and contributed to writing of the paper.

- Parts of Chapter 3 are based on the paper "Weakly supervised object recognition and localization with stable segmentations" by C. Galleguillos, B. Babenko, A. Rabinovich, and S. Belongie [57]. The dissertation author implemented parts of the system, and contributed to writing of the paper.

- Parts of Chapter 3 are based on the paper "Visual Tracking with Online Multiple Instance Learning" [15] and "Robust Object Tracking with Online Multiple Instance Learning" [16] by B. Babenko, M. -H. Yang, and S. Belongie. The dissertation author developed the algorithm and experiments, and wrote most of the paper.

- Parts of Chapter 4 are based on the paper "Similarity Metrics for Categorization: from Monolithic to Category Specific" by B. Babenko, S. Branson, and S. Belongie [11]. The dissertation author developed the algorithm and experiments, and wrote most of the paper.

- Chapter 5 are based on the paper "Multiple instance learning with manifold bags" by B. Babenko, N. Varma, P. Dollar, and S. Belongie [14]. The dis-

sertation author developed the algorithm and experiments, and wrote most of the paper.

VITA

2006	B. S. in Computer Science *summa cum laude*, University of California, San Diego
2008	M. S. in Computer Science, University of California, San Diego
2011	Ph. D. in Computer Science, University of California, San Diego

PUBLICATIONS

N. Ben-Haim, B. Babenko, and S. Belongie. Improving Web-based Image Search via Content Based Clustering. In *SLAM (CVPR Workshop)*, 2006.

B. Babenko, P. Dollár, and S. Belongie. Task Specific Local Region Matching. In *International Conference on Computer Vision (ICCV)*, 2007.

P. Dollár, B. Babenko, S. Belongie, P. Perona, and Z. Tu. Multiple Component Learning for Object Detection. In *European Conference on Computer Vision (ECCV)*, 2008.

C. Galleguillos, B. Babenko, A. Rabinovich, and S. Belongie. Weakly supervised object recognition and localization with stable segmentations. In *European Conference on Computer Vision (ECCV)*, 2008.

B. Babenko, P. Dollár, Z. Tu, and S. Belongie. Simultaneous Learning and Alignment: Multi-Instance and Multi-Pose Learning. In *Faces in Real-Life Images (ECCV Workshop)*, 2008.

B. Babenko, S. Branson, and S. Belongie. Similarity Metrics for Categorization: from Monolithic to Category Specific. In *International Conference on Computer Vision (ICCV)*, 2009.

B. Babenko, M. -H. Yang, and S. Belongie. Visual Tracking with Online Multiple Instance Learning. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.

Steve Branson, Catherine Wah, Boris Babenko, Florian Schroff, Peter Welinder, Pietro Perona, and Serge Belongie. Visual recognition with humans in the loop. In *European Conference on Computer Vision (ECCV)*, 2010.

B. Babenko, M. -H. Yang, and S. Belongie. Robust Object Tracking with Online Multiple Instance Learning. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, August 2011.

B. Babenko, N. Varma, P. Dollar, and S. Belongie. Multiple instance learning with manifold bags. In *International Conference on Machine Learning (ICML)*, 2011.

ABSTRACT OF THE DISSERTATION

Training Discriminative Computer Vision Models with Weak Supervision

by

Boris Babenko

Doctor of Philosophy in Computer Science

University of California, San Diego, 2012

Professor Serge Belongie, Chair

Statistical machine learning techniques have transformed computer vision research in the last two decades, and have lead to many breakthroughs in object detection, recognition and tracking. Such data-driven methods extrapolate rules from a set of labeled examples, freeing us from designing and tuning a system by hand for a particular application or domain. Discriminative learning methods, which directly learn to differentiate categories of data rather than modeling the data itself, have been shown to be particularly effective. However, the requirement of a large set of labeled examples becomes prohibitively expensive, especially if we consider scaling to a wide range of domains and applications. In this dissertation we explore weakly supervised methods of training discriminative models for a

number of computer vision applications. These methods require weaker forms of annotation that are easier and/or cheaper to obtain, and can learn in situations where the ground truth is inherently ambiguous. Many of the algorithms in this dissertation are based on a particular form of weakly supervised learning called Multiple Instance Learning (MIL). Our final contribution is a theoretical analysis of MIL that takes into account the characteristics of applications in computer vision and related areas.

1 Introduction

The integration of statistical learning methods into computer vision has proven to be transformative. Rather than relying on hand tuned models of real-world objects (e.g. [143]), learning based methods rely on large collections of *labeled data* to automatically train a system. Indeed some hand-tuned design must still be done to construct relevant statistics and features (e.g. [43]); however, learning based methods can adapt to new data domains, and can automatically discover patterns that would otherwise be difficult to find by hand.

Learning methods are typically split into two broad categories: discriminative and generative. *Generative* methods attempt to model the distribution of the data and use the model to make classification decisions. On the other hand, *discriminative* methods aim to directly minimize classification error. The latter class of methods (e.g. Support Vector Machines [129], Boosting [52]) have resulted in huge performance improvements in many vision problems. Perhaps the best example of this is the problem of face detection: after the seminal work of Viola & Jones [135] face detection has reached performance levels compatible with consumer products (e.g. most point-and-shoot cameras today come with face detection built in). Many other problems in computer vision have benefited from discriminative learning techniques: pedestrian detection [38, 49, 43], image categorization [75, 134], pose estimation [45], etc. While generative methods are appealing in that they could be used to synthesize or hallucinate novel data examples (much like the human brain), at present their performance is often significantly lower than that of discriminative approaches.

The main bottleneck of training a statistical model lies in obtaining a large amount of labeled data. For this reason, many different types of *weakly* supervised

learning paradigms have recently been explored [139, 50, 48, 136, 6, 90, 37, 134, 97]. The term "weakly supervised" is overloaded in the literature, but in this dissertation we define it as any form of supervised learning that involves some form of missing labels (i.e. latent variables). The advantages of such approaches are as follows:

- **Reducing labor costs.**
  The most obvious advantage for weak supervision is the reduction of required labeling. Since labeling is typically done by humans, strongly supervised learning can often become intractable due to the amount of labeling required.

- **Human precision and label ambiguity.**
  While human labels remain the gold standard in supervision, such labels are far from being perfect. In fact, certain labeling tasks are inherently *ambiguous*. For instance, consider the task of putting a bounding box around a face in an image. A single correct bounding box does not exist (e.g. if we consider we consider two bounding boxes that are just 1 pixel off from each other, a human providing ground truth would have a hard time telling the two apart). For this reason, it may be better to simply ask the user a binary question, "is there a face in this image?", and leave the precise location of the face as a latent variable.

- **Suboptimal labels.**
  In certain scenarios the variables we require for strong supervision may have no semantic meaning. Consider a bird detection task where we ask human annotators to provide locations of bird body parts (e.g. wing, beak). In this application, there may be a particularly salient region of birds that does not correspond to a semantic "part" – for example, the space between the bird legs. Therefore it may be difficult to specify which parts we desire to have labeled. In the weakly supervised setting we would simply ask for a binary presence/absence label (similar to the face detection scenario we described above), and allow the part identities and locations to remain latent in the hopes that an algorithm will recover the best regions automatically.

The constellation model of object recognition is perhaps the earliest and best known weakly supervised method in computer vision [139]. However, the constellation model and its many extensions [50, 48, 37] are generative style approaches. As mentioned earlier, for many vision applications, discriminative methods have been shown to be superior. Some recent work has started exploring discriminative approaches to weakly supervised learning [136, 6, 134, 97]. Many of these works employ a particular weakly supervised learning paradigm called Multiple Instance Learning [41] (discussed in much detail in later Chapters of this dissertation), and this dissertation extends these ideas further.

The goal of this dissertation is to explore methods for training *discriminative* models with weak supervision, with applications to computer vision problems. We consider a wide variety of applications including object detection and recognition, object tracking, and image categorization. Since many of the applications in this dissertation are based on Multiple Instance Learning, our final contribution is a theoretical analysis of this learning paradigm that takes into the particular characteristics of applications in computer vision and related areas.

2 Preliminaries

In this chapter we will present our notation and review some concepts and algorithms that will be useful in the following chapters. We will begin with a review of traditional, or "strongly", supervised learning, introduce weakly supervised learning (including Multiple Instance Learning), and review the Gradient Boosting framework that will be used in other parts of the dissertation.

2.1 Strongly Supervised Learning

In the strongly supervised learning setting a learner receives labeled examples (x_i, y_i) where $x \in \mathcal{X}$ (typically $\mathbb{R}^d$ for some integer $d > 0$) and $y_i \in \mathcal{Y}$ is a corresponding label. Binary classification, where $\mathcal{Y} = \{-1, +1\}$ or $\mathcal{Y} = \{0, 1\}$, is the simplest and most commonly studied variation; this will also be the case we focus on in most of this dissertation.

Upon receiving labeled examples, the learner returns a classifier $h \in \mathcal{H}$, where $\mathcal{H}$ is some hypothesis class and $h : \mathcal{X} \to \mathcal{Y}$. Training *discriminative* models typically involves searching for classifier that minimizes a loss function that is related to training error:

$$\mathcal{L}(h) = \sum_{i=1}^{n} \ell(y_i, h(x_i)) \tag{2.1}$$

where $\ell(\cdot)$ takes the true and predicted labels for a particular training example as inputs, and outputs a large positive value if the two disagree. Since $0-1$ loss (i.e. $\ell(\cdot)$ returns 0 if the prediction matches the ground truth, and 1 otherwise) is difficult to optimize directly, a *convex* upper bound to the training error is

commonly used instead. Examples of this include the hinge loss (used in Support Vector Machines [129]), and the exponential loss (used in AdaBoost [52]). Finally, note that for many learning algorithms the classifier also returns a *confidence* score in addition to the label. This is especially common for binary classification, since a binary label can be obtained from the confidence score via a simple threshold.

2.2 Weakly Supervised Learning

Now consider a learning problem where some portion of training examples is kept hidden from the learner. Our classifier will now take as input both the observed input x, and its corresponding latent variable $z \in \mathcal{Z}$: $h(x, z)$. At run time, given a novel input x, we must find the value of z, which typically consists of scanning over all possible values to see where the classifier responds most confidently: (e.g. going back to our face detection example, z would specify a particular region in an image, and we would scan over all such regions). In other words, we must solve the following optimization problem to produce a label for an input x using a trained classifier: $\max_z h(x, z)$ (here we are abusing the notation slightly, in that the max is taken with respect to the confidence that the classifier h outputs, rather than the label[1]). We must also make the corresponding change to the loss function:

$$\mathcal{L}(h) = \sum_{i=1}^{n} \ell(y_i, \max_z h(x, z)) \tag{2.2}$$

Unfortunately, the above objective function is no longer convex. Two popular heuristic approaches exist in the literature for performing the above optimization. The first is to use an alternating procedure: the first step consists of fixing the latent variables and finding a classifier, and the second step consists of finding the values for the latent variables given the latest classifier; the two steps are repeated in sequence. Note that when the latent variables are fixed, the learning problem reduces to the strongly supervised case. This type of procedure is similar in spirit

[1] A more precise but messier way of writing this is $h(x, \text{argmax}_z \tilde{h}(x, z))$, where $h(x, z)$ returns a label, and $\tilde{h}(x, z)$ returns a confidence score.

to the EM algorithm [40], as well as many recent weakly supervised learning techniques, including latent SVM [49, 146]. The other approach is to use a simple optimization procedure like gradient descent on the objective function (with a minor caveat that the function should be made smooth via approximating any hard max or min operators; see Section 2.3). Since the objective function is not convex, neither procedure is guaranteed to converge to a global optimum. Nevertheless, we find that in practice these methods work well. In this dissertation we will primarily use the latter heuristic in developing algorithms.

2.2.1 Multiple Instance Learning

Multiple Instance Learning (MIL) is a popular form of weakly supervised learning introduced in [41]. In this paradigm the learner receives *sets* of inputs, or *bags*, and a label for each bag. In particular, the bag is labeled positive if and only if it contains at least one positive input (though extensions exist [134], the problem is typically defined for binary classification). More formally, each bag is defined as $X_i = \{x_{i1}, \ldots, x_{im}\}$, where $x_{ij} \in \mathcal{X}$. Every instance has some true label $y_{ij} \in \{-1, +1\}$, but its value is not known during training (i.e. it is latent). The label of the bag can be expressed as follows:

$$y_i = \max_j(y_{ij}) \tag{2.3}$$

The goal is then to learn a classifier h using only bag labels y_i, such that $\max_j(h(x_{ij})) = y_i$ (and similarly on unseen data).

In the last decade there have been numerous algorithms proposed for training a classifier in the MIL setting. Some of these are extensions of popular supervised learning algorithms (e.g. boost [136], support vector machines (SVM) [6, 5, 30], neural networks [107], logistic regression [108]), while others are brand new procedures specific to MIL (e.g. [91, 90, 148, 21]). In this dissertation we will use the boosting algorithm for MIL called MILBoost [136], a review of which is presented in Section 2.3.1.

There are many applications in computer vision that can be solved with the MIL framework, a couple of which will be discussed in this dissertation. We

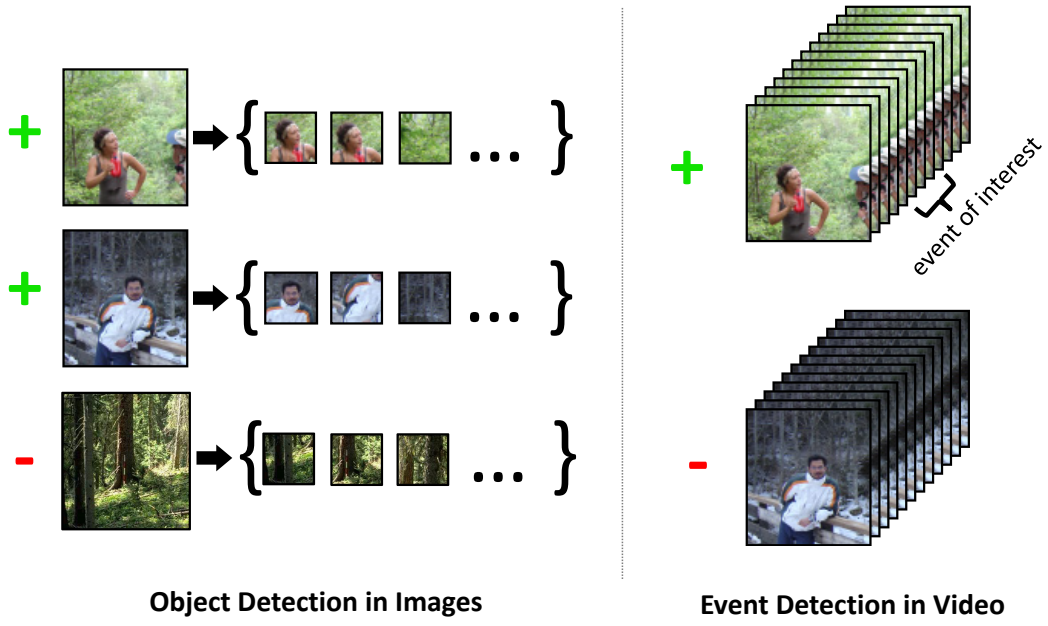


Figure 2.1: **Examples of computer vision applications of Multiple Instance Learning (MIL)**. See text for details.

will defer detailed discussion of related work to the appropriate chapters, but for now consider two examples for illustration (c.f. Fig. 2.1). In the first example we would like to train an object detector. Normally, this would require a dataset of cropped objects of interest as well as images that do not contain the objects. However, suppose we have images that are labeled as positive if they *contain* the object of interest, but its precise location is unknown. We can train a detector via MIL by treating each image as a bag, and breaking it up into many regions and treating each region as an instance – notice that at least one region from a positive bag should be the object of interest. In practice these regions are formed either by sliding a rectangular window around the image, or via some segmentation algorithm (see Fig. 2.2); a more detailed discussion of this will be the focus of Section 3.1 in Chapter 3. The other scenario in Fig. 2.1 is that of event detection in video. Here each video is treated as a bag, and is labeled positive if some event of interest occurs in it. Each bag is broken up into many instances (i.e. short video clips) via sliding window. Similar to the case of object detection, here a positive bag will contain a clip of the event of interest.

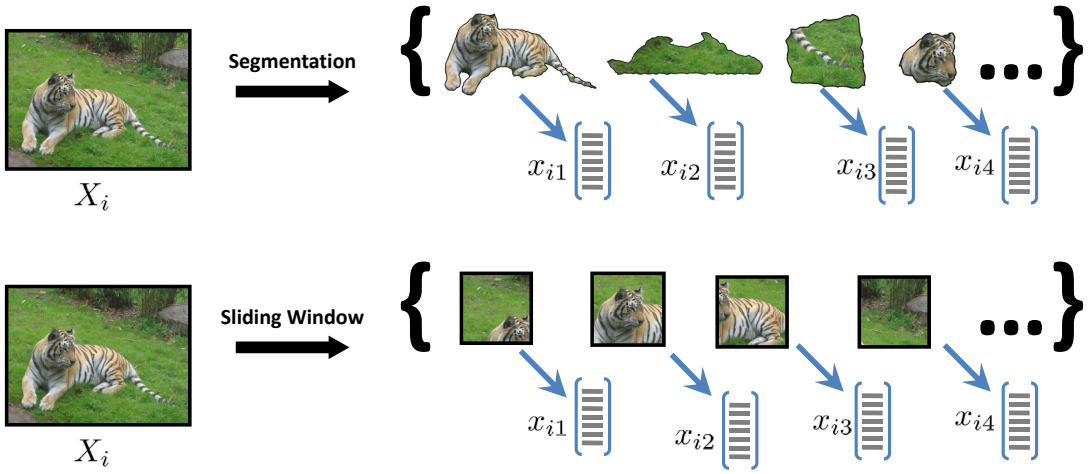


Figure 2.2: **Forming a bag out of an image**. See text for details.

2.3 Gradient Boosting

Boosting [52] has been shown to be a very successful approach for binary classification. In boosting, the goal is to train a classifier of the form:

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x), \tag{2.4}$$

where each $h_t : \mathcal{X} \to \mathcal{Y}$ is a *weak learner* whose performance may only be slightly above chance, and the α_t weigh the weak learners' relative importance. Boosting combines multiple weak learners into a single *strong* classifier with low error. Training proceeds sequentially. In each phase incorrectly classified examples receive more weight; details vary according to boosting algorithm.

Friedman [53] proposes an elegant method for deriving boosting algorithms for a wide range of loss functions. The general idea is to optimize a loss function $\mathcal{L}(H)$ by performing gradient descent on H (known as gradient descent in function space). Intuitively, the idea is to always select the h_t that most reduces the loss on the training data. Finding the best h_t proceeds in two stages: (1) compute the *optimal* weak classifier response and (2) from the available candidates select the weak classifier that best approximates the optimal response.

More formally, we can consider H as an n vector whose i^{th} component H_i has the value $H(x_i)$. The loss $\mathcal{L}$ is a function over H, and the goal is to

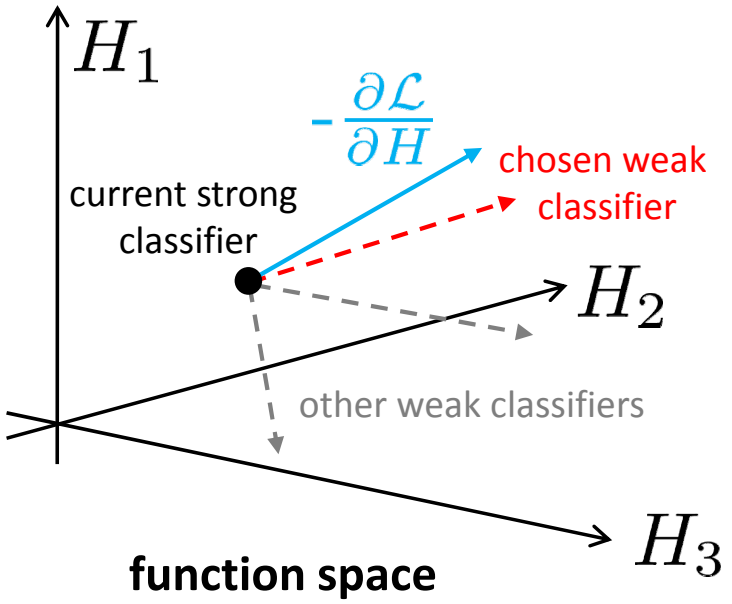


Figure 2.3: **Gradient Boosting Framework**. See text for details.

minimize $\mathcal{L}(H)$ with respect to H via gradient descent. In other words to find the optimal weak classifier response in each phase t we compute $-\frac{\partial \mathcal{L}}{\partial H}$, which is a vector with components $w_i \equiv -\frac{\partial \mathcal{L}}{\partial H_i}$. Ideally we would select h_t such that $w_i = h_t(x_i)$ $\forall i$; however, in practice we are limited in the choice of h_t. Therefore, Friedman proposes to find the h_t which is as close as possible to the gradient in function space:

$$h_t = \underset{h}{\operatorname{argmax}} \sum_{i=1}^{n} w_i h(x_i) \tag{2.5}$$

The general process is illustrated in Fig. 2.3. Once we compute h_t, the step size α_t can be found via a line search (again minimizing $\mathcal{L}(H)$). Putting everything together, we obtain the boosting procedure in Algorithm 1. Remaining details follow.

Log Likelihood

The negative log likelihood is a commonly used loss function for boosting, and we will use it in some of the later chapters. We begin by deriving a standard boosting algorithm using this loss function and gradient descent boosting. Note

Algorithm 1 Boosting via gradient descent in function space.

Input: Dataset $\{x_1, \ldots, x_n\}$, $\{y_1, \ldots, y_n\}$,$y_i \in \{-1, 1\}$

1: **for** $t = 1$ to T **do**
2: Compute weights $w_i = -\frac{\partial \mathcal{L}}{\partial H_i}$
3: Train weak classifier h_t using weights $|w_i|$
$h_t = \text{argmin}_h \sum_i \mathbf{1}(h(x_i) \neq y_i)|w_i|$
4: Find α_t via line search to minimize $\mathcal{L}(H)$
$\alpha_t = \text{argmin}_\alpha \mathcal{L}(H + \alpha h_t)$
5: Update strong classifier $H \leftarrow H + \alpha_t h_t$.
6: **end for**

that a related derivation is done in [53]; furthermore, the popular LogitBoost algorithm [55] uses similar criteria, though the actual optimization procedure differs.

If we define $p_i \equiv p(y_i = 1|x_i)$, the negative log likelihood is expressed as follows:

$$\mathcal{L}(H) = -\sum_{i=1}^{n} \Big(\mathbf{1}(y_i = 1) \log p_i + \mathbf{1}(y_i = -1) \log(1 - p_i) \Big). \tag{2.6}$$

To derive a boosting algorithm we must first define p_i in terms of $H_i = H(x_i)$. We follow [53, 55] and define:

$$p_i = \sigma\left(2H_i\right), \tag{2.7}$$

where $\sigma(v) = \frac{1}{1+exp(-v)}$ is the sigmoid. Note that $\sigma(v) \in [0, 1]$ and $\frac{\partial \sigma}{\partial v} = \sigma(v)(1 - \sigma(v))$. Finally, taking the derivatives:

$$\frac{\partial \mathcal{L}}{\partial p_i} = \begin{cases} \frac{-1}{p_i} & \text{if } y_i = 1 \\ \frac{1}{1-p_i} & \text{if } y_i = -1 \end{cases} \tag{2.8}$$

$$\frac{\partial p_i}{\partial H_i} = 2p_i(1 - p_i), \tag{2.9}$$

and using the chain rule, we get $w_i = -\frac{\partial \mathcal{L}}{\partial H_i} = -\frac{\partial \mathcal{L}}{\partial p_i}\frac{\partial p_i}{\partial h_i}$:

$$w_i = \begin{cases} 2(1 - p_i) & \text{if } y_i = 1 \\ -2p_i & \text{if } y_i = -1 \end{cases} \tag{2.10}$$

Intuitively, negatives with large p_i and positives with small p_i receive high weight.

Training weak classifier h_t

In the original gradient boosting paper weak classifiers are chosen by optimizing Eqn. (2.5), while in most boosting algorithms weak classifiers are chosen by optimizing weighted error:

$$h_t = \arg\min_h \sum_i \mathbf{1}(h(x_i) \neq y_i) w_i, \tag{2.11}$$

where $y_i \in \{-1, 1\}$. The criterion in Eqn. (2.11) is more common and many existing learning algorithms approximately minimize this cost function. Here we show that for binary classifiers the first cost function can be converted to the second. Let $y'_i = \text{sign}(w_i)$ (typically $y'_i = y_i$), and $w'_i = |w_i| / \sum_i (|w_i|)$. Then:

$$\begin{aligned} h_{t+1} &= \arg\min_h \sum_i -h(x_i) w_i \\ &= \arg\min_h \sum_i (2 \cdot \mathbf{1}(h(x_i) \neq y'_i) - 1)|w_i| \\ &= \arg\min_h \sum_i 2 \cdot \mathbf{1}(h(x_i) \neq y'_i)|w_i| \\ &= \arg\min_h \sum_i \mathbf{1}(h(x_i) \neq y'_i) w'_i \end{aligned}$$

This transformation allows us to use existing learning algorithms, e.g. decision stumps, that minimize weighted error or some approximation to it to train h_t.

Initial distribution

It is often useful to train with an initial (prior) distribution over the data, e.g. if more negative than positive training examples are available. Let ω_i be the prior on the i^{th} example. We can modify $\mathcal{L}$ as follows:

$$\mathcal{L}(H) = -\sum_{i=1}^{n} \omega_i \Big(\mathbf{1}(y_i = 1) \log p_i + \mathbf{1}(y_i = -1) \log(1 - p_i) \Big). \tag{2.12}$$

Observe that doubling the weight of x_i is the same as having two copies of x_i in the training data. The derivative of $\mathcal{L}$ becomes $\frac{\partial \mathcal{L}}{\partial p_i} = \frac{-\omega_i}{p_i}$ if $y_i = 1$ and $\frac{\omega_i}{1-p_i}$ if $y_i = -1$. Modifying the corresponding equations for w_i is trivial.

Real vs. Discrete h_t

Thus far we have assumed that the weak classifiers are binary, i.e. $h_t(x) \in \{-1, 1\}$. If, however, h_t outputs a real valued confidence or score, we can use this score directly. The procedure outlined in Fig. 1 remains unchanged. The alternative is to just use $\text{sign}(h_t(x))$. Using the terminology of [55], we refer to these variations as *real* and *discrete*, respectively.

Approximating Max

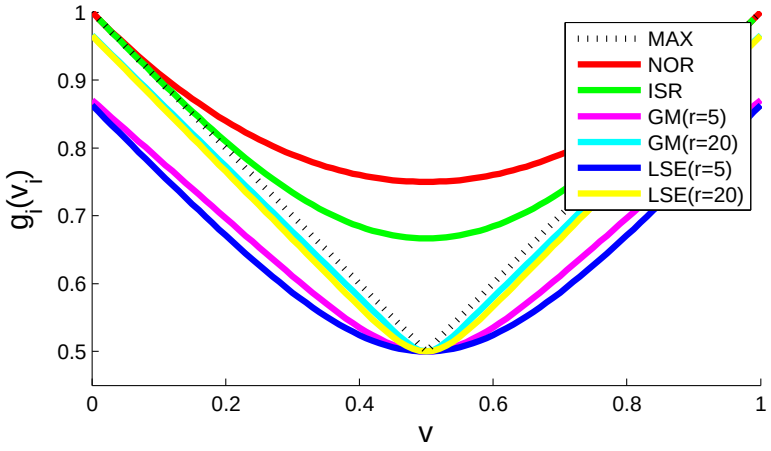


Figure 2.4: **Smooth approximations to a** max **operator**. See text for details.

Table 2.1: Four max approximations $g_\ell(v_\ell) \approx \max_\ell(v_\ell)$.

	$g_\ell(v_\ell)$	$\partial g_\ell(v_\ell)/\partial v_i$	domain
LSE	$\frac{1}{r}\ln\left(\frac{1}{m}\sum_\ell \exp(r v_\ell)\right)$	$\frac{\exp(r v_i)}{\sum_\ell \exp(r v_\ell)}$	$[-\infty, \infty]$
GM	$\left(\frac{1}{m}\sum_\ell v_\ell^r\right)^{\frac{1}{r}}$	$g_\ell(v_\ell)\frac{v_i^{r-1}}{\sum_\ell v_\ell^r}$	$[0, \infty]$
NOR	$1 - \prod_\ell (1 - v_\ell)$	$\frac{1-g_\ell(v_\ell)}{1-v_i}$	$[0, 1]$
ISR	$\frac{\sum_\ell v'_\ell}{1+\sum_\ell v'_\ell}$, $v'_\ell = \frac{v_\ell}{1-v_\ell}$	$\left(\frac{1-g_\ell(v_\ell)}{1-v_i}\right)^2$	$[0, 1]$

We now present an overview of differentiable approximations of the *max* operator. The general idea is to approximate the max over $\{v_1, \ldots, v_m\}$ by a

differentiable function $g_\ell(v_\ell)$, such that:

$$g_\ell(v_\ell) \approx \max_\ell(v_\ell) = v_* \tag{2.13}$$

$$\frac{\partial g_\ell(v_\ell)}{\partial v_i} \approx \frac{\mathbf{1}(v_i = v_*)}{\sum_\ell \mathbf{1}(v_\ell = v_*)} \tag{2.14}$$

Intuitively, if v_i is the unique max, then changing v_i changes the max by the same amount, otherwise changing v_i does not affect the max.

A number of approximations for g have been proposed. We summarize the choices used here in Table 2.1: a variant of log-sum-exponential (LSE) [25, 107], generalized mean (GM), noisy-or (NOR) [136], and the ISR model [71, 136]. In Fig. 2.4 we show the different models applied to $(v, 1-v)$ for $v \in [0, 1]$.

LSE and GM each have a parameter r the controls their sharpness and accuracy; $g_\ell(v_\ell) \to v_*$ as $r \to \infty$ (note that large r can lead to numerical instability). For LSE one can show that $v_* - \log(m)/r \leq g_\ell(v_\ell) \leq v_*$ [25] and for GM that $(1/m)^{1/r} v_* \leq g_\ell(v_\ell) \leq v_*$, where $m = |v_\ell|$. NOR and ISR are only defined over $[0, 1]$. Both have probabilistic interpretations and work well in practice; however, these models are best suited for small m as $g_\ell(v_\ell) \to 1$ as $m \to \infty$. Finally, all models are exact for $m = 1$, and if $\forall \ell \; v_\ell \in [0, 1]$, then $0 \leq g_\ell(v_\ell) \leq 1$ for all models.

2.3.1 MILBoost

We now turn to Multiple Instance Learning (MIL), and re-derive and generalize MILBoost [136]. As in the previous cases, we would like to optimize the negative log likelihood $\mathcal{L}$. To do so we must define $p_i \equiv p(y_i = 1|X_i)$, the probability of a bag X_i. We begin by defining the probability $p_{ij} \equiv p(y_{ij} = 1|x_{ij})$ of an instance x_{ij} in the same manner as before: $p_{ij} = \sigma(2H_{ij})$, where $H_{ij} = H(x_{ij})$. Given the instance probabilities, we define the bag probability p_i as the *maximum* over the instance probabilities p_{ij}. Using the max approximator g in place of the max we write:

$$p_i = g_j(p_{ij}) = g_j(\sigma(2H_{ij})) \tag{2.15}$$

This can be viewed as the probabilistic approximation of Eqn (2.3).

Algorithm 2 MILBoost

Input: Dataset $\{X_1, \ldots, X_n\}$, $\{y_1, \ldots, y_n\}$,$y_i \in \{-1, 1\}$

1: **for** $t = 1$ to T **do**
2: Compute weights $w_{ij} = -\frac{\partial \mathcal{L}}{\partial H_{ij}}$
3: Train weak classifier h_t using weights $|w_{ij}|$
$h_t = \text{argmin}_h \sum_{ij} \mathbf{1}(h(x_{ij}) \neq y_i)|w_{ij}|$
4: Find α_t via line search to minimize $\mathcal{L}(H)$
$\alpha_t = \text{argmin}_\alpha \mathcal{L}(H + \alpha h_t)$
5: Update strong classifier $H \leftarrow H + \alpha_t h_t$.
6: **end for**

Algorithm details are given in Algorithm 2. The optimization procedure for MIL is similar to the optimization procedure for regular boosting described in Section 2.3: compare Algorithms 1 and 2. Table 2.2 shows the expressions for instance weights for different variations of g.

Table 2.2: MIL equations for w_{ij} for different choices of g.

	$y_i = -1$	$y_i = 1$
LSE	$\frac{-2p_{ij}(1-p_{ij})}{1-p_i} \frac{\exp(rp_{ij})}{\sum_\ell \exp(rp_{i\ell})}$	$\frac{2p_{ij}(1-p_{ij})}{p_i} \frac{\exp(rp_{ij})}{\sum_\ell \exp(rp_{i\ell})}$
GM	$\frac{-2p_i}{1-p_i} \left(\frac{(p_{ij})^r - (p_{ij})^{r+1}}{\sum_\ell (p_{i\ell})^r} \right)$	$2 \left(\frac{(p_{ij})^r - (p_{ij})^{r+1}}{\sum_\ell (p_{i\ell})^r} \right)$
NOR	$-2p_{ij}$	$\frac{2p_{ij}(1-p_i)}{p_i}$
ISR	$\frac{-2\chi_{ij}p_i}{\sum_\ell \chi_{i\ell}}, \chi_{ij} = \frac{p_{ij}}{1-p_{ij}}$	$\frac{2\chi_{ij}(1-p_i)}{\sum_\ell \chi_{i\ell}}, \chi_{ij} = \frac{p_{ij}}{1-p_{ij}}$

Portions of this chapter are based on the following publications:

- "Multiple Component Learning for Object Detection" by P. Dollár, B. Babenko, S. Belongie, P. Perona, and Z. Tu [42]. The dissertation author contributed to algorithm development, implemented parts of the code and experiments, and contributed to writing of the paper.

3 Weakly Labeled Location

In this chapter we will explore weakly supervised computer vision systems where the precise location of an object or its parts is latent. Our applications will include object detection & recognition, as well as object tracking.

3.1 Object Recognition and Localization

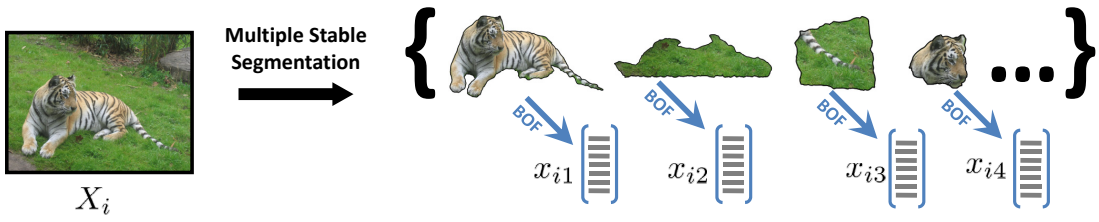


Figure 3.1: **Object recognition and localization with Multiple Instance Learning and Stable Segmentations (MILSS)**. We use Multiple Stable Segmentations to break an image up into many overlapping segments. This particular choice of segmentation procedure increases the chance of extracting the correct segment from an image, while keeping the total number of segments manageable. For each segment we compute the Bag-of-features representation, and these feature vectors are passed into MIL for training

The goal of object categorization is to locate and identify instances of an object category within an image. This task is challenging in real world scenes since objects may vary in scale, position, and viewpoint; in addition, they may be surrounded by background clutter, occluded by other objects, and obscured by poor image quality. To model these sources of variability, traditional approaches to object categorization require large labeled data sets of fully annotated training

images. Typical annotations in these "fully" labeled data sets provide masks or bounding boxes that specify the locations, scales, and orientations of objects in each training image. Though extremely valuable, this information is prone to error and is expensive to obtain. Without this information, however, traditional approaches to object categorization tend to learn spurious models of background artifacts, leading to lower accuracy during testing.

Some approaches for object categorization have successfully learned object models from weakly labeled data [50, 101, 111, 121, 124]. Weakly labeled training examples indicate which objects of interest are present in training images without specifying the pixels that are associated with them. From weakly labeled examples, the existing methods use standard techniques in statistical learning to model the "essence" of each category. Popular approaches include constellation models [50, 49, 37], region based methods [101, 124] and latent models such as pLSA and LDA, with bag of visual words [111, 121, 137]. While they excel at exploiting correlations between different image patches, they suffer from computationally expensive inference and background noise that is learned as part of the category model.

Recently, Multiple Instance Learning (MIL) models have been applied to object detection and recognition [32, 104, 134, 6, 136]. In the MIL framework a weakly labeled image is broken down into many pieces which constitute the instances of a bag. As discussed before, this the pieces are typically rectangular bounding boxes or segments (see Fig. 2.2). The bounding box approach fails to accurately localize the object of interest, especially when the object is non-rectangular (e.g. a snake or a airplane). On the other hand, segmentation is well known to be a difficult problem, and can often under or over segment the object of interest. If segmentation fails to accurately crop out the object of interest, a positive bag may end up containing no positive instance breaking the assumption of MIL (we will explore the theoretical foundations of this phenomenon in Chapter 5).

In this work we propose a pipeline for training object classifiers using MIL. In order to get accurate localization and avoid the problems of segmentation failure

we propose the use of Multiple Stable Segmentation (MSS). We demonstrate the efficiency and accuracy of our framework on two databases that present significant intra-class variation: Caltech 4 [50] and a landmark image database, Landmarks-18. The Caltech dataset, although highly popular in the computer vision community, is a rather artificial dataset, where objects often appear in isolation and with uniform backgrounds. The Landmarks-18 dataset on the other hand, is taken directly from common web albums and contains instances of popular landmarks in cluttered scenes with variable viewpoint, weather, and illumination.

3.1.1 Proposed Pipeline

We now described our proposed pipeline, which we call Multiple Instance Learning with Stable Segmentation (MILSS).

Object Classification with MIL

The problem of learning an object classifier from weakly labeled data can be elegantly framed as multiple instance learning. During training it is known for each image whether a certain object category is present, but the exact location of that object is unknown. If we split an image $\mathcal{I}_i$ into J multiple regions or segments $\{s_{i1}, s_{i2}..., s_{iJ}\}$, we can assume that one of the segments contains the object of interest. For each image we are given a category label $y_i = \{c_1, c_2, ..., c_C\}$; however, since the MIL problem is defined only for binary classification, we will train our classifiers in a one versus all manner. If we define $y_{ik} \in \mathbf{1}(y_i = c_k)$ to be a binary label indicating the presence of category k in image i, we can train C different classifiers. For each category k, we train a classifier $H^k : s \rightarrow \{1, 0\}$ using the training data set $\{(\mathcal{I}_1, y_{ik}), ...\}$. In practice, since our problem is multi-class it is more useful for us to also obtain the probability of the segment containing an object category k, $p(c_k|s)$. The boosting algorithm for MIL developed in [136] provides us with an effective way of learning these functions; we refer the reader to Section 2.3.1 for details on the algorithm.

Multiple Stable Segmentations

In the weakly supervised setting, for an image with a particular label we know that *some* region of the image contains the corresponding object. The number of possible image regions is, for all practical purposes, infinite. We therefore require a method of splitting the image up into a manageable number of "reasonable" regions or segments. Traditional segmentation methods [120, 87] will often lead to over or under segmentation, and require many parameters to be set a priori (e.g. typically these algorithms require the number of segments; of course, this is very difficult to estimate for a novel image). Instead, we turn to Multiple Stable Segmentations [105]. The method of multiple stable segmentations uses stability as a heuristic for a particular set of parameters, cue weightings and a model order.

For each choice of parameters (e.g., cue combinations $\vec{p}$ and number of segments q), the image is segmented using Normalized Cuts [120, 87]. The segmentation is considered stable if small perturbations of the image do not yield substantial changes in the segmentation. The image is perturbed and segmented T times and the following score is evaluated:

$$\Phi(q,\vec{p}) = \frac{1}{n - \frac{n}{q}} \left(\sum_{i=1}^{n} \sum_{j=1}^{T} \delta_{ij} - \frac{n}{q} \right), \text{where } \delta_{ij} = \begin{cases} 1 & \text{if } i = \text{j} \\ 0 & \text{otherwise} \end{cases}. \tag{3.1}$$

Here n is the number of pixels and δ_{ij} is equal to 1 if the i-th pixel is mapped to a different segment in the j-th perturbed segmentation, and zero otherwise. Thus Φ is a properly normalized[1] measure of the probability of a pixel to change label due to a perturbation of the image. Segmentations with a high stability score are retained. Notice that, in general, there may exist several stable segmentations for an image.

Segment Representation

To represent each segment in feature space we use the bag of features model (BoF) [50] to capture appearance information. Recently, the BoF image representation has found widespread application in object categorization due to its sim-

[1] In particular Φ ranges in $[0, 1]$ and it is not biased towards a particular value of q.

plicity and efficiency. We first detect salient regions in the segment and compute a feature vector for each region. These feature vectors are then mapped to a vocabulary of "visual words" which are computed using vector quantization. The BoF representation of an image segment is then a histogram of these visual words (often referred to as a signature).

Classifying Images and Localizing Objects

Given an image $\mathcal{I}_i$ we compute q stable segmentations resulting in multiple segments $\{s_{i1}, s_{i2}..., s_{iJ}\}$. For each segment s_{ij} we compute a BoF signature, with each signature corresponds to an instance of the bag. A segment s_{ij} is classified as follows:

$$y_{ij} = \underset{k}{\operatorname{argmax}}\, p(c_k|s_{ij}), \tag{3.2}$$

where $p(c_k|s_{ij})$ is the probability of the segment s_{ij} belonging to the category c_k. We classify an image $\mathcal{I}_i$ as proposed by [106]:

$$y_i = \underset{k}{\operatorname{argmax}} \sum_{j=1}^{J} p(c_k|s_{ij}). \tag{3.3}$$

The task of object localization generally corresponds to placing a bounding box, or preferably the actual object outline, around the object within the image. Since our framework uses segments for categorization, we utilize segment boundaries that yield highest recognition score in order to describe object locations [50]. For evaluating our localization performance for an image $\mathcal{I}_i$ classified overall as y_i and segment labels y_{ij}, we look for segments with labels such that $y_{ij} = y_i$. Then we check for overlapping segments and return the first n unique segment boundaries, with $n \ll J$.

3.1.2 Experimental Results

To evaluate the MILSS framework, we compare our approach to the state-of-the-art methods in object categorization. Existing MIL-based approaches often use the COREL dataset to evaluate their models for image categorization. However,

since we concentrate on object categorization, the performance of our approach is evaluated on Caltech 4 and a new dataset Landmarks-18.

Caltech 4 Dataset

Caltech 4 [50] is a well established dataset and is a standard benchmark for object categorization. Although simple, we utilize this dataset as a means of comparison with Mil-based methods. Following the experimental set up of [32, 104], we perform a category versus background classification. Table 3.1(a) presents the results of categorization accuracy for our method. Results are compared to existing MIL-based image categorization models [32, 104] and a non-MIL-based approach of [18]. The presented results are competitive with the rest of the algorithms. The average categorization accuracy for MILSS as well as ConMIL is 98%; while MILES is 97% and Bar-Hillel *et al.*'s algorithm is 93%. Note that the highest performance is achieved in the Airplanes category given that the stable segmentations were able to separate the background from the objects accurately.

Table 3.1: **Caltech 4 results**. (a) Comparison of categorization results between our framework, MIL-based models [32, 104] and a traditional object categorization approach [18] for Caltech 4 categories. Results in **bold** indicate the highest performance for each category. (b) MILSS Confusion matrix between the four categories for multi-class object recognition.

	Airplanes	Cars	Faces	Motorbikes
Training data	400	400	218	400
MILSS	**1**	.971	.976	.972
ConMIL [104]	.992	**.984**	.976	**.987**
MILES [32]	.980	.945	**.995**	.967
Bar-Hillel [18]	.897	.977	.917	.931

(a)

	A	F	L	M
Training data	400	218	100	400
Airplanes (A)	**.98**	.00	.01	.01
Faces (F)	.01	**.99**	.00	.00
Leopards (L)	.05	.01	**.93**	.01
Motorbikes (M)	.01	.00	.01	**.97**

(b)

Table 3.1(b) reports accuracy for multi-class object categorization. Instead of considering a background category, images belonging to each category acted as negative examples for models trained on the other categories. We compare our method to existing non-MIL-based object recognition frameworks: the dependent Hierarchical Dirichlet process (DHDP) [137] and constellation of parts model [50]. As shown in Table 3.2(a), MILSS reports an average recognition accuracy of 97%

while DHDP reports 98%. Looking closely at the categories, MILSS outperforms DHDP in three out of four of them. The Leopards category seems to be the most challenging for our framework, since it contains fewer images than the rest of the categories (100 for training and 100 for testing). In order to improve these results we could easily augment our training set with images from public repositories, as manual labeling is not required.

Table 3.2: **Caltech 4 Results**. (a) Results of multiple object categorization models for four Caltech categories. We compare our results to those of non MIL-based models. Results in **bold** indicate the highest performance for each category. (b) Average localization results of MILSS for four categories of Caltech.

	Airplanes	Faces	Leopards	Motorbikes	Mean
Training data	400	218	100	400	
MILSS	**.977**	**.986**	.927	.971	0.965
DHDP [137]	.961	.978	**1**	.967	0.976
Fergus [50]	.888	.862	-	**.977**	0.909

(a)

	MILSS
Airplanes	.932
Faces	.902
Leopards	.891
Motorbikes	.859
Mean	.896

(b)

In a multi-class setting, localization accuracy of MILSS is 90% on the Caltech 4 dataset. Our localization results are presented in detail in Table 3.2 (b). To quantify the accuracy of object localization we adopt the methodology of [50] and consider the overlap $\alpha = \frac{B \cap B_{gt}}{B \cup B_{gt}}$. Note that our method may be at a disadvantage in cases where the objects' contour areas B are smaller than the ground truth bounding box B_{gt}; thus it is difficult to make a direct comparison with the results in [50]. Since our method localizes objects using segment boundaries, the location and extent of the object is captured more precisely than those with bounding boxes, see Fig. 3.3.

Landmark Database

With the increasing popularity of digital photography and the user's desire to share their pictures in web albums, recognition of destinations and landmarks has become an interesting problem. Recognizing objects in real world images is a challenging task, as images are presented at a variety of viewpoints, scales, and illuminations; noise, background clutter, and occlusions also make the problem

more difficult. Since photo-sharing sites are a vast resource of weakly labeled image data, we easily gather large datasets to evaluate our framework.

Figure 3.2: **Landmarks-18 Dataset**. Two examples are shown per landmark and each row shows 9 categories. **Top row**: Arc de Triomphe, Ayres Rock, Bellsouth Building, Brandenburg Gate, Buckingham Palace, Burjal Arab, CN Tower, Centre Pompidou and Chrysler Building. **Bottom row**: Church Savior Spilled Blood, Eiffel Tower, Liberty Bell, Lincoln Memorial, Lincoln Memorial Statue, London Tower Bridge, Space Needle, Sydney Opera House and Taipei 101.

Here we introduce a new dataset called Landmarks-18, consisting of 18 different categories of landmarks, provided by Google Research and collected from public web albums. Landmarks-18 captures much more significant intra-class variability than standard benchmark datasets for object recognition. Figure 3.2 demonstrates the diversity of landmarks in the dataset.

Here we performed two different multi-class categorization experiments on Landmarks-18. Each experiment considers 10 different categories, where images in each category were divided randomly into 80%/20% for training and testing respectively. Experiments were performed with 5-fold cross validation to obtain statistically relevant average categorization results. Figure 3.4 shows confusion matrices for both experiments. The results show that Landmarks-18 is much more difficult for categorization than Caltech 4, due to the challenging characteristics

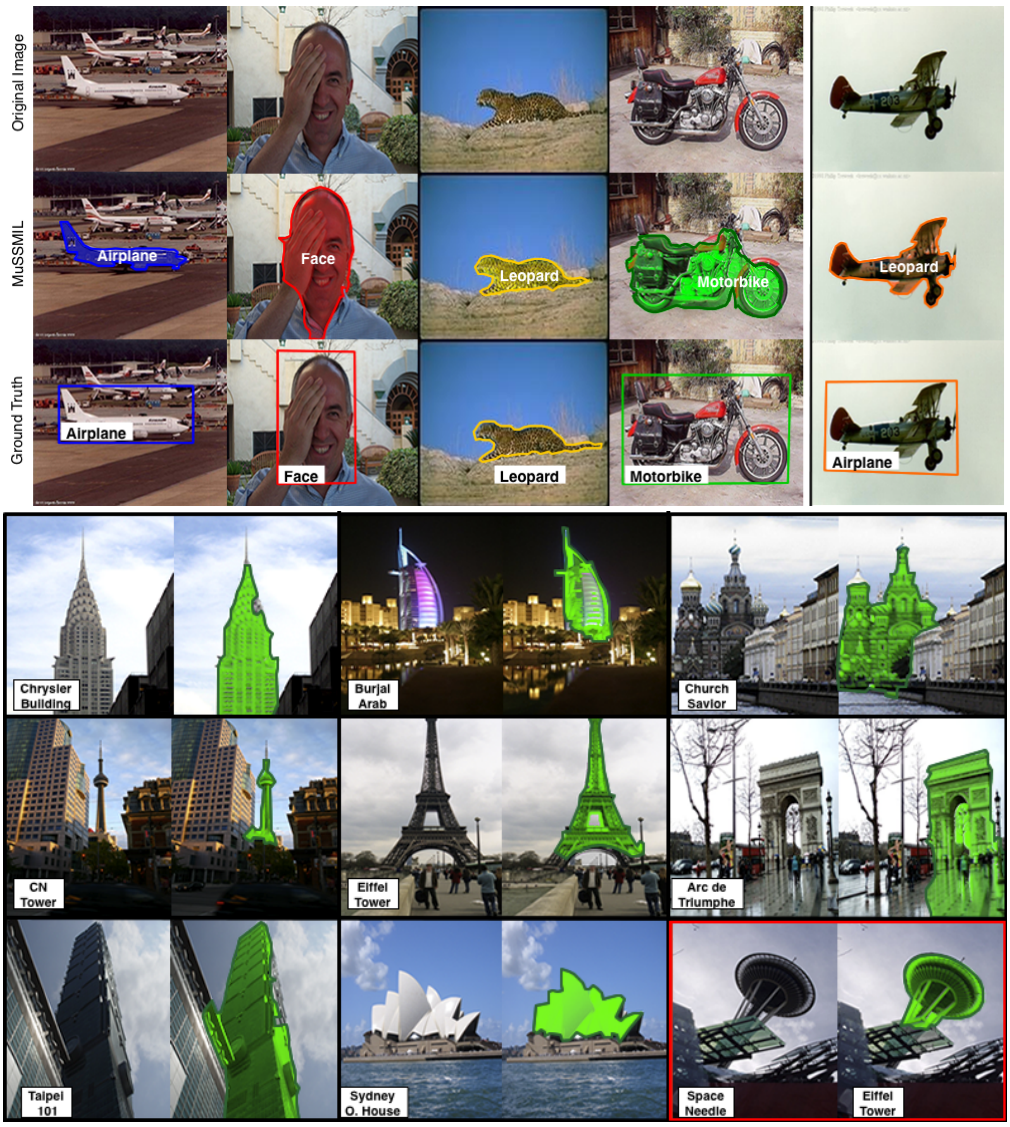


Figure 3.3: **Top image**: Examples of Caltech test images. First three columns correspond to successful image categorization and localization of objects in the scene. Last column correspond to a false positive. **Bottom row**: Examples of Landmarks-18 test images. Green segments represent the image region with the highest probability of being the landmark. Images enclosed by a red rectangle correspond to a false positive.

of its images and the larger number of classes. Despite this, MILSS achieves high categorization accuracy in both experiments. The outcome of both experiments indicate that Eiffel Tower, Taipei101, and Bellsouth Building are the most challenging categories. The main source of low recognition accuracy isnbetween visually similar categories such as Bellsouth Building vs. Chrysler Building. For this dataset we were unable to compare our results to other MIL-based categorization systems as code was not available.

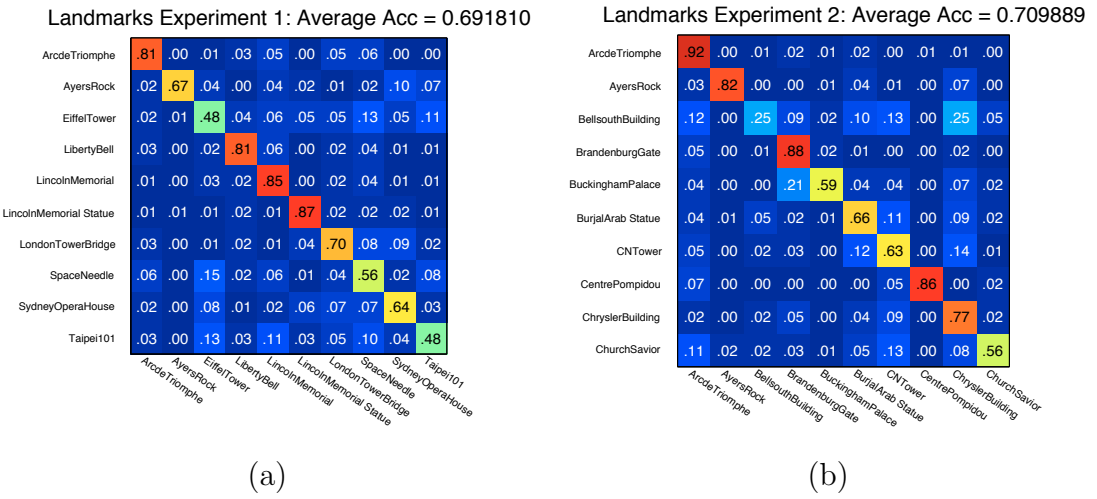


Figure 3.4: **Confusion matrices of categorization accuracy for the Landmark-18 dataset**. (a) Experiment 1; (b) Experiment 2.

To evaluate the importance of the multiple stable segmentations within MILSS, we also experimented with two different single segmentations ($q = 4$ and $q = 6$) using Normalized Cuts [120]. Figure 3.5 shows the average categorization accuracy for each method using 5-fold cross validation. With multiple stable segmentations categorization performance is improved in almost all categories. The average categorization accuracy for $q = 4, 6$ and multiple segmentations is 58.3%, 61.8% and 71.0% respectively. The total number of segmentations extracted from an image plays an important role in categorization accuracy. As noted by others, as the number of segments per image increases, so does the chance of having a segment that represents the object accurately [106, 88]. We believe that multiple stable segmentations provide a way of gathering the most meaningful segments, as

is reflected in our results.

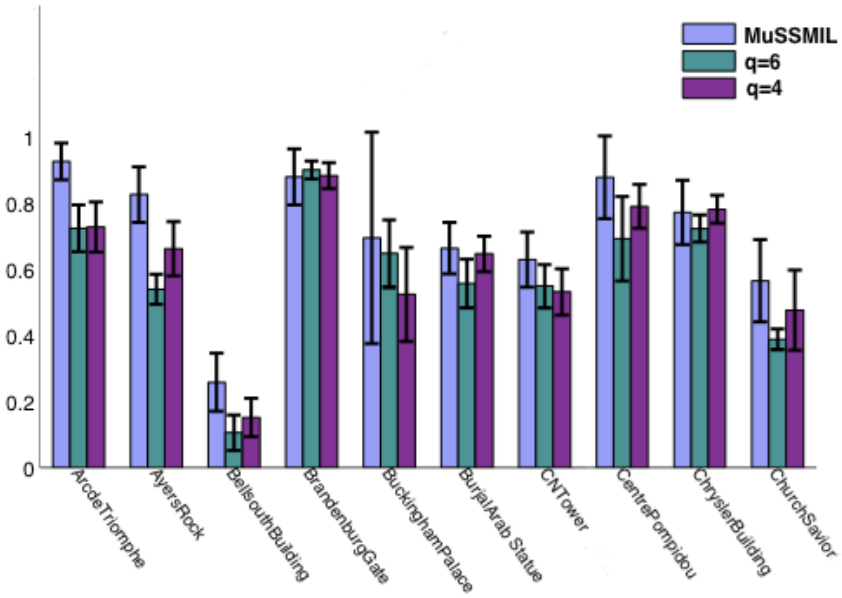


Figure 3.5: **Three different types of region extraction**. two single segmentations with number of segments equal to 4 and 6, and multiple stable segmentations. The average categorization accuracy for $q = 4, 6$ and multiple segmentations is 58.3%, 61.8% and 71.0% respectively. Multiple stable segmentations outperform (on average) all the other methods.

Implementation Details

The stability based image segmentation was implemented using Normalized Cuts [120, 36]. Five iterations, combining brightness and texture cues with $\vec{p} = \{0.4, 0.5, 0.6, 0.7\}$ were used to sample the parameter space. For the categorization experiments done for Caltech and Landmarks-18, we computed 5 different segmentations with $q = 2, \ldots, 6$ with a total of 20 segments per image. Computing a single segmentation takes about 20-30 seconds per image. For the BoF model we computed 5000 random SIFT [86] features at multiple scales (from 12 pixels up to the full image size) for each image segment. Visual words are obtained computing a hierarchical K-means with $K = 17$ and three levels. The computation of SIFT descriptors and signatures takes about 1 second per segment in a MATLAB/C

implementation. Constructing the vocabulary tree takes 40-50 minutes for ten categories. Training time for MilBoost on four Caltech categories takes about 1 day using 500 weak classifiers. Using ten categories of Landmarks-18 MilBoost take less than a day of training using 200 weak classifiers. Classification of all test images for ten categories is done in 0.5 seconds. All above operations were performed on a Pentium 2.8 GHz.

3.2 Object Detection with Parts

In this chapter we focus on the problem of detection of non-ridig objects. The main challenge lies in the amount of variability of non-rigid object categories, and the fact that most commonly used features are not invariant to such deformation. A growing trend in object detection is to construct part-based object detectors based on the observation that parts of an object *are* typically rigid. We follow this trend and propose an approach that uses Multiple Instance Learning (MIL) to train component classifiers (corresponding to rigid parts) from data where only object labels in the form of bounding boxes are provided, but part locations are unknown. The resulting method, Multiple Component Learning (MCL), learns individual component classifiers and combines these into an overall object classifier.

3.2.1 Related Work

A number of discriminative detection systems that learn from simple low-level features and large amounts of data have been proposed; typically their focus is either on the learning aspect [135, 128] or the design of appropriate features [38]. These methods require large amounts of labeled data to learn invariance to articulation, occlusion and intra-class variations. As mentioned, they have proven particularly successful for detecting rigid objects, achieving low false positive rates. MCL is similar to these methods in some regards, the key difference being that built into MCL is the domain knowledge that objects are composed of parts, that the mutual position of parts is somewhat variable and that parts may not all be visible. MCL is therefore much better suited for detecting articulated objects, e.g.

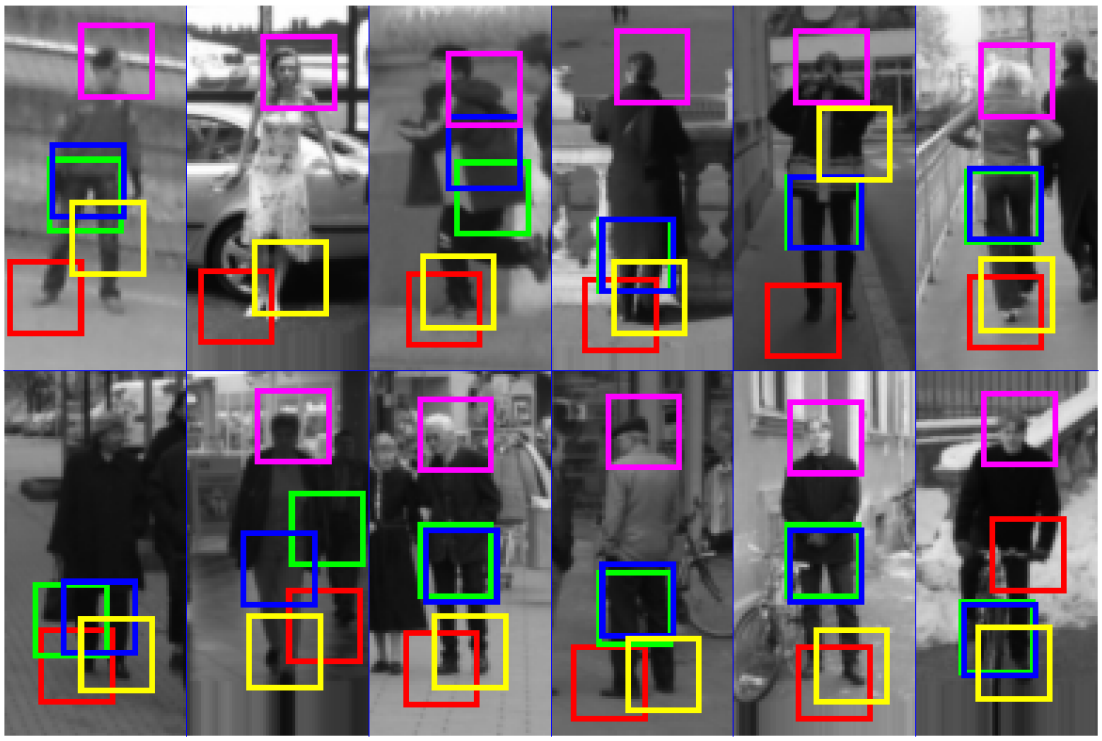


Figure 3.6: Response of first 5 learned components classifiers on randomly selected INRIA pedestrian test images (best viewed in color). At most one box is displayed per component after non-maximal suppression and thresholding. Three components correspond to semantically meaningful parts (head-magenta, left foot-red, right foot-yellow); 2 correspond to the region between between the legs. The components were learned with *no component labels* provided during training.

pedestrians, and remains robust in the presence of occlusion.

Part-based approaches have a rich history; one of the earliest approaches dates back to 1973 [51]. A number of different ways of extracting parts from images have since been proposed. One approach involves designing part detectors by hand [28] or providing a system with labeled part examples [93, 92]. Other than the obvious disadvantage of being labor intensive, these methods are restricted to using a limited number of possibly sub-optimal parts. An alternative approach involves searching for repeatedly occurring elements using different criteria such as frequency of appearance in the training data [4], lowering an empirical risk function [19], or increasing mutual information [133]. Unlike in MCL , the recurring elements in these methods are fairly simple, including edge fragments [4], Gaussian

models [19], or image fragments [133].

A simple but effective method of extracting parts is to crop small image patches, either using an interest point operator [139, 2] or by dense sampling [70]. The patches can be vector quantized to form 'codebooks' [139], and an image can then be represented using a 'bag of words' model [70]. Spatial information can be encoded using pairwise relationships [2] or with a spatial voting scheme [77]. Alternatively, a generative model can be used to model the object, e.g. the constellation model and its variants [81, 139, 37] have proven robust and capable of operating with little training data. Though effective in recognition tasks, these patch-based methods are limited by simple or fixed part appearance models, often relying on a patch distance measure in a predefined feature space, e.g. using normalized correlation or other patch descriptors. In MCL the component classifiers are learned from low level features, and although MCL requires more training data, the approach can lead to higher accuracy models.

Most closely related to our work is [49], which uses a formalism called latent SVMs to simultaneously learn part and object models. The resulting system is effective, though very different from our own. We emphasize that, as far as we are aware, aside from [49] MCL is the first part-based method that uses rich part appearance models without relying on part labels during training.

3.2.2 Multiple Component Learning

There are three primary challenges that we address in order to come up with our discriminative component-based object model. The first of these is how to learn a component classifier when only an object label is given. The second is how to learn *diverse* component classifiers given a method for learning a single component classifier. Finally, given multiple diverse component classifiers, we must combine these properly into an overall classifier for the object of interest. In this work we present a unified and effective solution to these challenges.

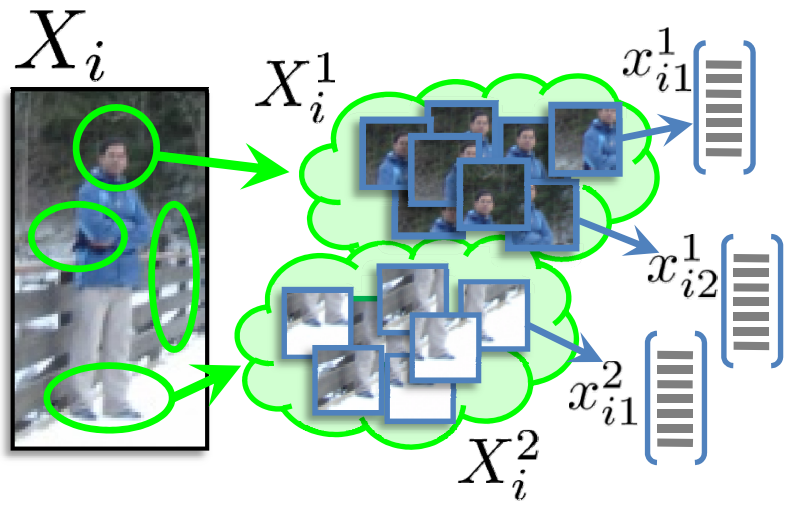


Figure 3.7: **Part-based detection with Multiple Component Learning (MCL)**. Pedestrians have limited articulation, so each component appears in a limited region of the image. Prior to training, T overlapping regions are randomly generated. For each region we extract patches, generating a list of image patch bags $\mathbf{X}_i = (X_i^1, \ldots, X_i^T)$.

Step 1: Learning a Single Component Classifier from Weakly Labeled Data

In order to learn component classifiers we turn to weakly supervised learning methods developed for object detection, where positive training images contain the object of interest, but, unlike the fully supervised case, the object location in each image is unknown [139, 37, 136]. Observe that learning a component classifier from images of objects where the components are in unspecified locations is an analogous problem. Thus, we can use weakly supervised learning to learn a single component classifier. Specifically we use multiple instance learning (MIL), creating a bag out of each training image (where instances are image patches of a fixed size). Notice that if we simply split our training images into patches and train a classifier using MIL the result will be a single component classifier (for example, if we use pedestrian images we may end up with a head detector because heads are particularly discriminative in pedestrian images).

Step 2: Learning Many Complimentary Component Classifiers

To learn a diverse collection of component classifiers we exploit the following fact: while the location of parts of an object may be vary in location, for many datasets we can assume that objects are *roughly* aligned. Thus, to learn multiple diverse classifiers we split each image, X_i, into p random regions $X_i^1, \ldots, X_i^T$. For each region we generate a bag of image patches $x_{i,j}^k$ and train a classifier using MIL. This results in T component classifiers that serve as our part detectors. See Fig. 3.7 for an illustration.

Step 3: Combining Component Classifiers

One we have trained T component classifiers we combine them into one object classifier as follows. First, note that our object classifier will itself be used in a sliding window fashion, classifying each image window in a large image. Each component classifier is trained to take an image *patch* as input. We use the component classifiers in a sliding window fashion to produce confidence scores for each pixel in the image window. These confidence maps are then treated as image channels, and passed into a Viola-Jones framework (i.e. Haar-like features are computed over these image channels). See Fig. 3.8 for an illustration.

An Alternative

An alternative to the last two steps is described in detail in [42]; we summarize it here briefly for completeness. Recall that in boosting, multiple weak learners, each of which may have fairly high error, are combined into a single strong classifier with a low overall error. Weak classifiers are trained sequentially with the weights of the training samples adjusted so that incorrectly classified examples receive more weight. Boosting is ideally suited both for learning diverse classifiers and combining them into an overall classifier. Thus, an alternative to the above method is to use component classifiers, trained using weakly supervised learning, as the weak classifiers in a boosting framework. This approach is more elegant than what we described above, but it is not immediately clear how to build in spatial information.

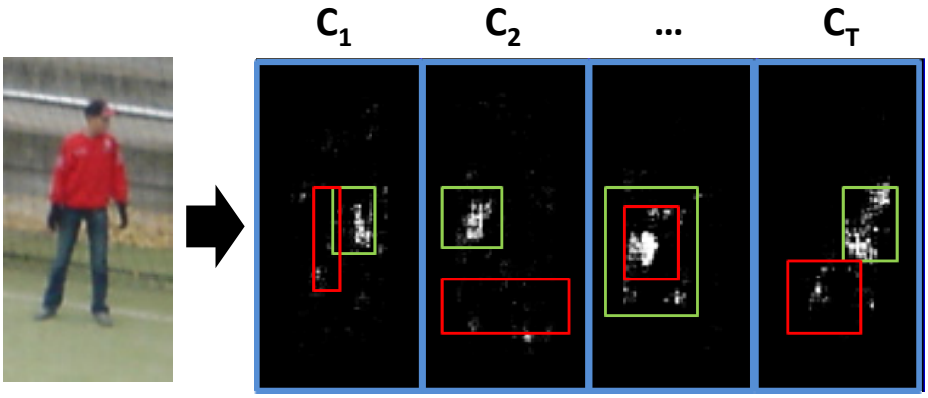


Figure 3.8: **Putting together component classifiers and incorporating spatial information**. Each of the T trained component classifiers is used in a sliding window fashion on an incoming image window. The output of these classifiers is treated as T image channels which are then passed into an object level classifier that computes Haar-like features [135] over these channels.

3.2.3 Experiments

We now present results on the INRIA pedestrian dataset [38]. A number of recent methods have targeted this data [38, 128, 114], We compare our results to each, using the training and evaluation methodology presented in [38], except to [114] as it appears the results reported in that work are inaccurate[2].

Here we provide details for training MCL on this dataset. Training windows are 128x64, and for each of these we extract ∼4K overlapping patches. We compute ∼10K random Haar features [135] per patch, using the original grayscale image, as well as gradient magnitude and 6 channels of gradient quantized by orientation. We use MILBoost to select 256 stump classifiers. All 2416 positive sets and ∼10K negative sets are used for training each MIL . Initially 50 binary masks in the shape of ellipses are randomly generated, and one MIL classifier is trained per mask, using sets generated from patches only in the masked region. The original binary masks are random and likely suboptimal; after training each MIL , we compute a new mask based on the MIL probability response images on the training positives and then retrain. This mask refinement step improves results. From among the 50 MIL

[2]See `http://www.cs.sfu.ca/~mori/research/papers/sabzmeydani_shapelet_cvpr07.html`

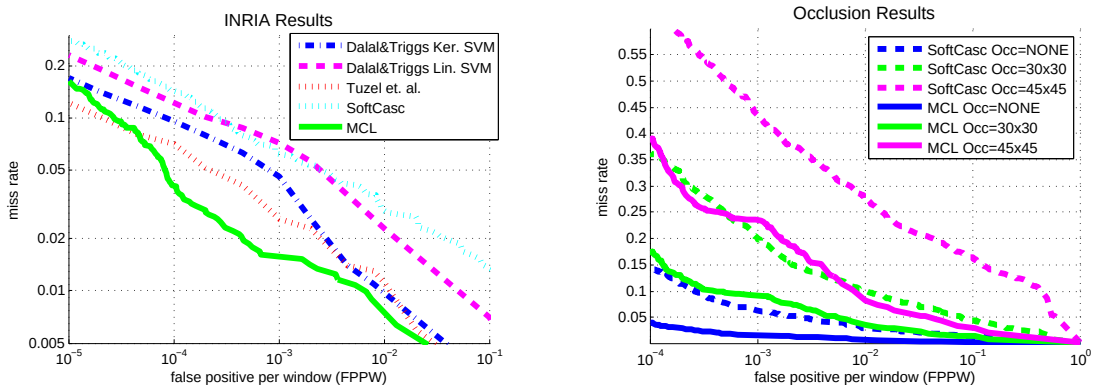


Figure 3.9: Results on *pedestrian detection*. **Left**: MCL outperforms all reported results at the time. At a false positive per window rate (FPPW) of 10^{-4}, a commonly used reference point, MCL has a miss rate of ∼4%, compared to ∼7% for [128] and ∼10% for [38]. For comparison we also implemented the *SoftCascade* approach described in [147], using the same candidate Haar features we use in MCL . Consistent with previously reported results, a cascade of Haars performs poorly. **Right**: Results on artificially generated occlusion. We overlaid random 30x30 or 45x45 patches into random locations in the pedestrian test images. We then regenerated the ROC curves for MCL as well as SoftCascade (which, like [38, 128], is not part-based). MCL on data with 30x30 performs similarly to SoftCascade on unoccluded data, and as the amount of occlusion increases, the gap between MCL and SoftCascade further increases.

candidate classifiers, AdaBoost is used to select the 20 best components. Next, we bootstrap ∼10K new negative windows from the training images from the false positives reported by MCL . The entire process is repeated 4 times to form a cascade. Our final MCL classifier is composed of 80 components (selected from 200 candidates).

Training takes 2 weeks on a modern PC (majority of time is spent in MILBoost training). Our classifier is composed of ∼20K Haar features (256 per MIL), compared to ∼6K in the Viola & Jones real-time face detector [135]; however, they are organized into multiple shallow cascades as opposed to a single deep cascade, so evaluation is slower. Simple feature sharing strategies should increase speed by an order of magnitude, reducing evaluation time to a few seconds per image.

The first five learned components are shown in Fig. 3.6. ROC curves comparing our method with Dalal and Triggs [38] and Tuzel et al. [128] are shown in

Fig. 3.9, left. Part-based approaches have a natural advantage when occlusions are present. We show the robustness of MCL to occlusions in Fig. 3.9, right.

Role of Data Alignment

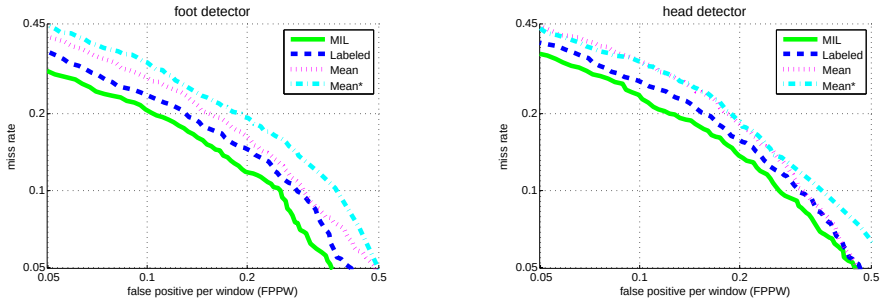


Figure 3.10: Effects of alignment on training part classifiers, see text for details.

We perform a final experiment to show that using *aligned* data results in higher accuracy (part) classifiers. Note that simultaneous alignment of articulated objects is often impossible without relying on a part-based model.

We labeled the head and feet in the 2416 INRIA training pedestrians. For each part, we trained a classifier using patches sampled from each pedestrian in 1 of 3 ways: at the *labeled* part location, at the *mean* part location, and in a region around the mean part location (*mil*). We used AdaBoost for the first two cases and the extended version of MILBoost for the third, using 256 Haars selected from an identical pool in each case. During testing, we applied each classifier to a region around the mean part location and recorded the maximum probability.

Results against bootstrapped negatives are shown in Fig. 3.10. Not surprisingly, *labeled* outperformed *mean*, showing alignment is beneficial. Additionally, *mean* performed better when we took the max probability over the region during testing rather than rely on the probability at the mean part location (*mean**). Also, consistent with the findings of [136], *mil* outperformed *labeled* even though it was trained in a weakly supervised manner (presumably our labeling is imperfect).

Together, these results make a strong case that data alignment is highly beneficial. As articulated objects typically cannot be fully aligned everywhere,

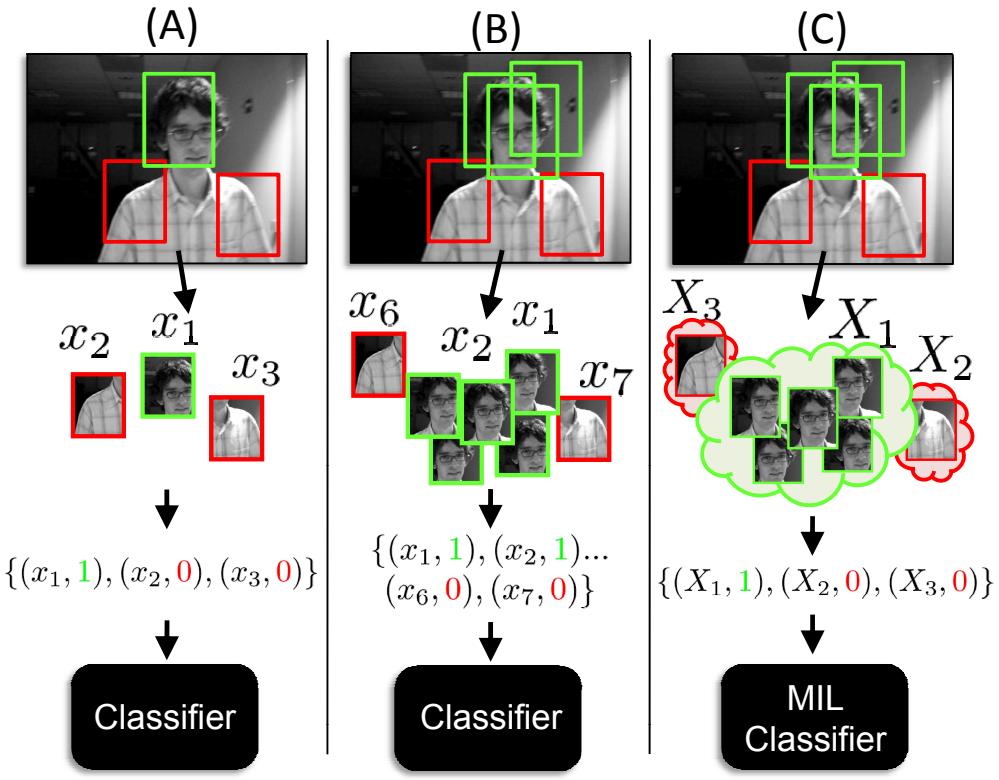


Figure 3.11: **Updating a discriminative appearance model**. (A) Using a single positive image patch to update a traditional discriminative classifier. The positive image patch chosen does not capture the object perfectly. (B) Using several positive image patches to update a traditional discriminative classifier. This can make it difficult for the classifier to learn a tight decision boundary. (C) Using one positive bag consisting of several image patches to update a MIL classifier. See Section 3.3.3 for empirical results of these three strategies.

MCL has an inherent advantage over methods that are not part-based.

3.3 Object Tracking

Object tracking is a well studied problem in computer vision and has many practical applications. The problem and its difficulty depend on several factors, such as the amount of prior knowledge about the target object and the number and type of parameters being tracked (e.g. location, scale, detailed contour). Although there has been some success with building trackers for specific object classes (e.g. faces [22], humans [68], mice [26], rigid objects [80]), tracking generic objects has

remained challenging because an object can drastically change appearance when deforming, rotating out of plane, or when the illumination of the scene changes. In this section we will study the problem of tracking an arbitrary object with no prior knowledge other than its location in the first video frame (sometimes referred to as "model-free" tracking). In particular, we will focus on the problem of tracking the location and scale of a single object, using a rectangular bounding box to approximate these parameters. It is plausible that the ideas presented here can be applied to other types of tracking problems like tracking multiple objects (e.g. [100]), tracking contours (e.g. [67, 132]), or tracking deformable objects (e.g. [115]), but this is outside the scope of our work. The goal of this work is to transfer the success of weakly supervised methods (in particular Multiple Instance Learning) in object detection to the problem of tracking.

3.3.1 Adaptive Appearance Models

A typical tracking system consists of three components: (1) an appearance model, which can evaluate the likelihood that the object of interest is at some particular location; (2) a motion model, which relates the locations of the object over time; and (3) a search strategy for finding finding the most likely location in the current frame. Here we will focus on the first of these three components; we refer the reader to [145] for a thorough review of the other components. An important choice in the design of appearance models is whether to model only the object [17, 110], or both the object and the background [82, 61, 84, 10, 9, 138, 35]. Many of the latter approaches have shown that training a model to separate the object from the background via a discriminative classifier can often achieve superior results. These methods are closely related to object detection – an area that has seen great progress in the last decade. In fact, some of these methods are referred to as "tracking-by-detection" or "tracking by repeated recognition" [95]. In particular, the recent advances in face detection [135] have inspired some successful real-time tracking algorithms [61, 84].

A major challenge that is often not discussed in the literature is how to choose positive and negative examples when updating the adaptive appearance

model. Most commonly this is done by taking the current tracker location as one positive example, and sampling the neighborhood around the tracker location for negatives. If the tracker location is not precise, however, the appearance model ends up getting updated with a sub-optimal positive example. Over time this can degrade the model, and can cause drift. On the other hand, if multiple positive examples are used (taken from a small neighborhood around the current tracker location), the model can become confused and its discriminative power can suffer (cf. Fig. 3.11 (A-B)). Alternatively, Grabner et al. [62] recently proposed a semi-supervised approach where labeled examples come from the first frame only, and subsequent training examples are left unlabeled. This method is particularly well suited for scenarios where the object leaves the field of view completely, but it throws away a lot of useful information by not taking advantage of the problem domain (e.g., it is safe to assume small interframe motion).

Object detection faces issues similar to those described above, in that it is difficult for a human labeler to be consistent with respect to how the positive examples are cropped. For this reason, as we have discussed earlier, Multiple Instance Learning is an appealing approach to training object detectors from weakly labeled data. In this paper we make an analogous argument, and propose to use a MIL based appearance model for object tracking (cf. Fig. 3.11(C)). In fact, in the object tracking domain there is even more ambiguity than in object detection because the tracker has no human input and has to bootstrap itself. Therefore, we expect the benefits of a MIL approach to be even more significant than in the object detection problem. In order to incorporate MIL into a tracker, an online MIL algorithm is required. The algorithm we propose (to our knowledge this is the first online MIL algorithm in the literature) is based on boosting and is related to the MILBoost algorithm [136] as well as the Online AdaBoost algorithm [103]. We present empirical results on challenging video sequences, which show that using an online MIL based appearance model can lead to more robust and stable tracking than existing methods in the literature.

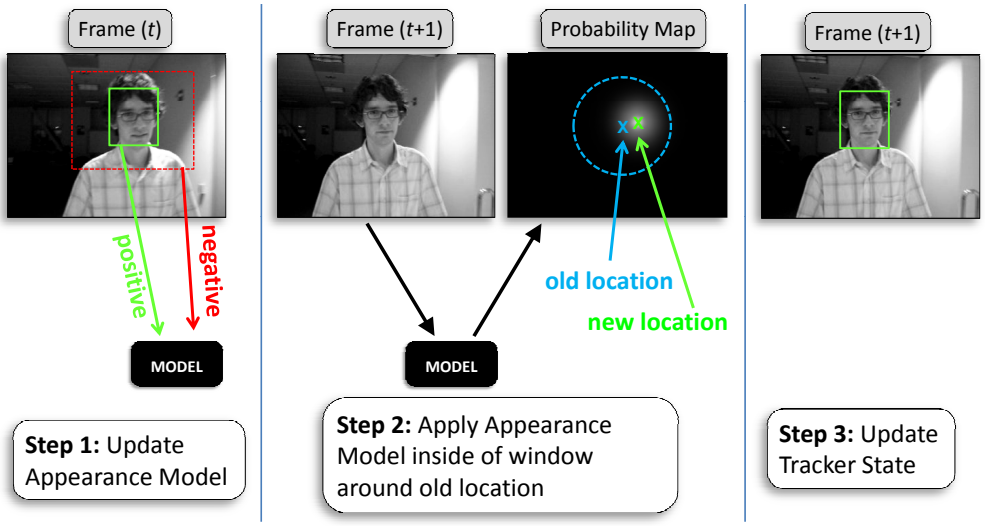


Figure 3.12: **Tracking by detection with a greedy motion model**. An illustration of how most tracking by detection systems work.

3.3.2 Tracking with Online MIL

In this section we introduce our tracking algorithm, MILTrack, which uses a MIL based appearance model. We begin with an overview of our tracking system which includes a description of the motion model we use. We then review online boosting [103, 61] and present a novel boosting based algorithm for online MIL (a combination of online boost and the MILBoost algorithm). Finally, we review various implementation details.

System Overview and Motion Model

The basic flow of the tracking system we implemented in this work is illustrated in Fig. 3.12 and summarized in Algorithm 3. Our image representation consists of a set of Haar-like features that are computed for each image patch [135, 44]; this is discussed in more detail in Section 3.3.2. The appearance model is composed of a discriminative classifier which is able to return $p(y = 1|x)$ (we will use $p(y|x)$ as shorthand), where x is an image patch (or the representation of an image patch in feature space) and y is a binary variable indicating the presence of the object of interest in that image patch. At every time step t, our tracker

Algorithm 3 MILtrack

Input: Video frame number k

1: Crop out a set of image patches, $X^s = \{x : ||\ell(x) - \ell^*_{t-1}|| < s\}$ and compute feature vectors.
2: Use MIL classifier to estimate $p(y = 1|x)$ for $x \in X^s$.
3: Update tracker location $\ell^*_t = \ell\Big(\text{argmax}_{x \in X^s}\, p(y|x)\Big)$.
4: Crop out two sets of image patches $X^r = \{x : ||\ell(x) - \ell^*_t|| < r\}$ and $X^{r,\beta} = \{x : r < ||\ell(x) - \ell^*_t|| < \beta\}$.
5: Update MIL appearance model with one positive bag X^r and $|X^{r,\beta}|$ negative bags, each containing a single image patch from the set $X^{r,\beta}$.

maintains the object location ℓ^*_t. Let $\ell(x)$ denote the location of image patch x (for now let's assume this consists of only the (x, y) coordinates of the patch center, and that scale is fixed; below we consider tracking scale as well). For each new frame we crop out a set of image patches $X^s = \{x : ||\ell(x) - \ell^*_{t-1}|| < s\}$ that are within some search radius s of the current tracker location, and compute $p(y|x)$ for all $x \in X^s$. We then use a greedy strategy to update the tracker location:

$$\ell^*_t = \ell\Big(\underset{x \in X^s}{\text{argmax}}\, p(y|x)\Big) \tag{3.4}$$

In other words, we do not maintain a distribution of the target's location at every frame, and our motion model is such that the location of the tracker at time t is equally likely to appear within a radius s of the tracker location at time $(t - 1)$:

$$p(\ell^*_t|\ell^*_{t-1}) \propto \begin{cases} 1 & \text{if } ||\ell^*_t - \ell^*_{t-1}|| < s \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

This could be extended with something more sophisticated, such as a particle filter, as is done in [72, 138, 110]; however, we again emphasize that our focus is on the appearance model.

Once the tracker location is updated, we proceed to update the appearance model. We crop out a set of patches $X^r = \{x : ||\ell(x) - \ell^*_t|| < r\}$, where $r < s$ is a scalar radius (measured in pixels), and label this bag positive (recall that in MIL we train the algorithm with labeled bags). In contrast, if a standard learning

algorithm were used, there would be two options: set $r = 1$ and use this as a single positive instance, or set $r > 1$ and label all these instances positive. For negatives we crop out patches from an annular region $X^{r,\beta} = \{x : r < ||\ell(x) - \ell_t^*|| < \beta\}$, where r is same as before, and β is another scalar. Since this generates a potentially large set, we then take a random subset of these image patches and label them negative. We place each negative example into its own negative bag, though placing them all into one negative bag yields the same result.

Incorporating scale tracking into this system is straightforward. First, we define an extra parameter λ to be the scale space step size. When searching for the location of the object in a new frame, we crop out image patches from the image at the current scale, ℓ_t^s, as well as one scale step larger and smaller, $\ell_t^s \pm \lambda$; once we find the location with the maximum response, we update the current state (both position and scale) accordingly. When updating the appearance model, we have the option of cropping training image patches only from the current scale, or from the neighboring scales as well; in our current implementation we do the former.

It is important to note that tracking in scale-space is a double-edged sword. In some ways the problem becomes more difficult because the parameter space becomes larger, and consequently there is more room for error. However, tracking this additional parameter may mean that the image patches we crop out are better aligned, making it easier for our classifier to learn the correct appearance. In our experiments we have noticed both behaviors – sometimes adding scale tracking helps, and other times it hurts performance.

Details on how all of the above parameters were set are in Section 3.3.3, although we use the *same* parameters throughout *all* the experiments. We continue with a more detailed review of MIL.

Online Boosting

Our algorithm for online MIL is based on the boosting framework [52] and is related to the work on Online AdaBoost [103] and its adaptation in [61]. The goal of boosting is to combine many weak classifiers $\tilde{h}(x)$ (usually decision stumps)

into an additive strong classifier:

$$H(x) = \sum_{k=1}^{K} \alpha_k \tilde{h}_k(x) \tag{3.6}$$

where α_k are scalar weights. There have been many boosting algorithms proposed to learn this model in batch mode [52, 54]; typically this is done in a greedy manner where the weak classifiers are trained sequentially. After each weak classifier is trained, the training examples are re-weighted such that examples that were previously misclassified receive more weight. If each weak classifier is a decision stump, then it chooses one feature that has the most discriminative power for the entire weighted training set. In this case boosting can be viewed as performing feature selection, choosing a total of K features, which is generally much smaller than the size of the entire feature pool. This has proven particularly useful in computer vision because it creates classifiers that are efficient at run time [135].

In [103], Oza develops an online variant of the popular AdaBoost algorithm [52], which minimizes the exponential loss function. This variant requires that all $\tilde{h}$ can be trained in an online manner. The basic flow of Oza's algorithm is as follows: for an incoming example x, each $\tilde{h}_k$ is updated sequentially and the weight of example x is adjusted after each update. Since the formulas for the example weights and classifier weights in AdaBoost depend only on the error of the weak classifiers, Oza proposes to keep a running average of the error of each $\tilde{h}_k$, which allows the algorithm to estimate both the example weight and the classifier weights in an online manner.

In Oza's framework if every $\tilde{h}$ is restricted to be a decision stump, the algorithm has no way of choosing the most discriminative feature because the entire training set is never available at one time. Therefore, the features for each $\tilde{h}_k$ must be picked a priori. This is a potential problem for computer vision applications, since they often rely on the feature selection property of boosting. Grabner et al. [61] proposed an extension of Oza's algorithm which performs feature selection by maintaining a pool of $M > K$ candidate weak stump classifiers h. When a new example is passed in, all of the candidate weak classifiers are updated in parallel. Then, the algorithm sequentially chooses K weak classifiers from this pool by

Algorithm 4 Online MILBoost (OMB)

Input: Dataset $\{X_i, y_i\}_{i=1}^N$, where $X_i = \{x_{i1}, x_{i2}, \ldots\}, y_i \in \{0, 1\}$

Update all M weak classifiers in the pool with data $\{x_{ij}, y_i\}$
Initialize $H_{ij} = 0$ for all i, j
for $k = 1$ to K **do**
  for $m = 1$ to M **do**
    $p_{ij}^m = \sigma\big(H_{ij} + h_m(x_{ij})\big)$
    $p_i^m = 1 - \prod_j \big(1 - p_{ij}^m\big)$
    $\mathcal{L}^m = \sum_i \Big(y_i \log(p_i^m) + (1 - y_i) \log(1 - p_i^m)\Big)$
  end for
  $m^* = \text{argmax}_m \, \mathcal{L}^m$
  $\tilde{h}_k(x) \leftarrow h_{m^*}(x)$
  $H_{ij} = H_{ij} + \tilde{h}_k(x)$
end for

Output: Classifier $H(x) = \sum_k \tilde{h}_k(x)$, where $p(y|x) = \sigma\big(H(x)\big)$

keeping running averages of errors for each, as in [103], and updates the weights of $\tilde{h}$ accordingly. We employ a similar feature selection technique in our Online MIL algorithm, although the criteria for choosing weak classifiers is different.

Online Multiple Instance Boosting

The algorithms in [103] and [61] rely on the special properties of the exponential loss function of AdaBoost, and therefore cannot be readily adapted to the MIL problem. We now present our novel online boosting algorithm for MIL. As before, we take a statistical view of boosting, where the algorithm is trying to optimize a specific objective function J. In this view, the weak classifiers are chosen sequentially to optimize the following criteria:

$$(\tilde{h}_k, \alpha_k) = \underset{\tilde{h} \in \mathcal{H}, \alpha}{\text{argmax}} \, J(H_{k-1} + \alpha \tilde{h}) \tag{3.7}$$

where H_{k-1} is the strong classifier made up of the first $(k-1)$ weak classifiers,

and $\mathcal{H}$ is the set of all possible weak classifiers. In batch boosting algorithms, the objective function J is computed over the entire training data set. In our case, for the current video frame we are given a training data set $\{(X_1, y_1), (X_2, y_2) \ldots\}$, where $X_i = \{x_{i1}, x_{i2} \ldots\}$. We would like to update our classifier to maximize log likelihood of this data (Equation 2.6). We model the instance and bag probabilities in the same manner as the MILBoost algorithm (refer to Chapter 2). To simplify the problem, we absorb the scalar weights α_t into the weak classifiers, by allowing them to return real values rather than binary.

In the same vein as [61], at all times our algorithm maintains a pool of $M > K$ candidate weak stump classifiers h. To update the classifier, we first update all weak classifiers in parallel, similar to [61]. Note that although instances are in bags, the weak classifiers in a MIL algorithm are instance classifiers, and therefore require instance labels y_{ij}. Since these are unavailable, we pass in the bag label y_i for all instances x_{ij} to the weak training procedure. We then choose K weak classifiers $\tilde{h}$ from the candidate pool sequentially, by maximizing the log likelihood of bags:

$$\tilde{h}_k = \underset{h \in \{h_1, \ldots, h_M\}}{\operatorname{argmax}} \mathcal{L}(H_{k-1} + h) \tag{3.8}$$

See Algorithm 4 for the pseudo-code of Online MILBoost.

Discussion

There are a couple important issues to point out about this algorithm. First, we acknowledge the fact that training the weak classifiers with positive labels for all instances in the positive bags is sub-optimal because some of the instances in the positive bags may actually not be "correct". The algorithm makes up for this when it is choosing the weak classifiers $\tilde{h}$ based on the *bag* likelihood loss function. We have also experimented using online GradientBoost [79] to compute weights (via the gradient of the loss function) for all instances, but found this to make little difference in accuracy while making the system slower. Second, if we compare Equations 3.7 and 3.8 we see that the latter has a much more restricted choice of weak classifiers. This approximation does not seem to degrade the performance of

the classifier in practice, as noted in [60]. Finally, we note that the likelihood being optimized in Equation 3.8 is computed only on the current examples. Thus, it has the potential of overfitting to current examples, and not retaining information about previously seen data. This is averted by using online weak classifiers that do retain information about previously seen data, which balances out the overall algorithm between fitting the current data and retaining history.

Implementation Details

Weak Classifiers

Recall that we require weak classifiers h that can be updated online. In our system each weak classifier h_k is composed of a Haar-like feature f_k and four parameters $(\mu_1, \sigma_1, \mu_0, \sigma_0)$ that are estimated online. The classifiers return the log odds ratio:

$$h_k(x) = \log \left[\frac{p_t\big(y = 1 | f_k(x)\big)}{p_t\big(y = 0 | f_k(x)\big)} \right] \tag{3.9}$$

where $p_t\big(f_t(x)|y = 1\big) \sim \mathcal{N}(\mu_1, \sigma_1)$ and similarly for $y = 0$. We let $p(y = 1) = p(y = 0)$ and use Bayes rule to compute the above equation. When the weak classifier receives new data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ we use the following update rules:

$$\begin{aligned} \mu_1 &\leftarrow \gamma\mu_1 + (1-\gamma)\frac{1}{n}\sum_{i|y_i=1} f_k(x_i) \\ \sigma_1 &\leftarrow \gamma\sigma_1 + (1-\gamma)\sqrt{\frac{1}{n}\sum_{i|y_i=1}\big(f_k(x_i) - \mu_1\big)^2} \end{aligned}$$

where $0 < \gamma < 1$ is a learning rate parameter. The update rules for μ_0 and σ_0 are similarly defined.

Image Features

We represent each image patch as a vector of Haar-like features [135], which are randomly generated, similar to [44]. Each feature consists of 2 to 4 rectangles, and each rectangle has a real valued weight. The feature value is then a weighted sum of the pixels in all the rectangles. These features can be computed efficiently using the integral image trick described in [135].

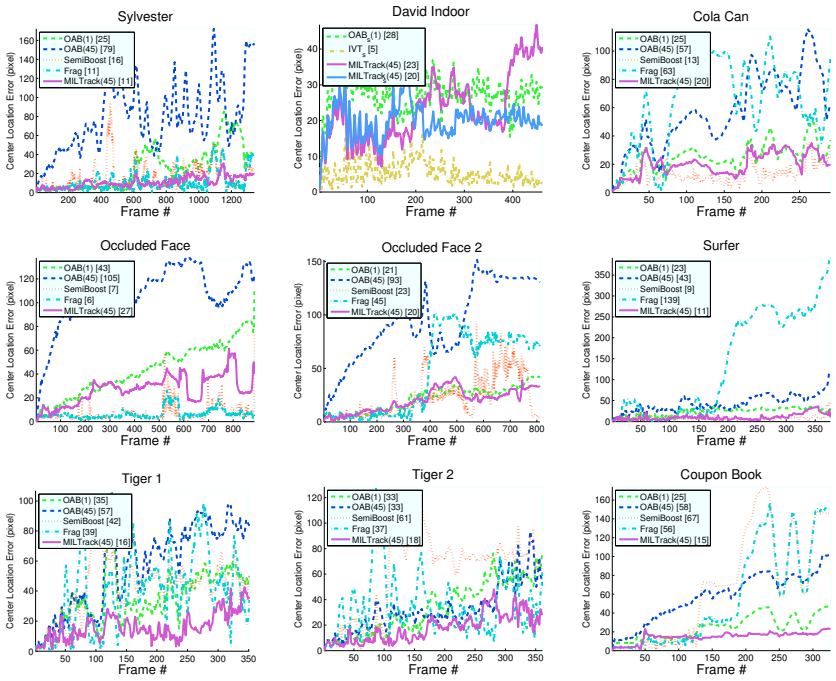


Figure 3.13: **Tracking Object Location**. Location Error Plots. See text for details.

3.3.3 Experiments

We tested our MILTrack system on several challenging video sequences, some of which are publicly available. For comparison, we implemented a tracker based on the Online AdaBoost (OAB) algorithm described in [61]. We plugged this learning algorithm into our system, and used the same features and motion model as for MILTrack (See Section 3.3.2). We acknowledge the fact that our implementation of the OAB tracker achieves worse performance than is reported in [61]; this could be because we are using simpler features, or because our parameters were not tuned per video sequence. However, our study is still valid for comparison because only the learning algorithm changes between our implementation of the OAB tracker and MILTrack, and everything else is kept constant. This allows us to isolate the appearance model to make sure that it is the cause of the performance difference.

One of the goals of this work is to demonstrate that using MIL results in

Table 3.3: **Tracking Object Location**. Average center location errors (pixels). Bold green font indicates best performance, red italics font indicates second best.

Video Clip	OAB(1)	OAB(45)	SemiBoost	Frag	MILTrack(45)
Sylvester	25	79	16	**11**	**11**
David Indoor	49	72	*39*	46	**23**
Coke Can	25	57	**13**	63	*20*
Occluded Face	43	105	*7*	**6**	27
Occluded Face 2	*21*	93	23	45	**20**
Surfer	23	43	**9**	139	*11*
Tiger 1	*35*	57	42	39	**16**
Tiger 2	*33*	*33*	61	37	**18**
Coupon Book	*25*	58	67	56	**15**

a more robust and stable tracker. For this reason ***all algorithm parameters were fixed for all the experiments.*** This holds for all algorithms we tested. For MILTrack and OAB the parameters were set as follows. The search radius s is set to 35 pixels. For MILTrack we sample positives in each frame using a radius $r = 4$ (we found that the algorithm is fairly robust for a range of values). This generates a total of 45 image patches comprising one positive bag (for clarity, we call this MILTrack(45)). For the OAB tracker we tried two variations. In the first variation we set $r = 1$ generating only one positive example per frame (we call this OAB(1)); in the second variation we set $r = 4$ as we do in MILTrack (although in this case each of the 45 image patches is labeled positive); we call this OAB(45). The reason we experimented with these two versions was to show that the superior performance of MILTrack is not simply due to the fact that we extract multiple positive examples per frame. In fact, as we will see shortly, when multiple positive examples are used for the OAB tracker, its performance degrades[3] (cf. Table 3.3 and Fig. 3.13). The scalar β for sampling negative examples was set to 50, and we randomly sample 65 negative image patches from the set $X^{r,\beta}$ (though during

[3]We also experimented with the LogitBoost loss function (as in [79], which penalizes noisy examples less harshly, and although it worked better than OAB, it did not outperform MILTrack. We omit the detailed results due to space constraints.

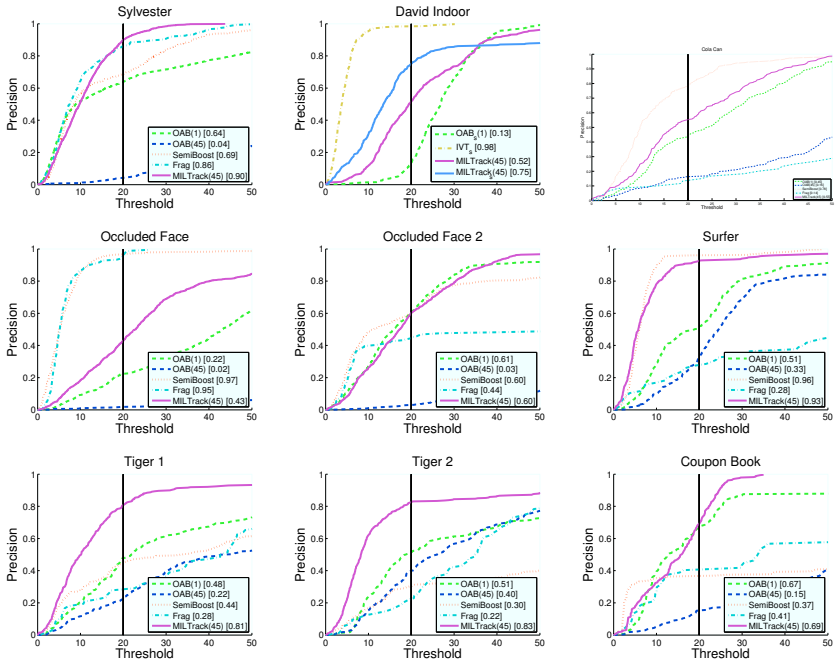


Figure 3.14: **Tracking Object Location**. Precision plots. See text for details.

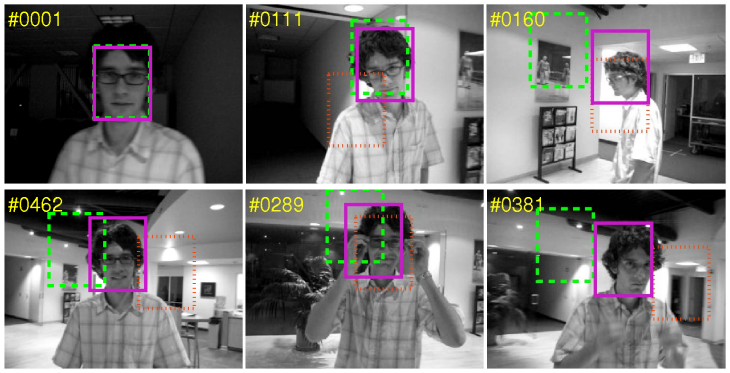
initialization with the first frame we sample 1000 patches). The learning rate γ for the weak classifiers is set to 0.85. Finally, the number of candidate weak classifiers M was set to 250, and the number of chosen weak classifiers K was set to 50.

To gauge absolute performance we also compare our results to three other algorithms, using code provided by the respective authors. The first of these is the SemiBoost tracker [62][4]; as mentioned earlier, this method uses label information from the first frame only, and then updates the appearance model via online semi-supervised learning in subsequent frames. This makes it particularly robust to scenarios where the object leaves the scene completely. However, the model relies strongly on the prior classifier (trained using the first frame). We found that on clips exhibiting significant appearance changes this algorithm often lost the object. The second algorithm is FragTrack [1][5]. This algorithm uses a static appearance model based on integral histograms, which have been shown to be very efficient. The appearance model is part based, which makes it robust to occlusions. For

[4]Code available at `http://www.vision.ee.ethz.ch/boostingTrackers/download.htm`.
[5]Code available at `http://www.cs.technion.ac.il/~amita/fragtrack/fragtrack.htm`.

(A) Sylvester

(B) David Indoor



(C) Coke Can

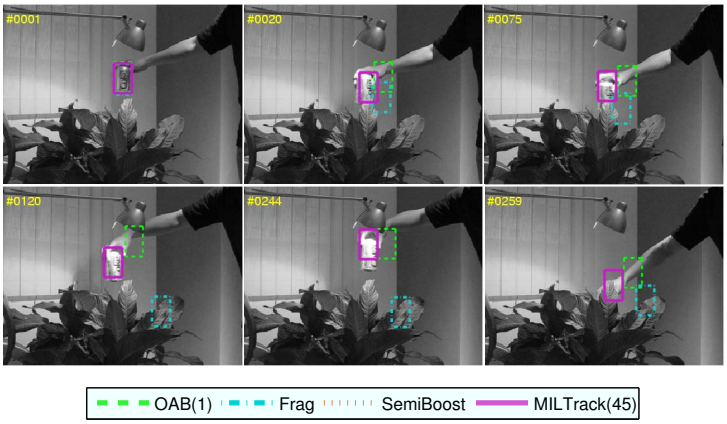
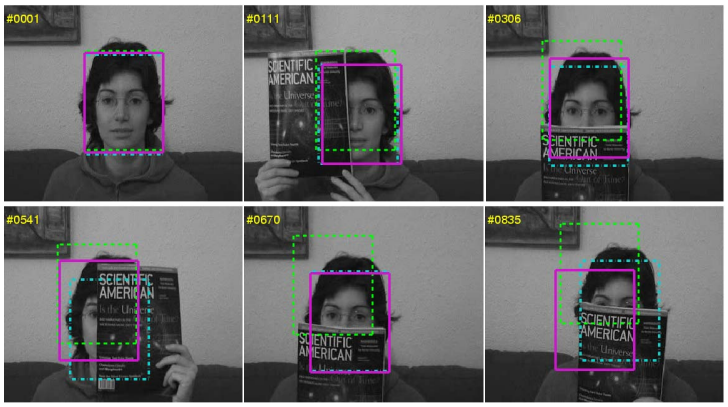
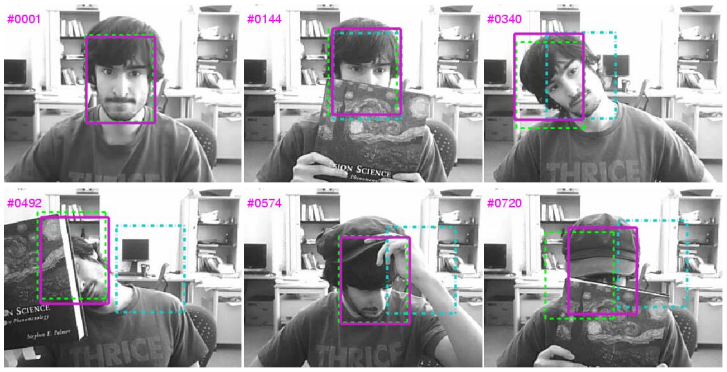


Figure 3.15: **Tracking Object Location**. Screenshots of tracking results, highlighting instances of out-of-plane rotation, occluding clutter, scale and illumination change. For the sake of clarity we only show three trackers per video clip.

(A) Occluded Face



(B) Occluded Face 2



(C) Surfer

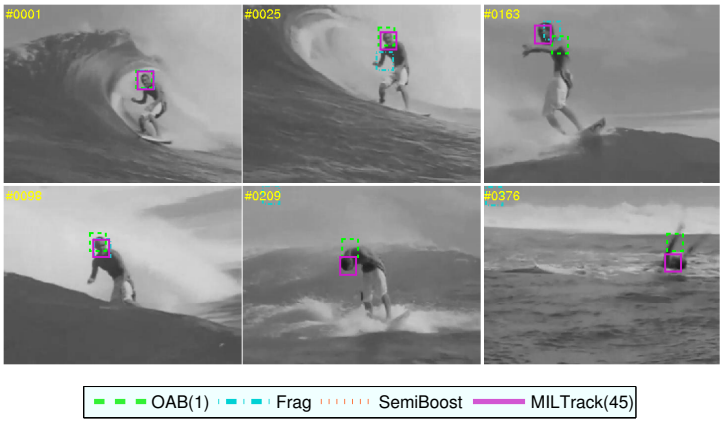


Figure 3.16: **Tracking Object Location**. Screenshots of tracking results, highlighting instances of out-of-plane rotation, occluding clutter, scale and illumination change. For the sake of clarity we only show three trackers per video clip.

Table 3.4: **Tracking Object Location**. Precision at a fixed threshold of 20. Bold green font indicates best performance, red italics font indicates second best.

Video Clip	OAB(1)	OAB(45)	SemiBoost	Frag	MILTrack(45)
Sylvester	0.64	0.04	0.69	*0.86*	**0.90**
David Indoor	0.16	0.08	*0.46*	0.45	**0.52**
Coke Can	0.45	0.16	**0.78**	0.14	*0.55*
Occluded Face	0.22	0.02	**0.97**	*0.95*	0.43
Occluded Face 2	**0.61**	0.03	*0.60*	0.44	*0.60*
Surfer	0.51	0.33	**0.96**	0.28	*0.93*
Tiger 1	*0.48*	0.22	0.44	0.28	**0.81**
Tiger 2	*0.51*	0.40	0.30	0.22	**0.83**
Coupon Book	*0.67*	0.15	0.37	0.41	**0.69**

both algorithms, we use the default parameters provided by the authors for all of our experiments. For experiments where we track both location and scale we compare to IVT [110], setting the parameters such that only location and scale are tracked (rather than a full set of affine parameters). For the trackers than involve randomness, all results are averaged over 5 runs.

The system was implemented in C++ (code and data available on our project website[6]), and runs at about 25 frames per second (FPS).

Evaluation Methodology

Evaluating a tracking algorithm is itself a challenge. Qualitative comparison on test video clips is most common; quantitative comparison typically involves plotting the center location error versus frame number. Since these plots can be difficult to interpret, it is useful to summarize performance by computing the mean error over all the frames of the video. However, this value sometimes fails to correctly capture tracker performance. For example, if a tracker tracks an object closely for most of the video, but loses track completely on the last several frames, the mean location error may be higher than a tracker that sticks with the object,

[6] `http://vision.ucsd.edu/project/tracking-online-multiple-instance-learning`

Table 3.5: **Tracking Object Location & Scale**. Location mean error. Bold green font indicates best performance, red italics font indicates second best.

Video Clip	$OAB_s(1)$	IVT_s	MILTrack(45)	$MILTrack_s(45)$
David Indoor	28	**5**	23	*20*
Cliff Bar	18	30	*12*	**9**
Twinings	17	14	**10**	**10**

Table 3.6: **Tracking Object Location & Scale**. Precision at a fixed threshold of 20. Bold green font indicates best performance, red italics font indicates second best.

Video Clip	$OAB_s(1)$	IVT_s	MILTrack(45)	$MILTrack_s(45)$
David Indoor	0.13	**0.98**	0.52	*0.75*
Cliff Bar	0.76	0.57	*0.90*	**0.98**
Twinings	0.74	0.70	**0.91**	*0.87*

though not as precisely. The preference between these two behaviors inevitably depends on the final application.

For the above reasons, in addition to presenting screen shots and location error analysis, we include precision plots, similar to the analysis in [31], and suggested in [145]. These plots show the percentage of frames for which the estimated object location was within some threshold distance of the ground truth. To summarize these plots, we chose the threshold 20 and report the precision at this point in the curve (e.g. this is the percent of frames for which the tracker was less than 20 pixels off from the ground truth); this threshold roughly corresponds to at least a 50% overlap between the tracker bounding box and the ground truth. Note that we could have used the PASCAL [47] overlap criteria throughout our evaluation; however, this would require us to label full bounding boxes (which is more time consuming), and would make it difficult to compare trackers that do and do not return estimated scale. Finally, note that when multiple trails were done, we computed error for each trial and averaged the errors rather than averaging the tracker outputs and computing error.

Tracking Object Location

We perform our experiments on 3 publicly available video sequences, as well as 6 of our own. For all sequences we labeled the ground truth center of the object for every 5 frames, and interpolated the location in the other frames (with the exception of the "Occluded Face" sequence, for which the authors of [1] provided ground truth). All video frames were converted to gray scale prior to processing.

The quantitative results are summarized in Tables 3.3 and 3.4, and plots are shown in Fig. 3.13 and 3.14; Fig. 3.15,3.16 and 3.17 show screen captures for some of the clips. Below is a more detailed discussion of the video sequences.

Sylvester & David Indoor

These two video sequences have been used in several recent tracking papers [110, 82, 61], and they present challenging lighting, scale and pose changes. Our algorithm achieves the best performance (tying FragTrack on the "Sylvester" sequence).

Occluded Face, Occluded Face 2

In the "Occluded Face" sequence, which comes from the authors of [1], FragTrack performs the best because it is specifically designed to handle occlusions via a part-based model. However, on our similar, but more challenging clip, "Occluded Face 2", FragTrack performs poorly because it cannot handle appearance changes well (e.g. when the subject puts a hat on, or turns his face). This highlights the advantages of using an adaptive appearance model.

CokeCan, Surfer The Coke Can sequence contains a specular object, which adds some difficulty. The "Surfer" clip was downloaded from Youtube; this clip would be easier to track if color information were used[7], but since we use grayscale images for all experiments this clip is fairly challenging. Both MILTrack and the SemiBoost tracker perform well on these clips (cf. Fig. 3.14).

Tiger 1, Tiger 2

These sequences exhibit many challenges, and contain frequent occlusions and fast motion (which causes motion blur). The two sequences show the toy tiger in many different poses, and include out of plane rotations (cf. Fig. 3.17 (A)). Our

[7]It would be straightforward to extend our system to use color – e.g. compute Haar features over color channels.

algorithm outperforms the others, often by a large margin.

Coupon Book

This clip illustrates a problem that arises when the tracker relies too heavily on the first frame. The appearance of the coupon book is changed after about 50 frames, by folding one of its pages; then an "imposter" coupon book is introduced to distract the trackers. MILTrack successfully tracks the correct coupon book, while FragTrack and the SemiBoost tracker are confused by the impostor object.

Tracking Object Location & Scale

Here we present results for both location and scale tracking. Scale tracking is independent of the appearance model, so our implementation of scale tracking for MILTrack is easily carried over to the OAB tracker. Note that the quantitative results we present are still based on object center location only; we do not measure error of scale estimation. This allows us to compare results of trackers that estimate scale and those with a fixed scale. Furthermore, gathering ground truth for object center is less time consuming than for a full bounding box.

David Indoor

This is the same clip that we studied in the previous section. Here we see a big advantage of using scale tracking – MILTrack with scale performs better than MILTrack without scale, and it performs better than OAB(1) with scale. However, the IVT tracker achieves the best result on this video clip. We believe IVT is particularly well suited to faces since it uses a subspace (PCA) appearance model. We will see in the next experiments that IVT does not work well in other scenarios.

Cliffbar In this clip the goal is to track an object that changes in scale and moves against a background that is very similar in texture. We see that the IVT tracker fails in this case, when the object is turned upside down. The IVT tracker uses a generative model, rather than discriminative, so it does not take into account the negative examples from the image. Because the background is so similar to the object of interest in this video clip, IVT ultimately loses the object and snaps to some part of the background. As before, we see that MILTrack with scale performs better than MILTrack without scale and OAB(1) with scale; overall

MILTrack achieves the best performance on this clip.

Twinings

This clip again shows the shortcomings of IVT – the clip shows a box of tea which is moved around and rotated (exposing new faces of the box). IVT fails when these out of plane rotations take place (see Fig. 3.18(C), frame #240 and beyond). Though the center location error is similar for both version of MILTrack (Fig. 3.19), we can see the version that includes scale search results in more satisfactory results (e.g. frame #134).

3.3.4 Discussion

There are still some interesting unanswered questions about adaptive appearance models. Although our method results in more robust tracking, it cannot completely avoid the types of problems that adaptive appearance trackers suffer from. In particular, if an object is completely occluded for a long period of time, or if the object leaves the scene completely, any tracker with an adaptive appearance model will inevitably start learning from incorrect examples and lose track of the object. Some interesting work exploring ways to deal with this issue has been presented in [62] and more recently in [122]. These methods attempt to combine a pre-trained object detector with an adaptively trained tracker. One interesting avenue for future work would be to combine these ideas with the ones presented in this paper. Another challenge is to track articulated objects which cannot be easily delineated with a bounding box. These types of objects may require a part-based approach, such as the recent methods in object detection [49, 42].

Finally, online algorithms for Multiple Instance Learning could be useful in areas outside of visual tracking. Work on better algorithms and theoretical analysis relating offline/batch MIL and online MIL is already under way (e.g. [96]), and we suspect more is to come.

3.4 Conclusions

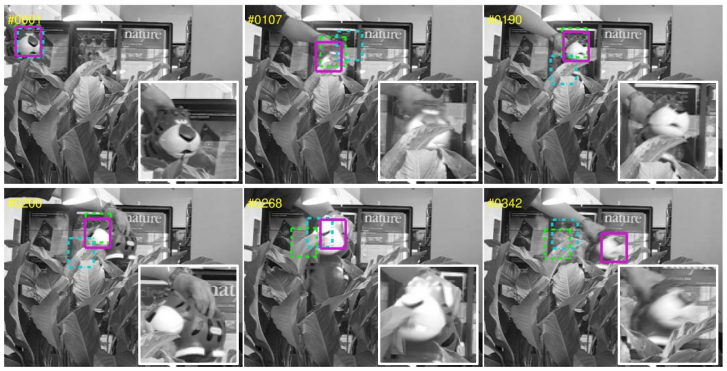
Portions of this chapter are based on the following publications:

- "Multiple Component Learning for Object Detection" by P. Dollár, B. Babenko, S. Belongie, P. Perona, and Z. Tu [42]. The dissertation author contributed to algorithm development, implemented parts of the code and experiments, and contributed to writing of the paper.

- "Weakly supervised object recognition and localization with stable segmentations" by C. Galleguillos, B. Babenko, A. Rabinovich, and S. Belongie [57]. The dissertation author implemented parts of the system, and contributed to writing of the paper.

- "Visual Tracking with Online Multiple Instance Learning" [15] and "Robust Object Tracking with Online Multiple Instance Learning" [16] by B. Babenko, M. -H. Yang, and S. Belongie. The dissertation author developed the algorithm and experiments, and wrote most of the paper.

(A) Tiger 2



(B) Coupon Book

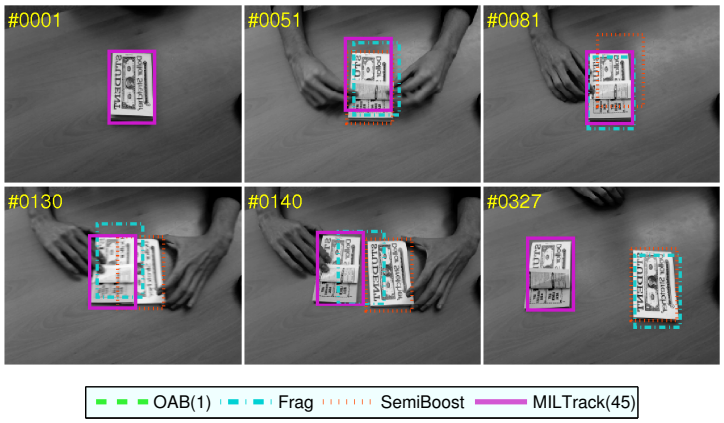
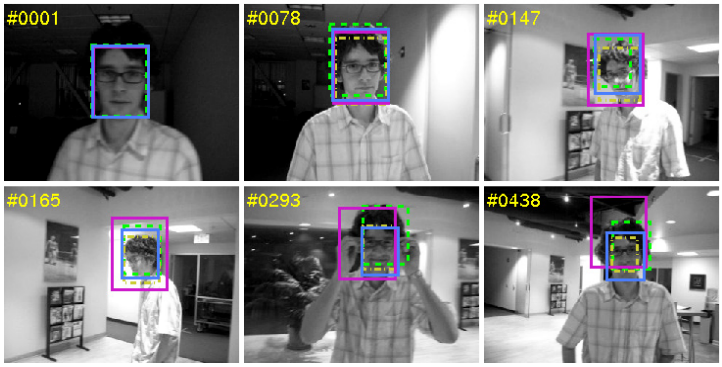
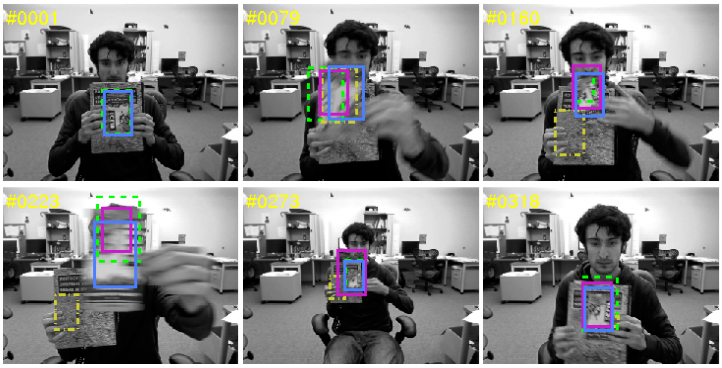


Figure 3.17: **Tracking Object Location**. Screenshots of tracking results, highlighting instances of out-of-plane rotation, occluding clutter, scale and illumination change. For the Tiger 2 clip we also include close up shots of the object to highlight the wide range of appearance changes. For the sake of clarity we only show three trackers per video clip.

(A) David Indoor



(B) Cliffbar



(C) Twinings

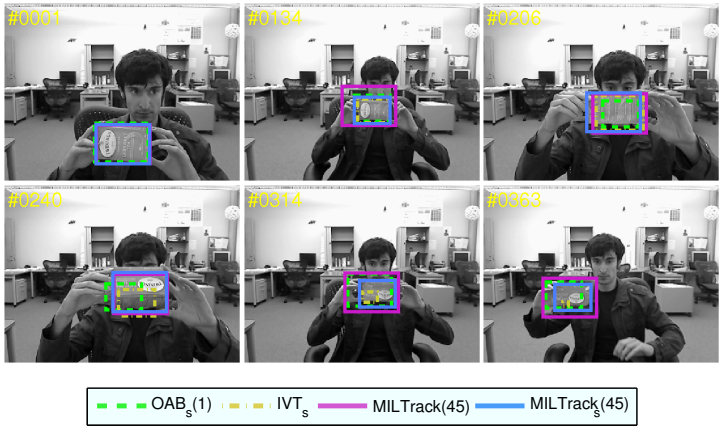


Figure 3.18: **Tracking Object Location & Scale**. Screenshots showing results for tracking both location and scale of objects. Note that the localization is much more precise when scale is one of the tracked parameters.

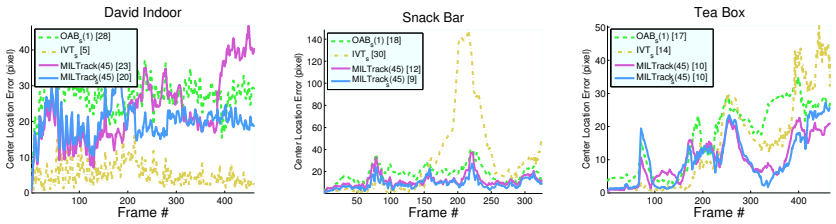


Figure 3.19: **Tracking Object Location & Scale**. Average center location errors. See text for details.

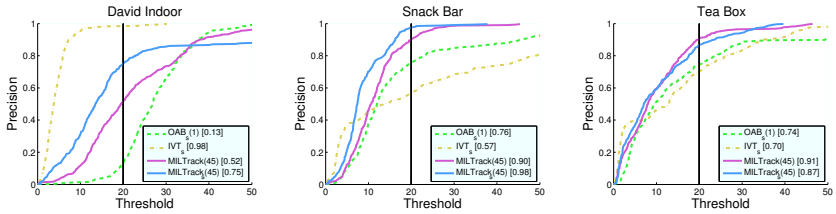


Figure 3.20: **Tracking Object Location & Scale**. Precisions plots. See text for details.

4 Weakly Labeled Categories

Assigning examples to categories is a surprisingly ambiguous task. Suppose we have an image of an apple – should this image be labeled as "Granny Smith", "apple" or "fruit"? It's easy to see that such categories are actually organized in a hierarchy and the correct answer to this question depends on the application. Regardless of what we desire as the output of our system, it may be in our interest to train the system with categories that are higher or lower in the hierarchy than what is provided during training. For example, the category "fruit" is extremely heterogeneous and can be difficult to learn with existing machine vision methods; it may in our interest to break this category into many smaller or fine-grained categories (i.e. sub-categories) for purposes of training. On the other hand, we may want to automatically discover higher level categories (i.e. super-categories) in the interest of a more efficient hierarchy-based recognition system. In this chapter we will explore applications where we would like to automatically discover sub or super-categories, treating these as latent variables in the data.

4.1 Object Detection with Sub-categories

In object detection, data alignment is often necessary to achieve good classification results. E.g., consider the Viola-Jones face detector [135]. The detector works best when trained with images that come from a single coherent group (e.g., frontal faces) and lie in approximate correspondence (e.g., eyes aligned). Generalizing Viola-Jones to multiple poses and offset training data remains an active area of research.

In this work we are interested in strategies for simultaneous learning and

Figure 4.1: **Object detection with Multiple Pose Learning**.

alignment. We have already discussed that data must be aligned such that training samples lie in correspondence and how this can be achieved automatically using Multiple Instance Learning. While this strategy is appropriate for dealing with translational offset, it is not appropriate for other cases like out of plane rotation of objects. in the latter case a more effective approach is to split the positive examples into sub-categories, each of which would contain well-aligned data. In this Section we will present an algorithm for simultaneously splitting data into sub-categories and training classifiers for each sub-category. Coincidentally, the algorithm is very similar to MILBoost and we will discuss this relationship.

4.1.1 Related Work

A number of papers have focused on grouping data either manually or automatically. E.g., one approach for face detection is to train multiple detectors, each on a manually defined range of orientations [69, 65]. A pre-processing step can be used to predict orientation [69]. However, manual grouping of data can be labor intensive and suboptimal. Alternatively, data can be clustered prior to

training [117], but the resulting clustering is likely suboptimal. Most related to this are methods that attempt to group the data during training [126, 127, 141], e.g. by learning a tree classifier that naturally splits the data [127, 141]. Although promising, these approaches often have heuristic splitting procedures and many parameters.

4.1.2 Multiple Pose Learning

We now present a boosting approach for simultaneously splitting data into groups and training a classifier for each group. Similar to standard supervised learning, we are given n samples $x_i \in \mathcal{X}$ and n corresponding labels $y_i \in \{-1, +1\}$. We assume, however, that there are K latent variables $y_i^k \in \{-1, +1\}$ associated with each sample. Each latent variable defines membership to one of the K groups. A sample is considered positive if it belongs to at least one of these groups, which can be expressed as follows:

$$y_i = \max_k(y_i^k) \tag{4.1}$$

Our goal is to simultaneously split the positive data into K groups and train K classifiers $H^1, \ldots, H^K$, one per group, so that $\max_k(H^k(x_i)) = y_i$. We call this problem *Multiple Pose Learning*. Note that the latent variables y_i^k are not known; if they were, we could train each H^k using standard learning approaches.

As before, we would like to optimize the negative log likelihood $\mathcal{L}$. However, we need to modify our definition of the probability of x_i. We define the probability $p_i^k \equiv p^k(y_i = 1|x_i)$ according to a single classifier H^k as $p_i^k = \sigma(2H_i^k)$, where $H_i^k = H^k(x_i)$. This is similar to Eqn. (2.7). Given all of the classifiers, we define the probability $p_i \equiv p(y_i = 1|x_i)$ as the *maximum* over the probabilities p_i^k. Using one of the approximations g of the max from Section 2.3, we write:

$$p_i = g_k(p_i^k) = g_k(\sigma(2H_i^k)) \tag{4.2}$$

As for MILBoost, this can be viewed as the probabilistic approximation of Eqn. (4.1). Note that p_i and consequently $\mathcal{L}$ depend on each H_i^k; to make this dependence explicit we write $\mathcal{L}(H^1, \cdots, H^K)$. We optimize $\mathcal{L}(H^1, \cdots, H^K)$ by coordinate descent, cycling through k, where in each phase we add a weak classifier to

Algorithm 5 MPLBoost

Input: Dataset $\{x_1, \ldots, x_n\}$, $\{y_1, \ldots, y_n\}$,$y_i \in \{-1, 1\}$, K

for $t = 1$ to T **do**
  for $k = 1$ to K **do**
    Compute weights $w_i^k = -\frac{\partial \mathcal{L}}{\partial H_i^k}$
    Train weak classifier h_t^k using weights $|w_i^k|$
      $h_t^k = \text{argmin}_h \sum_i \mathbf{1}(h(x_i) \neq y_i)|w_i^k|$
    Find α_t via line search to minimize $\mathcal{L}(\cdot, H^k, \cdot)$
      $\alpha_t = \text{argmin}_\alpha \mathcal{L}(\cdot, H^k + \alpha h_t^k, \cdot)$
    Update strong classifier $H^k \leftarrow H^k + \alpha_t^k h_t^k$.
  end for
end for

H^k while keeping all other weak classifiers fixed. The algorithm is summarized in Algorithm 5.

All that remains is to derive $w_i^k = -\frac{\partial \mathcal{L}}{\partial H_i^k}$. Using the chain rule we get:

$$\frac{\partial \mathcal{L}}{\partial H_i^k} = \frac{\partial \mathcal{L}}{\partial p_i} \frac{\partial p_i}{\partial p_i^k} \frac{\partial p_i^k}{\partial H_i^k}. \tag{4.3}$$

All other terms cancel. Plugging in $\frac{\partial \mathcal{L}}{\partial p_i}$ and $\frac{\partial p_i^k}{\partial H_i^k}$, derived in Eqns. (2.8-2.9), we get:

$$w_i^k = \begin{cases} \frac{2p_i^k(1-p_i^k)}{p_i} \frac{\partial p_i}{\partial p_i^k} & \text{if } y_i = 1 \\ \frac{-2p_i^k(1-p_i^k)}{1-p_i} \frac{\partial p_i}{\partial p_i^k} & \text{if } y_i = -1 \end{cases} \tag{4.4}$$

The form of $\frac{\partial p_i}{\partial p_i^k}$ depends on the choice of max approximator g, see Table 2.1. We list the equations for w_i^k for different choices of g in Table 4.1. Notice that the algorithm, and in particular the equations of the weights, are nearly identical to those of MILBoost (see Table 2.2).

4.1.3 Experiments

We used two datasets: LFW [66] and MNIST [76]. MNIST is a simple optical digit recognition dataset; we use it to illustrate certain properties of the

Table 4.1: MPL equations for w_i^k for different choices of g.

	$y_i = -1$	$y_i = 1$
LSE	$\frac{-2p_i^k(1-p_i^k)}{1-p_i} \frac{\exp(rp_i^k)}{\sum_\ell \exp(rp_i^\ell)}$	$\frac{2p_i^k(1-p_i^k)}{p_i} \frac{\exp(rp_i^k)}{\sum_\ell \exp(rp_i^\ell)}$
GM	$\frac{-2p_i}{1-p_i}\left(\frac{(p_i^k)^r-(p_i^k)^{r+1}}{\sum_\ell (p_i^\ell)^r}\right)$	$2\left(\frac{(p_i^k)^r-(p_i^k)^{r+1}}{\sum_\ell (p_i^\ell)^r}\right)$
NOR	$-2p_i^k$	$\frac{2p_i^k(1-p_i)}{p_i}$
ISR	$\frac{-2\chi_i^k p_i}{\sum_\ell \chi_i^\ell}, \chi_i^k = \frac{p_i^k}{1-p_i^k}$	$\frac{2\chi_i^k(1-p_i)}{\sum_\ell \chi_i^\ell}, \chi_i^k = \frac{p_i^k}{1-p_i^k}$

algorithms presented. The LFW dataset contains images of faces in the media. Although primarily intended for evaluating face recognition, we use it within a detection setting. For 'hard' non-face images we used false-positives returned by a Viola-Jones face detector. In all experiments we use random Haar features [44] as the weak classifiers.

The goal of MPL is to simultaneously group data and train a classifier for each group. Therefore a natural application of MPL is to train detectors for object categories that have several distinct views. E.g., out of plane rotations make it difficult to place faces into correspondence, which can make it difficult to train a precise face detector. On the other hand, if we can accurately group data during training, we not only achieve higher accuracy but also potentially more meaningful output (e.g. a face detector that also predicts facial pose).

LFW

We took 2000 LFW faces and synthesized three distinct groups: (1) faces kept as is, (2) faces rescaled by a factor of 2.5, and (2) faces rescaled by 2 and rotated 90°. In each case we use boosting to select $T = 64$ weak classifiers from 1000 candidates. We randomly split 4000 images into equal sized training and testing sets, and average all experiments over 10 trials. Figure 4.2(right) shows ROC curves comparing AdaBoost with four versions of MPLBoost. In Fig. 4.3 we show randomly selected test images grouped according to the learned classifiers. We see that MPLBoost recovers the groups reasonably well. Since we have the ground truth memberships y_i^k, we can quantify the quality of the grouping (up to

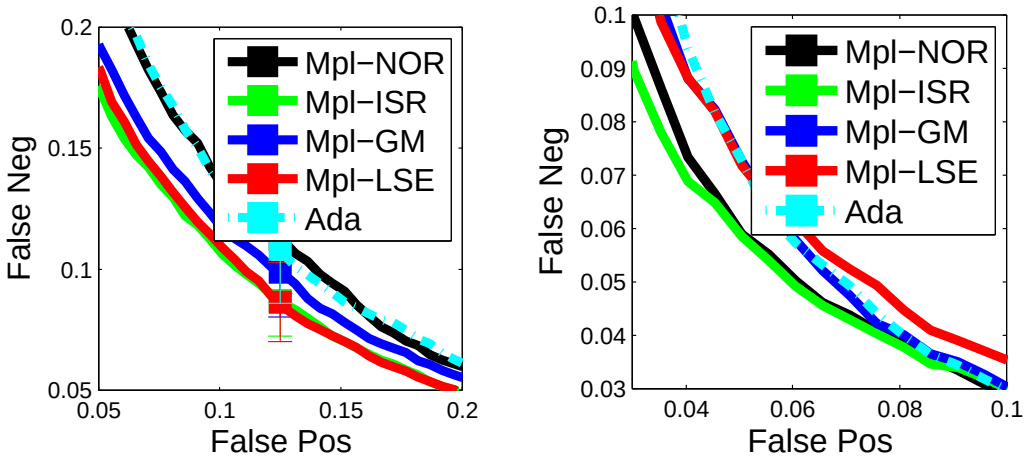


Figure 4.2: **MPL Results**. ROC plots comparing four variants of MPLBoost and standard learning (AdaBoost) with: *left:* MNIST; *right:* LFW.

a permutation of the group ids). For the faces, the grouping accuracy for the four softmax models were: NOR 0.81%, ISR 0.84%, GM 0.83%, LSE 0.88%.

MNIST

Our final experiment is similar to the one described above. We chose the digits 0-2 as positives and the others as negatives. The setup varied only slightly (1500 positives, 1500 negatives, 40% used for training, averaged over 10 trials). ROC curves are shown in Fig. 4.2 (left). We computed membership accuracy by assuming each positive digit forms its own group: NOR 0.83%, ISR 0.89%, GM 0.87%, LSE 0.87%.

4.2 Object Recognition with Super-categories

Classification methods that are typically used in object recognition require some notion of similarity over the inputs (e.g. a distance metric for nearest neighbor, a kernel for SVM). Therefore, a major focus in object recognition has been placed on developing powerful image representations that capture the necessary similarity information. For example, it is well known that combining features of

Figure 4.3: **MPL Alignment Results**. Randomly selected test images grouped according to the 3 classifiers trained with MPL. *Top:* MNIST; *Bottom:* LFW.

many different cues significantly improves recognition performance [78, 131]; the problem can therefore be reduced to finding the proper combination or weighting of these cues. This can be done by designing good features by hand, or by starting with many simple features (e.g. pixel values, or haar wavelets) and using training data to learn a good similarity metric (or equivalently an embedding). The latter goes by many different names and appears in various subareas of machine learning and computer vision: metric learning, cue combination/weighting, kernel combination/learning, feature selection, etc.. Though the specifics of each subarea may differ, the basic idea boils down to finding a powerful and discriminative image representation.

For the purpose of categorization, two approaches have thus far been explored: learning a global or "monolithic" similarity metric, and learning a similarity metric per category. The former problem is known as metric learning in the machine learning community. These methods learn either a linear embedding, which is often a much lower dimensionality than the input [142, 59, 140], or a non-linear one [33, 69]. More recently, these types of methods have shown up in computer vision. In particular, the line of work on similarity sensitive hashing (or learning an embedding into a binary space) [118, 125] has produced very promising results,

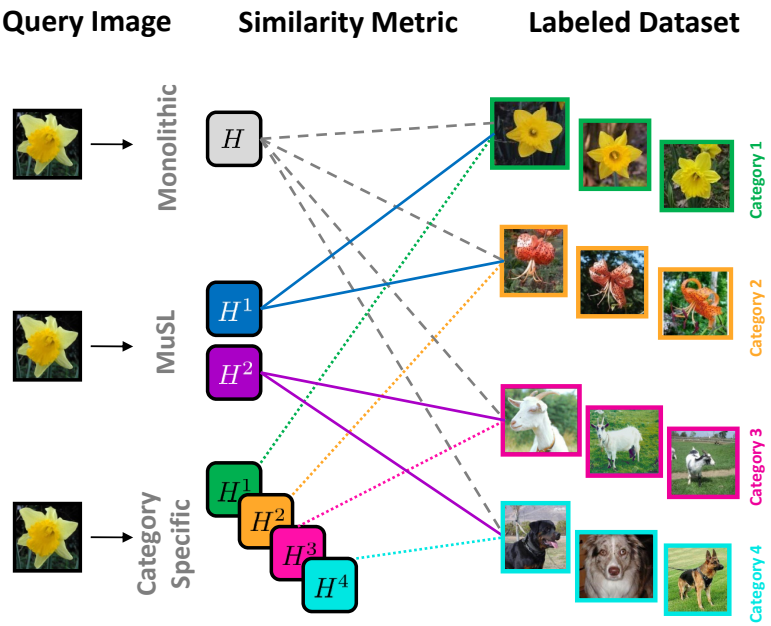


Figure 4.4: **Learning multiple distance metrics with Multiple Similarity Learning (MuSL)**. The top and bottom scenarios correspond to the two common ways of computing similarities between a query image and a labeled training set: in the top row we use a global, or "monolithic", similarity metric; in the bottom row we associate a different similarity metric with each category. The latter is more powerful, but does not scale well to large numbers of categories and cannot generalize to novel categories. The middle row shows a compromise between these two extremes, which we study in this paper.

enabling extremely efficient image retrieval.

There are several advantages to training a monolithic similarity metric. Such a metric can be used in a nearest neighbor classifier, which can lend itself to efficient classification [119]. Furthermore, the representation is the same for all data. This is convenient because the metric can easily generalize to novel categories. In other words, we could train a good similarity metric on C categories, and if we later receive training data for a few more categories, no re-training is necessary (assuming the training data from before was reasonably representative). This form of generalization will be an important focus of our work.

The other end of the spectrum is to train a similarity metric per category. This is typically done in a 1-vs-all manner. A common example of this is training a 1-vs-all classifier that performs some form of feature/cue weighting or selection

(e.g. SVM with kernel combination). The obvious advantage of these methods is improved performance. In fact, these types of methods have recently been shown to perform extremely well for object categorization [131]. This is not surprising since certain cues and features are important for some categories and not others. Therefore, it is difficult to capture all the relevant information in a single global metric. Other work has gone so far as to train a similarity metric per training example [56, 83].

One downside to training category specific metrics is, of course, the difficulty in scaling to large datasets. Computer vision datasets for recognition are growing at exponential rates. Consider, then, a problem with 10,000 categories. Training this many separate similarity metrics is impractical. Furthermore, we speculate that even if we did train this many metrics, many of them would be redundant. Another down side is that, unlike a monolithic similarity metric, it is unclear how this approach could generalize to new categories without an additional training phase.

Our intuition is that a *few good similarity metrics* could capture most of the necessary information and attain good performance without compromising efficiency. We see this as a happy medium in between the two extremes of generic and category specific (c.f. Fig. 4.4). The contributions of this work are twofold: (1) we study how performance changes across this spectrum, and (2) we propose an algorithm called MuSL that simultaneously groups categories together and trains a few similarity metrics, one for each group. We show how to assign one of the learned metrics to novel categories, and study how well these metrics generalize.

Related Work

There are several lines of work that are similar in spirit and motivation to ours. [126] proposes a procedure for training several boosted detectors in a multi-task fashion, forcing them to share features; [102] extends this work to train classifiers incrementally rather than in a single batch. This work, like ours, aims to exploit the redundancies in representation of similar categories. Similarly, in [48] prior information is shared between categories enabling the training of new

classifiers with much less training data. An approach called "dynamic learning" [144] offers an efficient way of training additional classifiers when new categories are introduced or existing categories are split. All of these approaches, however, still require the training of C classifiers for C categories. Instead we take a metric learning view; this enables us to reap some of the benefits of learning a monolithic similarity metric such as the ability to generalize to new categories without retraining.

Work on object taxonomies has similar themes. For example, [64] uses confusion matrices of trained classifiers to group categories together hierarchically. In our work, categories get grouped by virtue of sharing a similarity metric.

Perhaps the most similar approach to ours comes from the machine learning community: in [140] the input space is partitioned, and a metric is learned for each partition. This work requires the user to specify the partitions a priori (in practice, k-means on the input space is used). Our algorithm integrates grouping into the training procedure.

The rest of this paper is organized as follows. In Section 4.2.1 we review boosted similarity metric learning and introduce our algorithm. In Section 4.2.3 we go over implementation details and present object categorization results. We conclude in Section 4.2.4.

4.2.1 Boosting Similarity Classifiers

Recently, [118, 125] explored a boosting approach to embedding images into a binary space. Although this representation sacrifices some recognition power, it is efficient both in terms of computing similarities (which can be done with bit operations) and storage. Furthermore, training a boosted classifier is typically done in phases, which makes it well-suited for the type of problem we address. For these reasons, we choose this paradigm for our study, though the algorithms we discuss could be extended to other representations.

In this section we review the gradient boosting approach to learning similarity metrics, and introduce our proposed algorithm. We receive a dataset of the feature vectors $\{x_1, \ldots, x_n\}$, $x \in \mathbb{R}^d$ and category labels $\{\ell_1, \ldots, \ell_n\}$, where

$\ell_j \in \{1, \ldots, C\}$. Using this dataset we can construct pairs (x_{i1}, x_{i2}); for convenience we also define pair labels $y_i = \mathbf{1}[\ell_{i1} = \ell_{i2}]$. If n_c is the number of examples for each category, then there are $\mathcal{O}(n_c^2)$ possible positive pairs, and $\mathcal{O}(Cn_c^2)$ possible negative pairs. Since the latter can be quite large, in practice we subsample negative pairs.

The goal is to train a boosted similarity metric, which we can think of as a binary classifier that takes a pair as input:

$$H(x_{i1}, x_{i2}) = \sum_{t=1}^{T} h_t(x_{i1}, x_{i2}) \tag{4.5}$$

where each h_t is a weak similarity classifier. Following [118, 125], in our model these weak similarity metrics take the following form:

$$h_t(x_{i1}, x_{i2}) = |f_t(x_{i1}) - f_t(x_{i2})| \tag{4.6}$$

where $f_t : \mathbb{R}^d \to \mathbb{R}^1$. In this case, to compute H we embed each example/image x into a vector $F(x) = [f_1(x), f_2(x), \ldots f_T(x)]$, and then compute the L_1 distance in this new space: $H(x_{i1}, x_{i2}) = ||F(x_{i1}) - F(x_{i2})||_1$. Therefore, training such a strong classifier induces an embedding. We choose the function $f_t : \mathbb{R}^d \to \{0, 1\}$ so that our embedding is binary; this enables us to compute these distances very efficiently. In the simplest case, these functions are computed by thresholding a particular feature of x, $f_t(x) = \mathbf{1}[x[t] < \theta]$. [118] proposes an efficient algorithm to learn such a threshold given some training data. Note that for this choice of f_t, the overall metric $H(x_{i1}, x_{i2})$ ranges from 0 to T.

Monolithic Similarity Classifier

We begin with a review of how to train a single similarity metric for all data. For a given classifier H we can define the probability that a particular pair is positive as

$$p_i = \sigma\Big(\alpha\big(T/2 - H(x_{i1}, x_{i2})\big)\Big) \tag{4.7}$$

where $\sigma(a) = \frac{1}{1-\exp(-a)}$ is the sigmoid function, and α is a global scalar parameter. Using the above definition we can define the log likelihood over a set of training

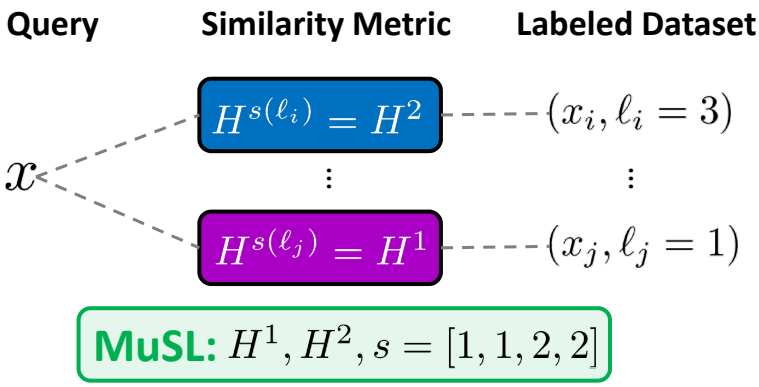


Figure 4.5: **The MuSL system** consists of a few similarity metrics (in this example just two: H^1, H^2) and an assignment vector s that maps each category to one of the metrics. To compute the distance from a query image to the i^{th} example in our labeled training set, we use the similarity metric $H^{s(\ell_i)}$ where ℓ_i is the label of the training example.

pairs:

$$\mathcal{L}(H) = \sum_{i|\ell_{i1}=\ell_{i2}} \log(p_i) + \sum_{i|\ell_{i1}\neq\ell_{i2}} \log(1-p_i) \tag{4.8}$$

To derive a boosting algorithm that optimizes the above objective function we apply Friedman's gradient boosting framework [53]. We interpret boosting as gradient ascent in function space. Each step of the ascent equates to adding a new weak classifier to H. The gradient $\frac{\partial \mathcal{L}(H)}{\partial H}$ gives us the direction in function space in which we should move; however, we are limited by our choice of weak classifier, and cannot move in arbitrary directions. We therefore seek a weak classifier that is as close as possible to this gradient. When the weak classifiers have binary output (as is the case for us), this is equivalent to minimizing weighted error on the training data [13], where the weights are defined as:

$$w_i = \left| \frac{\partial \mathcal{L}(H)}{\partial H} \right| \tag{4.9}$$

In each phase of training we compute these weights and train a weak classifier.

Note that the algorithm presented in [118, 125] is slightly different because an exponential objective function is used instead of log likelihood; we have found that empirical differences are insignificant between these two choices.

Algorithm 6 Multiple Similarity Learning (MuSL)

INPUT: Training data (x_{i1}, x_{i2}) and labels (ℓ_{i1}, ℓ_{i2})

1: **for** $t = 1$ to T **do**
2: **for** $k = 1$ to K **do**
3: Compute weights
$w_i^k = \frac{\exp(r\mathcal{L}_c^k)}{\sum_j \exp(r\mathcal{L}_c^j)} \left|p_i^k - y_i\right|$
4: Train weak classifier h_t^k using weights w_i^k
$h_t^k = \text{argmin}_h \sum_i w_i^k \mathbf{1}[h(x_{i1}, x_{i2}) \neq y_i]$
5: Update strong classifier $H^k \leftarrow H^k + h_t^k$.
6: **end for**
7: **end for**
8: Assign $s(c) = \text{argmax}_k \mathcal{L}_c(H^k)$ for $c = 1 \ldots C$

OUTPUT: Classifiers $H^1 \ldots H^K$, assignment vector s

Per-Category Similarity Classifiers

In this scenario we train a similarity classifier for each category: $H^1, \ldots, H^C$. We can then use these in a kNN framework as follows: upon receiving a novel example x we will compute similarities to all the examples in the training set. To compute the similarity of x and a training example x_i we will use the classifier H^{ℓ_i}.

For each class c, we construct a set of training pairs (x_{i1}, x_{i2}). A pair is positive if $\ell_{i1} = \ell_{i2} = c$ and negative if $\ell_{i1} \neq \ell_{i2}$ and $\ell_{i2} = c$. All other training pairs are not relevant to training H^c. Hence, we optimize the following objective:

$$\mathcal{L}_c(H) = \sum_{i|\ell_{i1}=c,\ell_{i2}=c} \log(p_i) + \sum_{i|\ell_{i1}\neq c,\ell_{i2}=c} \log(1 - p_i) \tag{4.10}$$

We can train each of the C classifiers separately, using the same training procedure as before – the only change is that we train only on relevant pairs rather than all pairs.

Recall that due to our choice of binary weak classifiers, all of the similarity metrics have the same range of output (0 to T). An advantage of this choice is that there are no issues about calibration that sometimes arise in 1-vs-all classification [109]. For other choices of weak classifiers, this issue would need to be addressed.

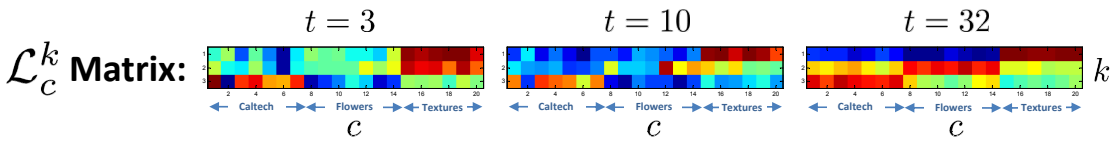


Figure 4.6: **Evolution of likelihoods:** as training proceeds the values of the matrix $\mathcal{L}_c^k$ converge, and clear category groupings become apparent. Above is a snapshot of the matrix (red indicates high likelihood, blue indicates low likelihood) for 3 different stages of training for the MERGED 20 dataset, where we combine 7 Caltech 256 categories, 7 Oxford Flowers categories, and 6 UIUC Texture categories (c.f. Section 4.2.3 for details). Above we see that by the end of the training procedure, the matrix $\mathcal{L}_c^k$ reflects the discovered grouping.

4.2.2 Multiple Similarity Learning (MuSL)

Finally, we are interested in training a small number of similarity metrics $H^1, \ldots, H^K$ where $K < C$. To do this, our algorithm groups categories together into K "super-categories". In other words, we will need to recover a vector s of length C, each entry of which is $s(c) \in \{1, \ldots, K\}$. To compute the similarity between a novel example x and an example from our training set x_i we use the similarity classifier $H^{s(\ell_i)}$ (c.f. Fig. 4.5 for an example). If this assignment vector were known a priori, our problem would reduce to standard training. We can therefore think of this vector as a latent variable. It is plausible to think of heuristic methods of finding this assignment vector as a pre-processing step. However, by separating this step from the training, the two cannot be jointly optimized.

We would like to solve the following optimization problem:

$$\max_{s, H^1 \ldots H^K} \sum_c \mathcal{L}_c^{s(c)}$$

where $\mathcal{L}_c^k = \mathcal{L}_c(H^k)$ is the log likelihood of category c when evaluated with classifier H^k. We can split the max over metrics H and entries of the assignment vector s, and the move the latter into the sum:

$$\max_{H^1, \ldots, H^K} \sum_c \max_{s(c) \in \{1 \ldots K\}} \mathcal{L}_c^{s(c)}$$

Therefore, to solve for the best similarity metrics we use the following objective

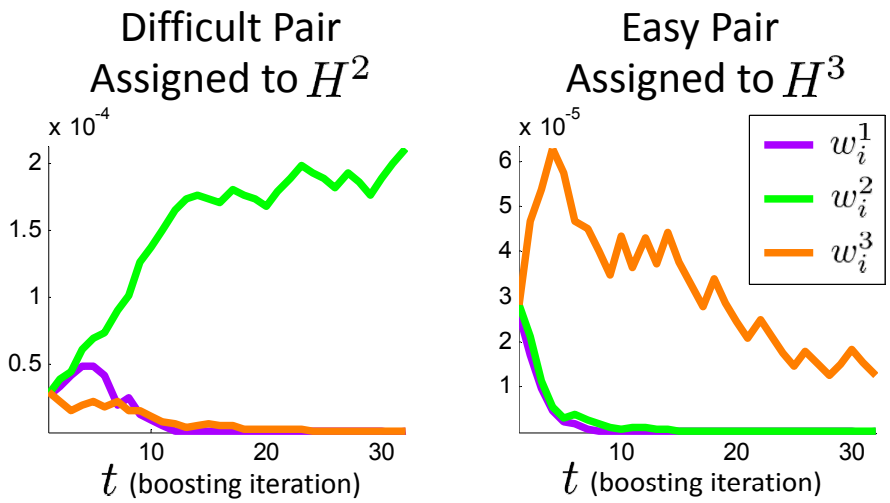


Figure 4.7: **Evolution of pair weights:** the plots above show the weight of two training pairs from the MERGED 20 dataset (c.f. Section 4.2.3 for details) with $K = 3$. As training proceeds, the leftmost term in Eqn. 4.13 softly assigns each category, and hence each training pair, to one of the K metrics; the weights corresponding to the other metrics quickly drop down to 0. In later stages of training the rightmost term in Eqn.4.13 begins to take effect, and the weights of the "difficult" pairs (left plot) stay high, while the weights of the "easy" pairs (right plot) begin to decline; this is similar to traditional boosting.

function[1]:

$$\mathcal{L}(H^1, \ldots, H^K) = \sum_c \max_{k \in \{1 \ldots K\}} \mathcal{L}_c^k \tag{4.11}$$

When $K = 1$ this objection function reduces to the one in Eqn. 4.10. As before we will derive a boosting algorithm to optimize this objective by performing gradient ascent in function space. However, when we try to take the derivative of the above objective function we run into trouble since the max operator is not differentiable. We therefore replace the max with a differentiable approximation [25]:

$$\begin{aligned} \mathcal{G}(a_1, \ldots, a_K) &= \frac{1}{r} \log \big(\sum_k \exp(r a_k) \big) \\ &\approx \max_k \{a_1, \ldots, a_K\} \end{aligned} \tag{4.12}$$

[1]Note that this objective is only concave for $K = 1$; in all other cases it is possible to get stuck in local maxima, though in our experiments we have not found this to be too problematic.

where r is a parameter controls the accuracy of the approximation (we discuss how to deal with possible numerical instability issues of the above equation in Section4.2.3).

We will optimize the objective $\mathcal{L}(H^1, \ldots, H^K)$ by coordinate ascent, updating each of the K classifiers one at a time. In each step we will add a new weak classifier h_t^k to the strong classifier H^k. We train the weak classifier h_t^k with weights derived as follows:

$$\begin{aligned} w_i^k &= \left| \frac{\partial \mathcal{L}(H^1, \ldots, H^K)}{\partial H^k} \right| \\ &= \left| \frac{\partial \mathcal{G}(\mathcal{L}_c^1, \ldots, \mathcal{L}_c^K)}{\partial \mathcal{L}_c^k} \frac{\partial \mathcal{L}_c^k}{\partial p_i^k} \frac{\partial p_i^k}{\partial H^k} \right| \\ &= \frac{\exp(r\mathcal{L}_c^k)}{\sum_j \exp(r\mathcal{L}_c^j)} \left| p_i^k - y_i \right| \end{aligned} \tag{4.13}$$

where p_i^k is the probability of a pair (computed using Eqn. 4.7) according to classifier H^k. The overall algorithm is summarized in Algorithm 1. The above formula has an intuitive interpretation. It is composed of two terms; the rightmost term gives higher weights to pairs that are currently misclassified (e.g. "difficult" pairs), similar to traditional boosting. The leftmost term is the familiar softmax equation [23], applied to the category specific likelihood $\mathcal{L}_c^k$. We can think of this as a soft approximation of $\mathbf{1}[k = \text{argmax}_j \, \mathcal{L}_c^j]$. This term will give higher weight to pairs where $\ell_{i2} = c$ if the similarity metric H^k is the "best" for category c. To gain further intuition for how these two terms interact we plot the evolution of weights for two training pairs in Fig. 4.7.

We initialize all weights to be uniform. As a result, the first few weak classifiers chosen by the procedure are not "tuned" to particular categories. However, as the training proceeds, the leftmost term in the weight equation starts to converge and effectively assign categories to each classifier (c.f. Fig. 4.6 to see how the values of $\mathcal{L}_c^k$ evolve during training).

Assignments and Out of Sample Extensions At the end of the training procedure we recover the assignment vector s as follows:

$$s(c) = \operatorname*{argmax}_k \mathcal{L}_c^k \tag{4.14}$$

This may be obvious for the categories that were included in the training data. The more interesting aspect of the above equation is that it can also be used for out of sample extension to novel categories. That is, suppose we trained K metrics on a few categories, and now we receive training data for one more category; we would like to assign one of the existing metrics to this new category. The above equation is appropriate for this scenario as well: we generate positive and negative training pairs for this category as before, compute log likelihood of each similarity classifier, and pick the best one.

Note that we do not enforce any sort of balance on the assignments; it is possible that all categories get assigned to the same similarity metric. In practice, however, this does not seem to be an issue because assigning metrics to categories more evenly is advantageous in terms of optimizing the objective function.

Re-training Recall that because the weights are initialized uniformly, the first weak classifiers are not tuned to specific categories. If the total number of weak classifiers is large, this would not have a significant effect. However, in our implementation we use a small number of weak classifiers (for efficiency reasons). Once the training phase is complete and we have recovered the category assignments, we *re-train* the similarity metrics in a standard way using the known assignments. This step tends to improve performance.

4.2.3 Experiments

In this section we present our results on object categorization. We begin with an overview of implementation details, which are not required to follow the rest of this Section.

Implementation Details

Image Representation We use a bag of words framework for constructing our weak classifiers for boosting, where each decision stump is a threshold on the count of some visual word. To construct our visual codebook, we use an algorithm similar to [94], where the vocabulary is constructed from a forest of random trees, and each node in a tree corresponds to a visual word. Tree nodes are split by

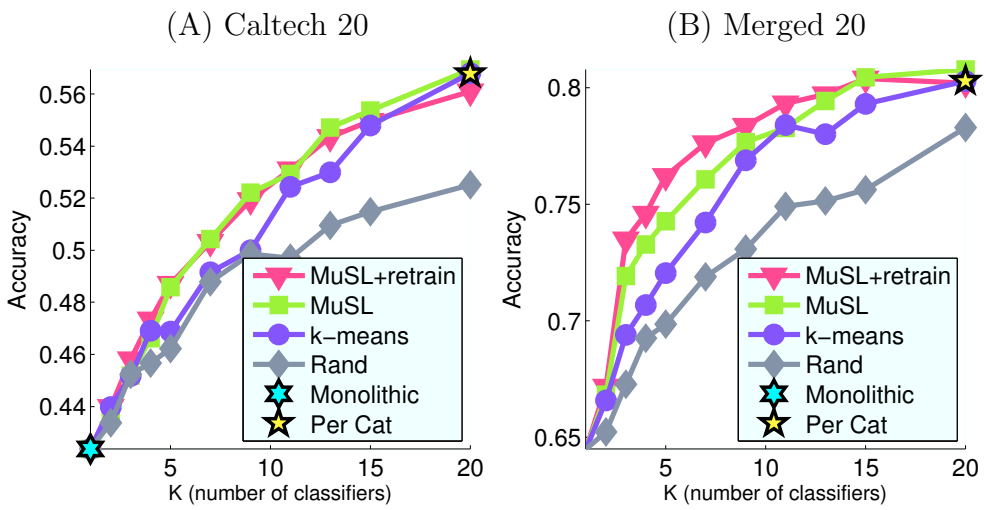


Figure 4.8: **Categorization performance** for two datasets, showing the two extremes, monolithic and per-category similarity metrics, as well as the algorithms discussed for a range of K values. For low values of K we are able to get significant improvement in performance. See Section 4.2.3 for details.

randomly sampling a small number of decision stumps (we used thresholded haar-like filters computed on some image channel), and choosing the split that minimizes the total class entropy on a labeled training set. We constructed two random forests, one using CIE-LUV color images, and another using histogram of gradients images with 8 orientation bins. The main advantage of this style of features is speed: assigning an interest point to a codeword involves evaluating a small number of haar-like features, and thus an image can be processed in a fraction of a second. It also allows for easy integration of multiple cues by using different types of image channels. We emphasize that our metric learning algorithm is not in any way tied to this style of features, and we expect that absolute performance could be improved by using other more advanced features.

Algorithm Parameters Here we briefly discuss the parameters and caveats of our algorithm. When constructing training pairs given labeled data we randomly sample 1,000 negative pairs for each example and take every possible positive pair.

One potential problem with the expression in Eqn. 4.13 is that the exponent can blow up for large values of $\mathcal{L}_c^k$. Recall the parameter r in Eqn. 4.12;

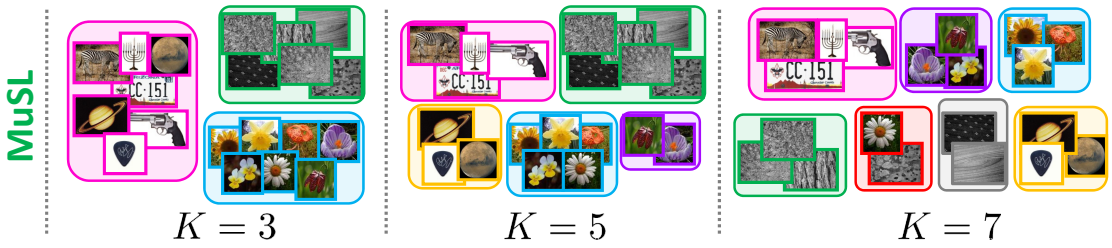


Figure 4.9: Discovered category groupings for the Merged 20 dataset for various values of K using MuSL. See Section 4.2.3 for details.

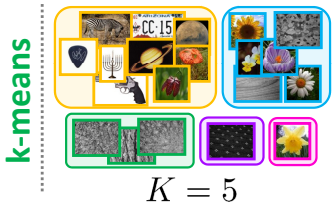


Figure 4.10: Discovered category groupings for the Merged 20 dataset for various values of K using the k-means heuristic. See Section 4.2.3 for details.

we can set this parameter to a small value to avoid numerical instability. The value of the likelihood $\mathcal{L}_c^k$ depends on the number of examples n. We found that setting $r = 500/n$ is a good compromise between avoid numerical problems and having reasonable accuracy in the max approximation. We set the number of weak classifiers $T = 32$. Finally, as suggested in [125], we set $\alpha = 0.1$.

Categorization

Our first task is to measure categorization performance as a function of K. We plug our learned similarity metrics into a nearest neighbor classifier, as described in Section 4.2.1. Recall that we are using embeddings into binary spaces, as was done in [125, 118] (e.g. $f_t : \mathbb{R}^d \to \{0, 1\}$). There are a few advantages to this choice. First, computing distances in binary space is extremely efficient. Furthermore, since each weak classifier relies on only one feature, boosting essentially performs feature selection. This can further speed up run time as we need to compute fewer features for an incoming image. Nevertheless, as mentioned before, this

choice also limits the power of our learned metrics to some degree. Therefore, our results are not state of the art in terms of absolute performance – we are mainly interested in studying the relative performance of these algorithms as we change the value of K.

We compare the MuSL algorithm to two baselines, both of which perform grouping of categories as a pre-processing step, and train similarity metrics in a standard way. The first baseline is to group the categories randomly. The second is to use the k-means clustering algorithm [23] to group the categories. To do this, we first need a way of representing each category in some feature space. While it is not clear what the "correct" way of doing this is, we use the following heuristic: we take a mean of all the data points corresponding to that category $\overline{x}_c = 1/n_c \sum_{\ell_i=c} x_i$; this results in C vectors, one for each category. We experimented with other k-means based heuristics, but found this to be the only one that gives reasonable performance.

We perform experiments with two different datasets described below. For all experiments we use 30 training images per category for training, and 10 to 20 images per category for testing. The results shown are averages of 20 trials with different train/test splits. In all experiments we report the average recognition accuracy (mean of the diagonal of the confusion matrix).

Caltech

Our first experiment uses the Caltech 256 (CT) dataset [63]. We took the 30 easiest categories[2] and randomly chose a subset of 20 out of these (the other 10 will be used in later experiments); we call this dataset "Caltech 20". Categorization accuracy for various values of K are shown in Fig. 4.8(A). As we slide the value of K from 1 to 20 the recognition performance gradually increases; the difference between the two extremes is 14%. Randomly grouping categories together performs worse than the "smarter" ways of discovering super-categories, suggesting that certain groupings are better than others. Finally, MuSL tends to perform the best out of all the methods, though in this application the k-means

[2]Difficulty is measured by performance of a standard classification method; see [63] for details.

heuristic performs fairly well. We will see some disadvantages to the k-means heuristic in the next two sections.

Merged One strength of our algorithm is its ability to handle datasets that consist of a wide variety of heterogenous categories. We constructed a dataset we called "Merged 20"; it consists of 7 categories from CT [63], 6 categories from Oxford Flowers-17 (FL) [98], and 6 categories from UIUC Textures (UT) [74]. The CT categories were chosen randomly from the 20 categories in the previous section; the FL and UT categories were chosen randomly from their respective datasets. Fig. 4.8(B) shows a plot of recognition accuracy versus K. We see that for low values of K there is a dramatic increase in performance – for $K = 5$ recognition accuracy is 10% higher than for a single metric. MuSL outperforms both the random and the k-means heuristic methods of grouping; for $K = 5$ MuSL achieves an accuracy almost 5% higher than the others. Finally, we see that the re-trained MuSL metrics increase performance by a couple percent.

It is interesting to qualitatively inspect the super-categories that MuSL discovers; one of the advantages of this dataset is that we have some expectation for what these super-categories could be. Fig. 4.9 shows the discovered groupings for $K = 3, 5, 7$ for a particular train/test fold. For $K = 3$ the groupings correspond to the 3 source datasets that we merged; MuSL recovers these super-categories automatically. The k-means method of grouping, on the other hand, does not recover these super-categories. For $K = 5$ MuSL breaks the 3 source datasets down further, in a seemingly intuitive manner. For example, the 'mars', 'saturn' and 'guitar-pick' categories get grouped together; these objects are roughly circular and have strong gradients on their boundaries due to constant backgrounds. Again, the super-categories discovered by k-means are semantically arbitrary (c.f. Fig.4.10 for an example).

Finally, we note that the groupings produced by MuSL are much more consistent, or "stable", over the different train/test folds: using the stability measure defined in [73], for $K = 3$ MuSL groupings are 96% stable, while k-means groupings are only 50% stable. This means that super-categories discovered by MuSL are almost always the same, whereas those discovered by k-means are fairly random.

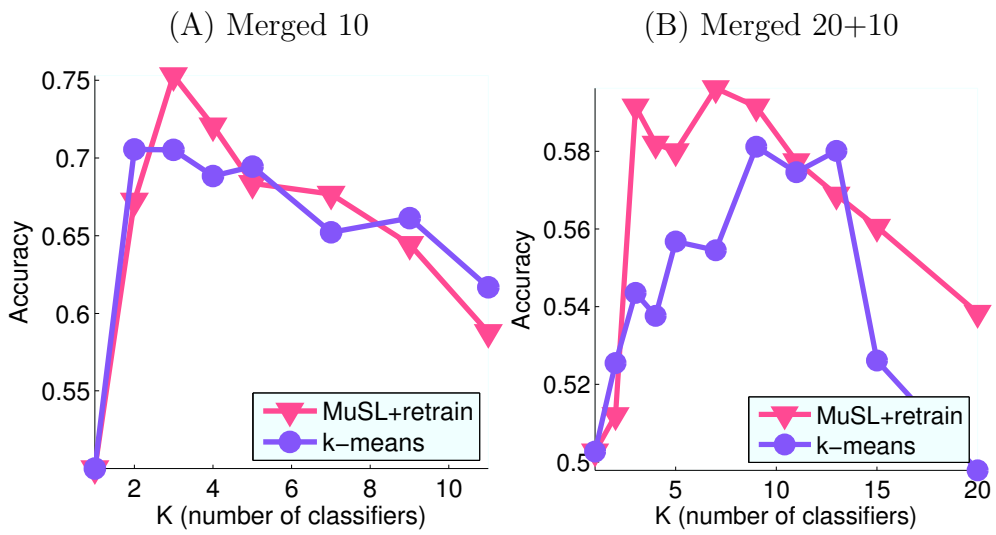


Figure 4.11: **Generalizing to novel categories**. Using a few similarity metrics achieves much better performance than using just one; however, using too many can overfit to the original categories. See Sec. 4.2.3 for details.

Generalizing to New Categories

We would like to study the ability of the learned similarity metrics to generalize to new categories. It is important to highlight the difference of this generalization as opposed to the traditional definition. Here we are interested in generalizing to novel categories (both train and test data), rather than generalizing to test data that consists of the same categories as the training data. To this end, we create a new dataset we call "Merged 10"; it consists of 10 new categories (4 from CT, 3 from FL, and 3 from UT) that were not included in Merged 20. We train similarity metrics only on the original categories from the previous section (Merged 20), and test on these new categories, as well as a combination of all 30 categories, which we call "Merged 20+10". To do this for $K > 1$ we use the procedure described in Sec.4.2.2 to assign one of the learned similarity metrics to each of the new categories. For the Merged 10 data, when $K = 3$ we observe that the assignments are as expected – each of the source datasets get grouped together as before. This suggests that the assignment procedure is reasonable.

We plot the performance of MuSL (the re-trained version) and the k-means

heuristic method as we sweep through values of K in Fig. 4.11. The left plot shows performance only on the new categories, and the right plot shows performance on all 30 categories. The resulting plots are surprising. Unlike before, performance does *not* strictly improve as K gets larger. Instead, performance actually starts to degrade for higher values of K. This suggests that training with a lower value of K results in more general similarity metrics. Therefore, having too many similarity metrics is not only computationally wasteful (in terms of training time and storage), the performance is actually worse when adding new categories to the data. At the same time, training just one similarity metric performs much worse (almost 25% worse than when $K = 3$ on the Merged 10 dataset). Furthermore, we see that MuSL performs better than the k-means heuristic method for both datasets – for the Merged 20+10 dataset the difference is about 5%.

4.2.4 Conclusions

In this paper we presented a method for learning a few similarity metrics from labeled data. We studied how performance changes in between the two extremes of a single metric and a metric per object category, and showed that the performance of the latter can be matched fairly closely with a small number of metrics. We also studied how these learned metrics generalize to novel categories; such generalization is strongly desirable if we wish to scale to large datasets. Here we saw that training *too many* metrics is actually detrimental to this type of generalization. The algorithm we proposed, MuSL, simultaneously trains a few similarity metrics and groups categories together. Though a number of heuristic methods for grouping categories as a pre-processing step are possible, our method is principled and tends to perform better because the optimization of metrics and grouping is done jointly. In particular, our method is well suited to scenarios where the categories exhibit a super-categorical structure.

There are several paths we would like to explore in the future. First, we are currently working on experiments with larger datasets to see how these methods scale, and investigating more powerful features. Though our focus in this work is categorization, the ideas we presented could prove to also be useful in the domain of

image retrieval. For example, given an image query and a large unlabeled dataset of images, the task of retrieving images similar to the query is ambiguous when we acknowledge the fact that there is no single "correct" similarity metric. Perhaps it is best to return several sets of results to the user, one for each of the similarity metrics. We intend to explore these ideas in the future.

4.3 Conclusions

Portions of this chapter are based on the following publications:

- "Simultaneous Learning and Alignment: Multi-Instance and Multi-Pose Learning" by B. Babenko, P. Dollár, Z. Tu, and S. Belongie [13]. The dissertation author contributed to algorithm development, implemented code and experiments, and contributed to writing of the paper.
- "Similarity Metrics for Categorization: from Monolithic to Category Specific" by B. Babenko, S. Branson, and S. Belongie [11]. The dissertation author developed the algorithm and experiments, and wrote most of the paper.

5 Theoretical Analysis of Multiple Instance Learning

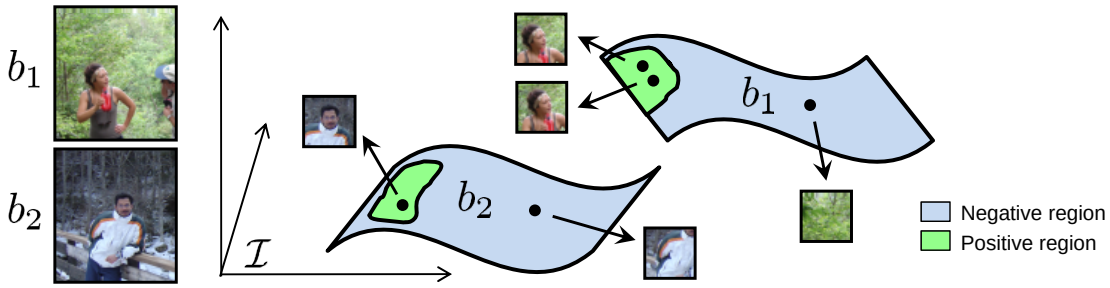


Figure 5.1: **Manifold bags**. In this example the task is to predict whether an image contains a face. Each bag is an image, and individual instances are image patches of a fixed size. Examples of two positive bags b_1 and b_2 (left), and a visualization of the instance space $\mathcal{I}$ (right) are shown. The two bags trace out low-dimensional manifolds in $\mathcal{I}$; in this case the manifold dimension is two since there are two degrees of freedom (the x and y location of the image patch). The green regions on the manifolds indicate the portion of the bags that is positive.

In this chapter we will explore the theoretical aspects of Multiple Instance Learning. In particular we will argue that existing analysis of MIL is not appropriate for many applications in computer vision (and other areas like computer audition), and propose an alternative. Note that in this chapter we will use slightly different notation than the rest of the dissertation to be consistent with related work.

Theoretical PAC-style analysis of MIL problems has also seen progress in the last decade [8, 24, 85, 113, 112]. Typical analysis formulates the MIL problem as follows: a fixed number of instances, r, is drawn from an instance space $\mathcal{I}$ to form

a bag. The sample complexity for bag classification is then analyzed in terms of the bag size (r). Most of the theory work has focused on reducing the dependence on r under various settings. For example, [24] showed that if one has access to a noise tolerant learner and the bags are formed by drawing r independent samples from a fixed distribution over $\mathcal{I}$, then the sample complexity grows linearly with r. Recently, [113] showed that if one can minimize the empirical error on bags, then even if the instances in a bag have arbitrary statistical dependence, sample complexity grows only logarithmically with r.

The above line of work is rather restrictive. Any dependence on r makes it impossible to apply these generalization bounds to problems where bags have infinitely many instances – a typical case in practice. Consider the following motivating example: we would like to predict whether an image contains a face, as in [136]. Putting this in the MIL framework, a bag is an entire image, which is labeled positive if and only if there is a face in that image. The individual instances are image patches. Notice that in this scenario the instances collectively form (a discrete approximation to) a low-dimensional manifold; see Figure 5.1. Here we expect the sample complexity to scale with the geometric properties of the underlying manifold bag rather than the number of instances per bag.

This situation arises in many other MIL applications where some type of sliding window is used to break up an object into many overlapping pieces: images [6, 136], video [3, 29], audio [116, 89], and sensor data [123]. Consider also the original molecule classification task that motivated [41] to develop MIL, where a bag corresponds to a molecule, and instances are different shapes that molecule can assume. Even in this application, "as the molecule changes its shape, it traces out a manifold through [feature] space" [91]. Thus, manifold structure is an integral aspect of these problems that needs to be taken into account in MIL analysis and algorithm design.

In this work we analyze the MIL framework for bags containing potentially infinite instances. In this setting a bag is drawn from a bag distribution, and is labeled positive if it contains at least one positive instance. In order to have a tractable analysis, we impose a structural constraint on the bags: we assume that

bags are low dimensional manifolds in the instance space, as discussed above. We show that the geometric structure of such bags is intimately related to the PAC-learnability of MIL problems. We investigate how learning is affected if we have have access to only a limited number of instances per manifold bag. We then discuss how existing MIL algorithms, that are designed for finite sized bags, can be adapted to learn from manifold bags efficiently using an iterative querying heuristic. Our experiments on real-world data (image and audio) validate the intuition of our analysis and show that our querying heuristic works well in practice.

5.1 Problem Formulation and Analysis

Let $\mathcal{I}$ be the domain of instances (for the purposes of our discussion we assume it to be $\mathbb{R}^N$ for some large N), and let $\mathcal{B}$ be the domain of bags. Here we impose a structural constraint on $\mathcal{B}$: each bag from $\mathcal{B}$ is a low dimensional manifold over the instances of $\mathcal{I}$. More formally, each bag $X \in \mathcal{B}$ is a smooth bijection[1] from $[0,1]^n$ to some subset of $\mathcal{I}$ ($n \ll N$). The geometric properties of such bags are integral to our analysis. We will thus do a quick review of the various properties of manifolds that will be useful in our discussion.

5.1.1 Differential Geometry Basics

Let f be a smooth bijective mapping from $[0,1]^n$ to $M \subset \mathbb{R}^N$. We call the image of f (i.e. M) a manifold (note that M is compact and has a boundary). The *dimension* of the domain (n in our case) corresponds to the latent degrees of freedom and is typically referred to as the intrinsic dimension of the manifold M (denoted by $\textsc{dim}(M)$).

Since one of the key quantities in classic analysis of MIL is the bag size, we require a similar quantity to characterize the "size" of M. One natural way to characterize this is in terms of the volume of M. The *volume* (denoted by $\textsc{vol}(M)$) is given by the quantity $\int_{u_1,\ldots,u_n} \sqrt{\det(J^\intercal J)} du_1 \ldots du_n$, where J is the

[1]Here we are only considering a restricted class of manifolds – those that are globally diffeomorphic to $[0,1]^n$. This is only done for convenience. The results here are generalizable to arbitrary (compact) manifolds.

$N \times n$ Jacobian matrix of the function f, with individual entries defined as $J_{ij} := \partial f_i / \partial u_j$.

Unlike a finite size bag, a finite volume manifold $M \subset \mathbb{R}^N$ can be arbitrarily "complex" – it can twist and turn in all sorts of ways in the surrounding space. We therefore need to also get a handle on its curviness. Borrowing the notation from computational geometry literature, we can characterize the complexity of M via its condition number (see [99]). We say that the *condition number* of M (denoted by $\textsc{cond}(M)$) is $\frac{1}{\tau}$, if τ is the largest number such that the normals of length $r < \tau$ at any two distinct points in M don't intersect. One can bound the sectional curvature of M at any point by $1/\tau$. Hence, when τ is large, the manifold is relatively flat and vice versa.

With these definitions, we can define a structured family of bag spaces.

Definition 1. *We say that a bag space $\mathcal{B}$ belongs to class (V, n, τ), if for every $X \in \mathcal{B}$, we have*[2] *that* $\textsc{dim}(X) = n$, $\textsc{vol}(X) \leq V$, *and* $\textsc{cond}(X) \leq 1/\tau$.

In what follows, we will assume that $\mathcal{B}$ belongs to class (V, n, τ). We now provide our main results, with all the supporting proofs provided in [14]

5.1.2 Learning with Manifold Bags

Since we are interested in PAC-style analysis, we will be working with a fixed hypothesis class $\mathcal{H}$ over the instance space $\mathcal{I}$ (that is, each $\mathbf{h} \in \mathcal{H}$ is of the form $\mathbf{h} : \mathcal{I} \to \{0, 1\}$). The corresponding *bag* hypothesis class $\overline{\mathcal{H}}$ over the bag space $\mathcal{B}$ (where each $\bar{h} \in \overline{\mathcal{H}}$ is of the form $\bar{h} : \mathcal{B} \to \{0, 1\}$) is defined as the set of classifiers $\{\bar{h} : \mathbf{h} \in \mathcal{H}\}$ where, for any $X \in \mathcal{B}$, $\bar{h}(X) \stackrel{\text{def}}{=} \max_{\alpha \in [0,1]^n} \mathbf{h}(X(\alpha))$. We assume that there is some unknown instance classification rule $\mathbf{h}^* : \mathcal{I} \to \{0, 1\}$ that gives the true labels for all instances.

The learner gets access to m bag/label pairs $(b_i, y_i)_{i=1}^m$, where each bag X_i is drawn independently from an unknown but fixed distribution $\mathcal{D}_{\mathcal{B}}$ over $\mathcal{B}$, and is labeled according to the MIL rule $y_i \stackrel{\text{def}}{=} \max_{\alpha \in [0,1]^n} \mathbf{h}^*(X_i(\alpha))$. We denote a sample of size m as S_m.

[2]Technically b is a function and *not* a manifold. For readability, we will occasionally abuse the notation and use b to mean the manifold produced by the image of b in the instance space $\mathcal{I}$.

Our learner should ideally return the hypothesis $\bar{h}$ that achieves the lowest bag generalization[3] error: $\text{err}(\bar{h}) \stackrel{\text{def}}{=} \mathbb{E}_{X \sim \mathcal{D}_{\mathcal{B}}}[\bar{h}(X) \neq y]$. This, of course, is not possible as the learner typically does not have access to the underlying data distribution $\mathcal{D}_{\mathcal{B}}$. Instead, the learner has access to the sample S_m, and can minimize the *empirical* error: $\widehat{\text{err}}(\bar{h}, S_m) \stackrel{\text{def}}{=} \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\{\bar{h}(X_i) \neq y_i\}$. Various PAC results relate these two quantities in terms of the properties of $\overline{\mathcal{H}}$.

Perhaps the most obvious way to bound $\text{err}(\bar{h})$ in terms of $\widehat{\text{err}}(\bar{h}, S_m)$ is by analyzing the VC-dimension of the bag hypotheses, $\text{VC}(\overline{\mathcal{H}})$, and applying the standard VC-bounds (see e.g. [130]). While finding the VC-dimension of the bag hypothesis class is non-trivial, the VC-dimension of the corresponding *instance* hypotheses, $\text{VC}(\mathcal{H})$, is well known for many popular choices of $\mathcal{H}$. [113] showed that for finite sized bags the VC-dimension of *bag* hypotheses (and thus the generalization error) can be bounded in terms of the VC-dimension of the underlying *instance* hypotheses. Although one might hope that this analysis could be carried over to bags of infinite size that are well structured, this turns out to not be the case.

$\text{VC}(\overline{\mathcal{H}})$ is Unbounded for Arbitrarily Smooth Manifold Bags

We begin with a surprising result which goes against our intuition that requiring bag smoothness should suffice in bounding $\text{VC}(\overline{\mathcal{H}})$. We demonstrate that requiring the bags to be low-dimensional, arbitrarily flat manifolds with fixed volume is not enough to get a handle on generalization error even for one of the simplest instance hypothesis classes (set of hyperplanes in $\mathbb{R}^N$). In particular,

Theorem 2. *For any $V > 0$, $n \geq 1$, $\tau < \infty$, let $\mathcal{B}$ contain all manifolds M such that $\dim(M) = n$, $\text{VOL}(M) \leq V$, and $\text{COND}(M) \leq 1/\tau$ (i.e. $\mathcal{B}$ is the largest member of class (V, n, τ)). Let $\mathcal{H}$ be the set of hyperplanes in $\mathbb{R}^N$ ($N > n$). Then for any $m \geq 1$, there exists a set of m bags $X_1, \ldots, X_m \in \mathcal{B}$, such that the corresponding bag hypothesis class $\overline{\mathcal{H}}$ (over the bag space $\mathcal{B}$) realizes all possible 2^m labelings.*

[3] One can also talk about the generalization error over instances. As noted in previous work (e.g., sabato09), PAC analysis of the instance error typically requires stronger assumptions.

Thus, VC($\overline{\mathcal{H}}$) is unbounded making PAC-learnability seemingly impossible. To build intuition for this apparent richness of $\overline{\mathcal{H}}$, and possible alternatives to bound the generalization error, let us take a quick look at the case of one-dimensional manifolds in $\mathbb{R}^2$ with halfspaces as our $\mathcal{H}$. For any m, we can place a set of m manifold bags in such a way that all labelings are realizable by $\overline{\mathcal{H}}$ (see Fig. 5.2 for an example where $m = 3$; see Appendix of [14] for a detailed construction).

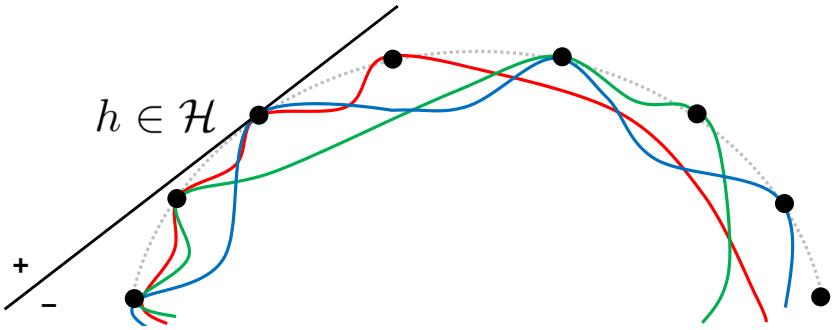
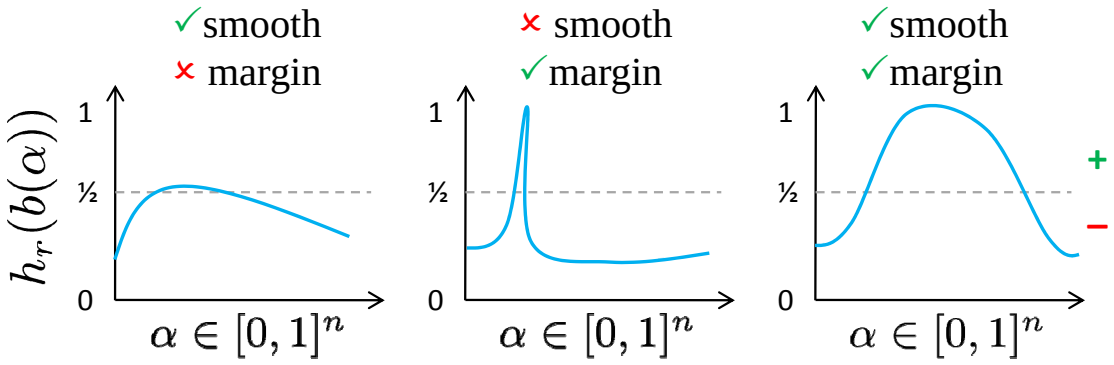


Figure 5.2: **Bag hypotheses over manifold bags have unbounded VC-dimension.** Three bags (colored blue, green and red) go around the eight anchor points (shown as black dots) that are arranged along a section of a circle. Notice that the hyperplanes tangent to the anchor points achieve all possible bag labelings. The hypothesis **h** shown above, for example, labels the red and blue bags positive, and the green bag negative.

The key observation is that in order to label a bag positive, the instance hypothesis needs to label just a *single* instance in that bag positive. Considering that our bags have an infinite number of points, the positive region can occupy an arbitrarily small fraction of a positively labeled bag. This gives our bag hypotheses immense flexibility even when the underlying instance hypotheses are quite simple.

It seems that to bound err($\bar{h}$) we must ensure that a non-negligible portion of a positive bag be labeled positive. A natural way of accomplishing this is to use a real-valued version of the instance hypothesis class (i.e., classifiers of the form $\mathbf{h}_r : \mathcal{I} \to [0, 1]$, and labels determined by thresholding), and requiring that functions in this class (a) be *smooth*, and (b) label a positive bag with a certain *margin*. To understand why these properties are needed, consider three ways that $\mathbf{h}_r$ can label the instances of a positive bag X as one varies the latent parameter

α (i.e., x-axis corresponds to instances, y-axis corresponds to classifier output):



In both the left and center panels, $\mathbf{h}_r$ labels only a tiny portion of the bag positive: in the first case $\mathbf{h}_r$ barely labels any instance above the threshold of 1/2, resulting in a small margin; in the second case, although the margin is large, $\mathbf{h}_r$ changes rapidly along the bag. Finally, in the right panel, when both the margin and smoothness conditions are met, a non-negligible portion of X is labeled positive.

We shall thus study how to bound the generalization error in this setting.

Learning with a Margin

Let $\mathcal{H}_r$ be the real-valued relaxation of $\mathcal{H}$ (i.e. each $\mathbf{h}_r \in \mathcal{H}_r$ is now of the form $\mathbf{h}_r : \mathcal{I} \to [0,1]$). In order to ensure smoothness we impose a λ-Lipschitz constraint on the instance hypotheses: $\forall \mathbf{h}_r \in \mathcal{H}_r,\ x, x' \in \mathcal{I},\ |\mathbf{h}_r(x) - \mathbf{h}_r(x')| \leq \lambda \|x - x'\|_2$. We denote the corresponding bag hypothesis class as $\overline{\mathcal{H}}_r$. Note that the true bag labels are still binary in this setting (i.e. determined by $\mathbf{h}^*$).

Similar to the VC-dimension, the "fat-shattering dimension" of a real-valued bag hypothesis class, $\text{FAT}_\gamma(\overline{\mathcal{H}}_r)$, relates the generalization error to the empirical error at margin γ (see for example [7]):

$$\widehat{\text{err}}_\gamma(\bar{h}_r, S_m) \stackrel{\text{def}}{=} \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\{\text{MARGIN}(\bar{h}_r(X_i), y_i) < \gamma\}, \tag{5.1}$$

where $\text{MARGIN}(x, y) \stackrel{\text{def}}{=} \begin{cases} x - 1/2 & y = 1 \\ 1/2 - x & \text{otherwise} \end{cases}$.

Recall that it was not possible to bound generalization error in terms of the instance hypotheses using VC dimension. However, analogous to Sabato & Tishby's analysis of finite size bags [113], we *can* bound generalization error for manifold bags in terms of the fat-shattering dimension of instance hypotheses, $\text{FAT}_\gamma(\mathcal{H})$. In particular, we have the following:

Theorem 3. *Let $\mathcal{B}$ belong to class (V, n, τ). Let $\mathcal{H}_r$ be λ-Lipschitz smooth (w.r.t. ℓ_2-norm), and $\overline{\mathcal{H}}_r$ be the corresponding bag hypotheses over $\mathcal{B}$. Pick any $0 < \gamma < 1$ and $m \geq \text{FAT}_{\gamma/16}(\mathcal{H}_r) \geq 1$. For any $0 < \delta < 1$, we have with probability at least $1 - \delta$ over an i.i.d. sample S_m (of size m), for every $\bar{h}_r \in \overline{\mathcal{H}}_r$:*

$$err(\bar{h}_r) \leq \widehat{err}_\gamma(\bar{h}_r, S_m) + O\left(\sqrt{\frac{n^2 \text{FAT}_{\frac{\gamma}{16}}(\mathcal{H}_r)}{m} \log^2\left(\frac{Vm}{\gamma^2 \tau_0^n}\right) + \frac{1}{m} \ln \frac{1}{\delta}}\right),$$

where $\tau_0 = \min\{\frac{\tau}{2}, \frac{\gamma}{8}, \frac{\gamma}{8\lambda}\}$.

Observe that the complexity term in Eq. (5.2) is independent of the "bag size"; it has instead been replaced by the volume and other geometric properties of the manifold bags. The other term captures the sample error for individual hypotheses at margin γ. Thus a natural strategy for a learner is to return a hypothesis that minimizes the empirical error while maximizing the margin.

5.1.3 Learning from Queried Instances

So far we have analyzed the MIL learner as a black box entity, which can minimize the empirical bag error by somehow accessing the bags. Since the individual bags in our case are low-dimensional manifolds (with an infinite number of instances), we must also consider *how* these bags are accessed by the learner. Perhaps the simplest approach is to query ρ instances uniformly from each bag, thereby "reducing" the problem to standard MIL (with finite size bags) for which there are algorithms readily available (e.g., [91, 6, 148, 136]). More formally, for a bag sample S_m, let $p_1^i, \ldots, p_\rho^i$ be ρ independent instance samples drawn uniformly from the (image of) bag $X_i \in S_m$, and let $S_{m,\rho} \stackrel{\text{def}}{=} \bigcup_{i,j} p_j^i$ be

the set of all instances. Assuming that our manifold bags have well-conditioned boundaries, the following theorem relates the empirical error of sampled bags, $\widehat{\mathrm{err}}_\gamma(\bar{h}_r, S_{m,\rho}) \stackrel{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}\{\textsc{margin}(\max_{j\in[\rho]} \mathbf{h}(p_j^i), y_i) < \gamma\}$, to the generalization error.

Theorem 4. *Let $\mathcal{B}$ belong to class (V, n, τ). Let $\mathcal{H}_r$ be λ-Lipschitz smooth (w.r.t. ℓ_2-norm), and $\overline{\mathcal{H}}_r$ be the corresponding bag hypotheses over $\mathcal{B}$. Pick any $0 < \delta_1, \delta_2 < 1$, then with probability at least $1 - \delta_1 - \delta_2$, over the draw of m bags (S_m) and ρ instances per bag ($S_{m,\rho}$), for all $\bar{h}_r \in \overline{\mathcal{H}}_r$ we have the following:*

Let $\frac{1}{\kappa} \stackrel{def}{=} \max_{X_i \in S_m}\{\textsc{cond}(\partial X_i)\}$ (where ∂X_i is the boundary of the manifold bag X_i) and set $\tau_1 = \min\{\frac{\tau}{32}, \frac{\kappa}{8}, \frac{\gamma}{9\lambda}, \frac{\gamma}{9}\}$. If

$$\rho \geq \Omega\left(\left(V/\tau_1^{\mathrm{c}_0 n}\right)\left(n + \ln\left(\frac{mV}{\tau_1^n \delta_2}\right)\right)\right),$$

then

$$err(\bar{h}_r) \leq \widehat{err}_{2\gamma}(\bar{h}_r, S_{m,\rho}) + O\left(\sqrt{\frac{n^2 \mathrm{FAT}_{\frac{\gamma}{16}}(\mathcal{H}_r)}{m} \log^2\left(\frac{Vm}{\gamma^2 \tau_0^n}\right) + \frac{1}{m}\ln\frac{1}{\delta_1}}\right),$$

where $\tau_0 = \min\{\frac{\tau}{2}, \frac{\gamma}{8}, \frac{\gamma}{8\lambda}\}$ and c_0 is an absolute constant.

Notice the effect of the two key parameters in the above theorem: the number of training bags, m, and the number of queried instances per bag, ρ. Increasing either quantity improves generalization – increasing m drives down the error (via the complexity term), while increasing ρ helps improve the confidence (via δ_2). While ideally we would like both quantities to be large, increasing these parameters is, of course, computationally burdensome for a standard MIL learner. Note, however, the difference between m and ρ: increasing m comes at an *additional cost* of obtaining extra labels, whereas increasing ρ does not. We would therefore like an algorithm that can take advantage of using a large ρ while avoiding computational costs.

Iterative Querying Heuristic

As we saw in the previous section, we would ideally like to train with a large number of queried instances, ρ, per training bag. However, this may be impractical in terms of both speed and memory constraints. Suppose we have access to a black box MIL algorithm $\mathcal{A}$ that can only train with $\hat{\rho} < \rho$ instances per bag at once.

We propose a procedure called Iterative Querying Heuristic (IQH), described in detail in Algorithm 1 (the main steps are highlighted in blue).

Algorithm 7 Iterative Querying Heuristic (IQH)

Input: Training bags $(b_1, \ldots, b_m)$, labels $(y_1, \ldots, y_m)$, parameters T, ω and $\hat{\rho}$

1: Initialize $I_i^0 = \emptyset$, $\mathbf{h}_r^0$ as any classifier in $\mathcal{H}_r$.

2: **for** $t = 1, \ldots, T$ **do**

3: **Query ω new candidate instances per bag**:
$Z_i^t := I_i^{t-1} \cup \{p_1^i, \ldots, p_\omega^i\}$ where $p_j^i \sim b_i$, $\forall i$.

4: **Keep $\hat{\rho}$ highest scoring inst. using $\mathbf{h}_r^{t-1}$**:
$I_i^t \subset Z_i^t$ s.t. $|I_i^t| = \hat{\rho}$ and $\mathbf{h}_r^{t-1}(p) \geq \mathbf{h}_r^{t-1}(p')$
for all $p \in I_i^t, p' \in Z_i^t \setminus I_i^t$.

5: **Train $\bar{h}_r^t$ with the selected instances**:
$\bar{h}_r^t \leftarrow \mathcal{A}(\{I_1^t \ldots I_m^t\}, \{y_1 \ldots y_m\})$.

6: **end for**

7: **Return $\mathbf{h}_r^T$** and the corresponding $\bar{h}_r^T$

Notice that IQH uses a total of $T\hat{\rho}$ instances per bag for training (T iterations times $\hat{\rho}$ instances per iteration). Thus, setting $T \approx \rho/\hat{\rho}$ should achieve performance comparable to using ρ instances at once. The free parameter ω controls how many new instances are considered in each iteration.

The intuition behind IQH is as follows. For positive bags, we want to ensure that at least one of the queried instances is positive; hence we use the current estimate of the classifier to select the most positive instances. For negative bags, we know all instances are negative. In this case we select the instances that are closest to the decision boundary of our current classifier (corresponding to the most difficult negative instances); the motivation for this is similar to bootstrapping

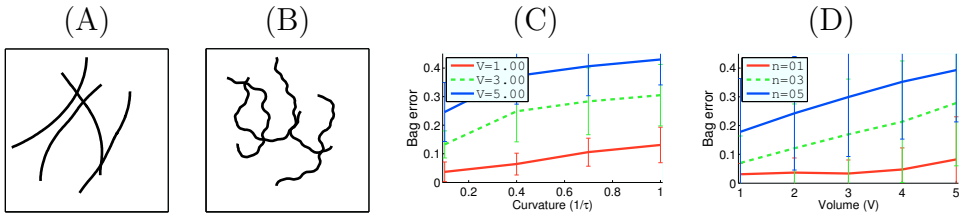


Figure 5.3: **Synthetic Data Results:** Examples of four synthetically generated bags in $\mathbb{R}^2$ with **(A)** low curvature and **(B)** high curvature. **(C)** and **(D)**: Test error scales with the manifold parameters: volume (V), curvature ($\frac{1}{\tau}$), and dimension (n).

negative examples [49] and some active learning techniques [34]. We then use these selected instances to find a better classifier.

Thus one expects IQH to take advantage of a large number of instances per bag, without actually having to train with all of them at one time.

5.2 Experiments

Recall that we have shown that the generalization error is bounded in terms of key geometric properties of the manifold bags, such as curvature ($1/\tau$) and volume (V). Here we will experimentally validate that generalization error does indeed scale with these quantities, providing an empirical lower bound. Additionally, we study how the choice of ρ affects the error, and show that our Iterative Heuristic (IQH) is effective in reducing the number of instances needed to train in each iteration. In all our experiments we use a boosting algorithm for MIL called MILBoost [136] as the black box $\mathcal{A}$; additional experiments with the MI-SVM algorithm [6] are available in [14]. Both algorithms show similar trends and we expect the same for any other choice of $\mathcal{A}$. Note that we use IQH only where specified.

5.2.1 Synthetic Data

We begin with a carefully designed synthetic dataset, where we have complete control over the manifold curvature, volume and dimension, and study its effects on the generalization. The details on how we generate the dataset are pro-

vided in [14]; see Figure 5.3 (A) and (B) for examples of the generated manifolds.

For the first set of experiments, we study the interplay between the volume and curvatue while keeping the manifold dimension fixed. Here we generated one-dimensional curves of specified volume (V) and curvature ($1/\tau$) in $\mathbb{R}^2$. We set $\mathbf{h}^*$ to be a vertical hyperplane and labeled the samples accordingly (see [14]). For training, we used 10 positive and 10 negative bags with 500 queried instances per bag (forming a good cover); for testing we used 100 bags. Figure 5.3 (C) shows the test error, averaged over 50 trials, as we vary these parameters. Observe that for a fixed V, as we increase $1/\tau$ (making the manifolds more curvy) generalization error goes up.

For the next set of experiments, we want to understand how manifold dimensionality affects the error. Here we set the ambient dimension to 10 and varied the manifold dimension (with all other experiment settings as before). Figure 5.3 (D) shows how the test error scales for different dimensional bags as we vary the volume ($1/\tau$ set to 1). These results corroborate the general intuition of our analysis, and give an empirical verification that the error indeed scales with the geometric properties of a manifold bag.

5.2.2 Real Data

In this section we present results on image and audio datasets. We will see that the generalization behavior is consistent with our analysis across these different domains. We also study the effects of varying ρ on generalization error, and see how using IQH helps achieve similar error rates with less instances per iteration.

INRIA Heads. For these experiments we chose the task of head detection (e.g., positive bags are images which contain at least one head). We used the INRIA Pedestrian Dataset [38], which contains both pedestrian and non-pedestrian images, to create an INRIA Heads dataset as follows. We manually labeled the location of the head in the pedestrian images. The images were resized such that the size of the head is roughly 24×24 pixels; therefore, instances in this experiment are image patches of that size. For each image patch we computed Haar-like

Figure 5.4: **INRIA Heads:** for our experiments we have labeled the heads in the INRIA Pedestrian Dataset [38]. We can construct bags of different volume by padding the head region. The above figure shows positive bags for two different amounts of padding.

features on various channels as in [43], which corresponds to our instance space $\mathcal{I}$.

Using the ground truth labels, we generated 2472 positive bags by cropping out the head region with different amounts of padding (see Figure 5.4), which corresponds to changing the volume of the manifold bags. For example, padding by 6 pixels results in a bag that is a 30×30 pixel image. To generate negative bags we cropped 2000 random patches from the non-pedestrian images, as well as non-head regions from the pedestrian images. Unless otherwise specified, padding was set to 16.

TIMIT Phonemes. Our other application is in the audio domain, and is analogous to the image data described above. The task here was to detect whether a particular phoneme is spoken in an audio clip (we arbitrarily chose the phoneme "s" to be the positive class). We used the TIMIT dataset [58], which contains recordings of over 600 speakers reading text; the dataset also contains phoneme annotations. Bags in this experiment are audio clips, and instances are audio pieces of length 0.2 seconds (i.e. this is the size of our sliding window). As in the image experiments, we had ground truth annotation for instances, and generated bags of various volumes/lengths by padding. We computed features as follows: we split

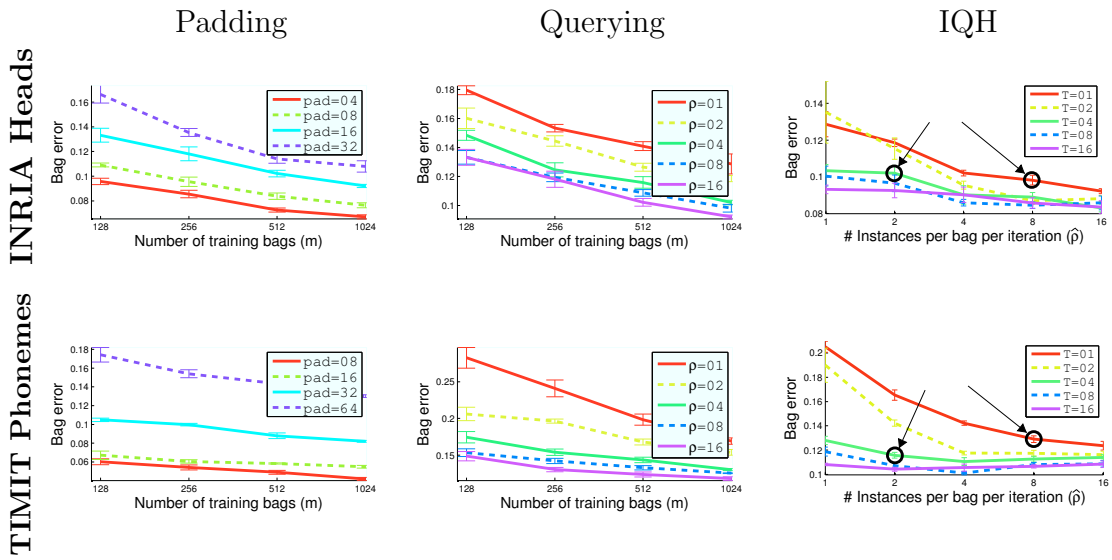


Figure 5.5: **Image and Audio Results:** three different experiments (columns) – varying padding (volume), number of queried instances, and number of IQH iterations – on two different datasets (rows); see text for details. Note that x-axes are in logarithmic scale. All reported results are averages over 5 trials.

each sliding window into 25 millisecond pieces, computed Mel-frequency cepstral coefficients (MFCC) [39, 46] for each piece, and concatenated them to form a 104 dimensional feature vector for each instance. The reported padding amounts are in terms of a 5 millisecond step size (e.g., padding of 8 corresponds to 40 milliseconds of concatenation). Unless otherwise specified, padding was set to 64.

Results. Our first set of experiments involved sweeping over the amount of padding (corresponding to varying the volume of bags). We train with a fixed number of instances per bag, $\rho = 4$. Results for different training set sizes (m) are shown in the first column of Figure 5.5. As observed in the synthetic experiments, we see that increasing the padding (volume) leads to poorer generalization for both datasets. This corroborates our basic intuition that learning becomes more difficult with manifolds of larger volume.

In our second set of experiments, the goal was to see how generalization error is affected by varying the number of queried instances per bag, which compliments Theorem 4. Results are shown in the middle column of Figure 5.5. Observe the interplay between m and ρ: increasing either, while keeping the other fixed, drives

the error down. Recall, however, that increasing m also requires additional labels while querying more instances per bag does not. The number of instances indeed has a significant impact on generalization – for example, in the audio domain, querying more instances per bag can improve the error by up to 15%. As per our analysis, these results suggest that to fully leverage the training data, we must query many instances per bag. Since training with a large number of instances can become computationally prohibitive, this further justifies the Iterative Querying Heuristic (IQH) described in Section 5.1.3.

Our final set of experiments evaluates the proposed IQH method (see Algorithm 1). The number of training bags, m, was fixed to 1024, and the number of candidate instances per iteration, ω, was fixed to 32 for both datasets. Note that $T = 1$ corresponds to querying instances and training MILBoost once (i.e. no iterative querying). Results are shown in the right column of Figure 5.5. These results show that our heuristic works quite well. Consider the highlighed points in both plots: using IQH with $T = 4$ and just 2 instances per bag during training we are able to achieve comparable test error to the naive method (i.e. $T = 1$) with 8 instances per bag. Thus, using IQH, we can obtain a good classifier while needing to use less memory and computational resources per iteration.

5.3 Conclusion

We have presented a new formulation of MIL where bags are manifolds in the instance space, rather than finite sets of instances. This scenario often appears in practice, but has thus far been overlooked in theoretical analysis and algorithm design. We showed that manifold geometry is intimately related to PAC-learnability for this formulation. Our experimental results corroborate the basic intuition of our analysis. Our iterative querying technique enables us to achieve good generalization error while needing to use less memory and computational resources, and should thus be of immediate practical value. We hope that our work encourages further research into leveraging manifold structure in designing MIL algorithms.

Portions of this chapter are based on the following publications:

- "Multiple instance learning with manifold bags" by B. Babenko, N. Varma, P. Dollar, and S. Belongie [14]. The dissertation author developed the algorithm and experiments, and wrote most of the paper.

6 Conclusions & Future Work

In this dissertation we explored a number of weakly supervised methods for training discriminative models and their applications in computer vision. We looked at a wide variety of applications including object detection & recognition, object tracking and image categorization. Finally, we formalized the Multiple Instance Learning framework in a setting that more closely fits such applications in computer vision, and obtained theoretical upper bounds on learning performance.

Although machine learning techniques bring us closer to purely data-driven algorithms, at this point in time some hand tuning and engineering is necessary for each application domain. In computer vision most of the tuning consists of designing features. Thus, one important avenue for future work is to develop features that are general enough to span many application domains.

In the last couple of years, human labor for annotating data has become easier and cheaper to obtain with the advent of crowdsourcing marketplaces (e.g. Amazon Mechanical Turk). While weakly supervised methods will still be important in dealing with ambiguous labeling tasks, methods that directly address the interface between human annotators and learning systems should be explored in the future (e.g. [27]).

Bibliography

[1] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *CVPR*, volume 1, pages 798–805, 2006.

[2] S. Agarwal and D. Roth. Learning a sparse repr. for object det. In *ECCV*, 2002.

[3] S. Ali and M. Shah. Human action recognition in videos using kinematic features and multiple instance learning. *PAMI*, 2008.

[4] Y. Amit and D. Geman. A computational model for visual selection. *Neural Computation*, 11:1691–1715, 1999.

[5] S. Andrews. *Learning from Ambiguous Examples*. PhD thesis, Brown University, 2007.

[6] S. Andrews, T. Hofmann, and I. Tsochantaridis. Multiple instance learning with generalized support vector machines. *A.I.*, pages 943–944, 2002.

[7] M. Anthony and P. Bartlett. *Neural network learning: Theoretical foundations*. Cambridge Univ Pr., 1999.

[8] P. Auer, P. M. Long, and A. Srinivasan. Approximating hyper-rectangles: Learning and pseudo-random sets. In *Proc. of ACM Symposium on Theory of Comp.*, 1997.

[9] S. Avidan. Support vector tracking. *PAMI*, 26(8):1064–1072, 2004.

[10] S. Avidan. Ensemble tracking. In *CVPR*, volume 2, pages 494–501, 2005.

[11] B. Babenko, S. Branson, and S. Belongie. Similarity Metrics for Categorization: from Monolithic to Category Specific. In *International Conference on Computer Vision (ICCV)*, 2009.

[12] B. Babenko, P. Dollár, and S. Belongie. Task Specific Local Region Matching. In *International Conference on Computer Vision (ICCV)*, 2007.

[13] B. Babenko, P. Dollár, Z. Tu, and S. Belongie. Simultaneous Learning and Alignment: Multi-Instance and Multi-Pose Learning. In *Faces in Real-Life Images (ECCV Workshop)*, 2008.

[14] B. Babenko, N. Varma, P. Dollar, and S. Belongie. Multiple instance learning with manifold bags. In *International Conference on Machine Learning (ICML)*, Bellevue, WA, 2011.

[15] B. Babenko, M.-H. Yang, and S. Belongie. Visual Tracking with Online Multiple Instance Learning. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[16] B. Babenko, M.-H. Yang, and S. Belongie. Robust Object Tracking with Online Multiple Instance Learning. *IEEE Transactions of Pattern Analysis and Machine Intelligence (PAMI)*, August 2011.

[17] A. O. Balan and M. J. Black. An adaptive appearance model approach for model-based articulated object tracking. In *CVPR*, volume 1, pages 758–765, 2006.

[18] A. Bar-Hillel, T. Hertz, and D. Weinshall. Object class recognition by boosting a part-based model. In *CVPR*, 2005.

[19] A. Bar-Hillel, T. Hertz, and D. Weinshall. Object class recognition by boosting a part-based model. In *CVPR*, 2005.

[20] N. Ben-Haim, B. Babenko, and S. Belongie. Improving Web-based Image Search via Content Based Clustering. In *SLAM (CVPR Workshop)*, 2006.

[21] J. Bi, Y. Chen, and J. Wang. A sparse support vector machine approach to region-based image categorization. In *CVPR*, 2005.

[22] S. Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *CVPR*, pages 232–237, 1998.

[23] C. Bishop. *Pattern Recognition and Machine Learning*. NJ, USA. Springer, Heidelberg, 2006.

[24] A. Blum and A. Kalai. A Note on Learning from Multiple-Instance Examples. *Mach. Learning*, 30(1):23–29, 1998.

[25] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge Univ. Press, 2004.

[26] K. Branson and S. Belongie. Tracking multiple mouse contours (without too many samples). In *CVPR*, volume 1, 2005.

[27] S. Branson, C. Wah, B. Babenko, F. Schroff, P. Welinder, P. Perona, and S. Belongie. Visual recognition with humans in the loop. In *European Conference on Computer Vision (ECCV)*, Heraklion, Crete, Sept. 2010.

[28] R. Brunelli and T. Poggio. Face recog.: features vs. templates. *PAMI*, 15(10), 1993.

[29] P. Buehler, A. Zisserman, and M. Everingham. Learning sign language by watching TV (using weakly aligned subtitles). In *CVPR*, 2009.

[30] R. Bunescu and R. Mooney. Multiple instance learning for sparse positive bags. In *ICML*, 2007.

[31] A. Chan and N. Vasconcelos. Modeling, clustering, and segmenting video with mixtures of dynamic textures. *PAMI*, 30(5):909–926, 2008.

[32] Y. Chen, J. Bi, and J. Wang. MILES: Multiple-instance learning via embedded instance selection. *PAMI*, 28(12):1931–1947, 2006.

[33] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005.

[34] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 1994.

[35] R. T. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. *PAMI*, 27(10):1631–1643, 2005.

[36] T. Cour, F. Benezit, and J. Shi. Spectral segmentation with multiscale graph decomposition. *CVPR*, 2005.

[37] D. J. Crandall and D. P. Huttenlocher. Weakly supervised learning of part-based spatial models for visual object recognition. In *ECCV*, 2006.

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

[39] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE TASSP*, 1980.

[40] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[41] T. Dietterich, R. Lathrop, and T. Lozano-Perez. Solving the multiple-instance problem with axis parallel rectangles. *A.I.*, 1997.

[42] P. Dollár, B. Babenko, S. Belongie, P. Perona, and Z. Tu. Multiple Component Learning for Object Detection. In *European Conference on Computer Vision (ECCV)*, 2008.

[43] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *BMVC*, 2009.

[44] P. Dollár, Z. Tu, H. Tao, and S. Belongie. Feature mining for image classification. In *CVPR*, 2007.

[45] P. Dollár, P. Welinder, and P. Perona. Cascaded pose regression. In *CVPR*, 2010.

[46] D. Ellis. PLP and RASTA (and MFCC, and inversion) in Matlab. `http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/`, 2005.

[47] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results.

[48] L. Fe-Fei, R. Fergus, and P. Perona. A Bayesian approach to unsupervised one-shot learning of object categories. In *ICCV*, 2003.

[49] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object Detection with Discriminatively Trained Part-Based Models. *PAMI*, 2009.

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

[51] M. Fischler and R. Elschlager. The Representation and Matching of Pictorial Structures. *IEEE Transactions on Computers*, 100(22):67–92, 1973.

[52] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Comp. and Sys. Sci.*, 55:119–139, 1997.

[53] J. Friedman. Greedy function approximation: A gradient boosting machine. *Ann. of Stat.*, 29(5):1189–1232, 2001.

[54] J. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38:367–378, 2002.

[55] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. Technical report, Stanford Univ., 1998.

[56] A. Frome, Y. Singer, F. Sha, and J. Malik. Learning Globally-Consistent Local Distance Functions for Shape-Based Image Retrieval and Classification. In *ICCV*, 2007.

[57] C. Galleguillos, B. Babenko, A. Rabinovich, and S. Belongie. Weakly Supervised Object Recognition and Localization with Stable Segmentations. In *ECCV*, 2008.

[58] J. Garofolo et al. TIMIT Acoustic-Phonetic Continuous Speech Corpus. `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S1`, 1993.

[59] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *NIPS*, 2005.

[60] H. Grabner and H. Bischof. On-line boosting and vision. In *CVPR*, pages 260–267, Washington, DC, USA, 2006.

[61] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *BMVC*, pages 47–56, 2006.

[62] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *ECCV*, 2008.

[63] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, CalTech, 2007.

[64] G. Griffin and P. Perona. Learning and Using Taxonomies For Fast Visual Categorization. In *CVPR*, 2008.

[65] C. Huang, H. Ai, Y. Li, and S. Lao. Vector boosting for rotation invariant multi-view face detection. In *ICCV*, 2005.

[66] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusets, Amherst, 2007.

[67] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. *ECCV*, 1064:343–356, 1996.

[68] M. Isard and J. Maccormick. Bramble: a bayesian multiple-blob tracker. In *ICCV*, volume 2, pages 34–41, 2001.

[69] M. Jones and P. Viola. Face Recognition Using Boosted Local Features. Technical Report TR2003-25, MERL, 2003.

[70] F. Jurie and B. Triggs. Creating efficient codebooks for vis. recog. In *ICCV*, 2005.

[71] J. Keeler, D. Rumelhart, and W. Leow. Integrated segmentation and recognition of hand-printed numerals. In *NIPS*, 1990.

[72] Z. Khan, T. Balch, and F. Dellaert. A rao-blackwellized particle filter for eigentracking. In *CVPR*, volume 2, 2004.

[73] T. Lange, V. Roth, M. Braun, and J. Buhmann. Stability-based validation of clustering solutions. *Neural Computation*, 16(6):1299–1323, 2004.

[74] S. Lazebnik, C. Schmid, and J. Ponce. Sparse texture representation using affine-invariant neighborhoods. In *CVPR*, 2003.

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

[76] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio. Gradient-based learning applied to document recognition. In *IEEE*, 1998.

[77] B. Leibe, A. Leonardis, and B. Schiele. Robust Object Detection with Interleaved Categorization and Segmentation. *IJCV*, pages 1–31, 2005.

[78] B. Leibe and B. Schiele. Analyzing Appearance and Contour Based Methods for Object Categorization. In *CVPR*, 2003.

[79] C. Leistner, A. Saffari, P. Roth, and H. Bischof. On robustness of on-line boosting-a competitive study. In *3rd IEEE ICCV Workshop on On-line Computer Vision*, 2009.

[80] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *PAMI*, 28(9):1465, 2006.

[81] T. Leung, M. Burl, and P. Perona. Finding faces in cluttered scenes using random labeled graphmatching. In *ICCV*, pages 637–644, 1995.

[82] R. Lin, D. Ross, J. Lim, and M.-H. Yang. Adaptive Discriminative Generative Model and Its Applications. In *NIPS*, pages 801–808, 2004.

[83] Y. Lin, T. Liu, C. Fuh, and T. Sinica. Local ensemble kernel learning for object category recognition. In *CVPR*, 2007.

[84] X. Liu and T. Yu. Gradient feature selection for online boosting. In *ICCV*, pages 1–8, 2007.

[85] P. Long and L. Tan. PAC Learning Axis-aligned Rectangles with Respect to Product Distributions from Multiple-Instance Examples. *Machine Learning*, 1998.

[86] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 60(2):91–110, 2004.

[87] J. Malik, S. Belongie, J. Shi, and T. Leung. Textons, contours and regions: Cue integration in image segmentation. In *ICCV*, 1999.

[88] T. Malisiewicz and A. Efros. Improving spatial support for objects via multiple segmentations. *BMVC*, 2007.

[89] M. Mandel and D. Ellis. Multiple-instance learning for music information retrieval. In *ISMIR*, 2008.

[90] O. Maron. *Learning from Ambiguity*. PhD thesis, Massachusetts Institute of Technology, 1998.

[91] O. Maron and T. Lozano-Perez. A framework for multiple-instance learning. In *NIPS*, 1998.

[92] K. Mikolajczyk, C. Schmid, and A. Zisserman. Human Detection Based on a Probabilistic Assembly of Robust Part Detectors. In *ECCV*, 2004.

[93] A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *PAMI*, 23(4):349–361, 2001.

[94] F. Moosmann, B. Triggs, and F. Jurie. Fast discriminative visual codebooks using randomized clustering forests. In *NIPS*, 2007.

[95] G. Mori and J. Malik. Recovering 3d human body configurations using shape contexts. *PAMI*, 28(7):1052–1062, 2006.

[96] L. Mu, J. Kwok, and L. Bao-liang. Online Multiple Instance Learning with No Regret. In *CVPR*, 2010.

[97] M. Nguyen, L. Torresani, F. de la Torre, and C. Rother. Weakly supervised discriminative localization and classification: a joint learning process. In *ICCV*, 2009.

[98] M. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In *CVPR*, 2006.

[99] P. Niyogi, S. Smale, and S. Weinberger. Finding the homology of submanifolds with high confidence from random samples. *Disc. Computational Geometry*, 2006.

[100] K. Okuma, A. Taleghani, N. De Freitas, J. Little, and D. Lowe. A boosted particle filter: Multitarget detection and tracking. *ECCV*, pages 28–39, 2004.

[101] A. Opelt, M. Fussenegger, and P. Auer. Generic object recognition with boosting. *PAMI*, 28(3):416–431, 2006.

[102] A. Opelt, A. Pinz, and A. Zisserman. Incremental learning of object detectors using a visual shape alphabet. In *CVPR*, 2006.

[103] N. C. Oza. Online Ensemble Learning. *Ph.D. Thesis, University of California, Berkeley*, 2001.

[104] G. Qi, X. Hua, Y. Rui, T. Mei, J. Tang, and H. Zhang. Concurrent Multiple Instance Learning for Image Categorization. In *CVPR*, 2007.

[105] A. Rabinovich, T. Lange, J. Buhmann, and S. Belongie. Model order selection and cue combination for image segmentation. In *CVPR*, 2006.

[106] A. Rabinovich, A. Vedaldi, and S. Belongie. Does image segmentation improve object categorization? *UCSD Technical Report CSE CS2007-0908*, 2007.

[107] J. Ramon and L. De Raedt. Multi instance neural networks. *ICML, Workshop on Attribute-Value and Relational Learning*, 2000.

[108] S. Ray and M. Craven. Supervised versus multiple instance learning: an empirical comparison. *ICML*, 2005.

[109] R. Rifkin and A. Klautau. In Defense of One-Vs-All Classification. *JMLR*, 5:101–141, 2004.

[110] D. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *IJCV*, 77(1):125–141, 2008.

[111] B. Russell, A. Efros, J. Sivic, W. Freeman, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *CVPR*, 2006.

[112] S. Sabato, N. Srebro, and N. Tishby. Reducing Label Complexity by Learning From Bags. In *AISTATS*, 2010.

[113] S. Sabato and N. Tishby. Homogeneous multi-instance learning with arbitrary dependence. In *COLT*, 2009.

[114] P. Sabzmeydani and G. Mori. Det. peds. by learning shapelet ftrs. In *CVPR*, 2007.

[115] M. Salzmann, V. Lepetit, and P. Fua. Deformable surface tracking ambiguities. In *CVPR*, 2007.

[116] L. Saul, M. Rahim, and J. Allen. A statistical model for robust integration of narrowband cues in speech. *Comp. Speech and Language*, 15:175–194, 2001.

[117] E. Seemann, B. Leibe, and B. Schiele. Multi-Aspect Detection of Articulated Objects. In *CVPR*, 2006.

[118] G. Shakhnarovich. *Learning Task-Specific Similarity*. PhD thesis, Massachusetts Institute of Technology, 2006.

[119] G. Shakhnarovich, T. Darrell, and P. Indyk. *Nearest-neighbor methods in learning and vision: Theory and practice*. MIT Press, 2005.

[120] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000.

[121] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering object categories in image collections. In *CVPR*, 2005.

[122] S. Stalder, H. Grabner, and L. van Gool. Beyond Semi-Supervised Tracking: Tracking Should Be as Simple as Detection, but not Simpler than Recognition. In *Online Learning in Computer Vision (OLCV) Workshop*, 2009.

[123] M. Stikic and B. Schiele. Activity recognition from sparsely labeled data using multi-instance learning. In *Location and Context Awareness*, 2009.

[124] S. Todorovic and N. Ahuja. Extracting subimages of an unknown category from a set of images. In *CVPR*, 2006.

[125] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large databases for recognition. In *CVPR*, 2008.

[126] A. Torralba, K. Murphy, and W. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *CVPR*, volume 2, 2004.

[127] Z. Tu. Probabilistic Boosting-Tree: Learning Discriminative Models for Classification, Recognition, and Clustering. In *ICCV*, 2005.

[128] O. Tuzel, F. Porikli, and P. Meer. Human Detection via Classification on Riemannian Manifolds. In *CVPR*, 2007.

[129] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

[130] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Prob. and Its App.*, 1971.

[131] M. Varma and D. Ray. Learning The Discriminative Power-Invariance Trade-Off. In *ICCV*, 2007.

[132] L. Vese and T. Chan. A multiphase level set framework for image segmentation using the Mumford and Shah model. *IJCV*, 50(3):271–293, 2002.

[133] M. Vidal-Naquet and S. Ullman. Object recognition with informative features and linear classification. In *ICCV*, 2003.

[134] S. Vijayanarasimhan and K. Grauman. Keywords to Visual Categories: Multiple-Instance Learning for Weakly Supervised Object Categorization. In *CVPR*, 2008.

[135] P. Viola and M. Jones. Fast multi-view face detection. In *CVPR*, 2001.

[136] P. Viola, J. Platt, and C. Zhang. Multiple instance boosting for object detection. In *NIPS*, 2005.

[137] G. Wang, Y. Zhang, and L. Fei-Fei. Using dependent regions for object categorization in a generative framework. In *CVPR*, 2006.

[138] J. Wang, X. Chen, and W. Gao. Online selecting discriminative tracking features using particle filter. In *CVPR*, volume 2, pages 1037–1042, 2005.

[139] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *ECCV*, 2000.

[140] K. Q. Weinberger, J. Blitzer, and L. K. Saul. Fast Solvers and Efficient Implementations for Distance Metric Learning. In *ICML*, 2008.

[141] B. Wu and R. Nevatia. Cluster Boosted Tree Classifier for Multi-View, Multi-Pose Object Detection. In *ICCV*, 2007.

[142] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *NIPS*, 2003.

[143] G. Yang and T. Huang. Human face detection in a complex background. *Pattern recognition*, 27(1):53–63, 1994.

[144] T. Yeh and T. Darrell. Dynamic visual category learning. In *CVPR*, 2008.

[145] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys (CSUR)*, 38(4), 2006.

[146] C. Yu and T. Joachims. Learning structural svms with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1169–1176. ACM, 2009.

[147] C. Zhang and P. Viola. Multiple-instance pruning for learning efficient cascade detectors. In *NIPS*, 2007.

[148] Q. Zhang and S. Goldman. EM-DD: An improved multiple-instance learning technique. In *NIPS*, 2002.