Copyright
by
Sudheendra Narasimhan Vijayanarasimhan
2011

The Dissertation Committee for Sudheendra Narasimhan Vijayanarasimhan
certifies that this is the approved version of the following dissertation:

Active Visual Category Learning

Committee:

Kristen Grauman, Supervisor

J. K. Aggarwal

Inderjit Dhillon

Raymond Mooney

Antonio Torralba

Active Visual Category Learning

by

Sudheendra Narasimhan Vijayanarasimhan, B.Tech., M.S.C.S

DISSERTATION
Presented to the Faculty of the Graduate School of
The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN
May 2011

Dedicated to my parents…

Active Visual Category Learning

Publication No. ____________________

Sudheendra Narasimhan Vijayanarasimhan, Ph.D.
The University of Texas at Austin, 2011

Supervisor: Kristen Grauman

Visual recognition research develops algorithms and representations to autonomously recognize visual entities such as objects, actions, and attributes. The traditional protocol involves manually collecting training image examples, annotating them in specific ways, and then learning models to explain the annotated examples. However, this is a rather limited way to transfer human knowledge to visual recognition systems, particularly considering the immense number of visual concepts that are to be learned.

I propose new forms of active learning that facilitate large-scale transfer of human knowledge to visual recognition systems in a cost-effective way. The approach is cost-effective in the sense that the division of labor between the machine learner and the human annotators respects any cues regarding which annotations would be easy (or hard) for either party to provide. The approach is large-scale in that it can deal with a large number of annotation types, multiple human annotators, and huge pools of unlabeled data. In particular, I consider three important aspects of the problem:

(1) cost-sensitive multi-level active learning, where the expected informativeness of any candidate image annotation is weighed against the predicted cost of obtaining it in order to choose the best annotation at every iteration. (2) budgeted batch active learning, a novel active learning setting that perfectly suits automatic learning from crowd-sourcing services where there are multiple annotators and each annotation task may vary in difficulty. (3) sub-linear time active learning, where one needs to retrieve those points that are most informative to a classifier in time that is sub-linear in the number of unlabeled examples, i.e., without having to exhaustively scan the entire collection.

Using the proposed solutions for each aspect, I then demonstrate a complete end-to-end active learning system for scalable, autonomous, online learning of object detectors. The approach provides state-of-the-art recognition and detection results, while using minimal total manual effort. Overall, my work enables recognition systems that continuously improve their knowledge of the world by learning to ask the right questions of human supervisors.

Table of Contents

Abstract **v**

List of Figures **xii**

List of Tables **xv**

Chapter 1. Introduction **1**

1.1 Overview of Thesis 6

1.1.1 Cost-sensitive Active Learning with Multi-level Queries 6

1.1.2 Selection under a Budget for Multiple Simultaneous Annotators 9

1.1.3 Sub-linear Time Active Learning for Web-scale Data 11

1.1.4 A Large-scale System for Autonomous Online Visual Learning 13

1.2 Main Contributions 14

1.3 Road Map 15

Chapter 2. Related Work **17**

2.1 Reducing the Level of Supervision 17

2.1.1 Weakly Supervised Recognition 19

2.1.2 Unsupervised Recognition 22

2.1.3 Alternative Sources of Training Data 23

2.2 Reducing the Amount of Supervision 25

2.2.1 Semi-supervised Visual Learning 25

- 2.2.2 Active Learning . . . 26
- 2.3 Reducing the Effort of Providing Supervision . . . 31
  - 2.3.1 Labeling Services . . . 32
  - 2.3.2 Image Annotation Games . . . 33
  - 2.3.3 Interactive Segmentation . . . 34

Chapter 3. Cost-sensitive Active Learning with Multi-level Queries 36

- 3.1 Multi-level Active Prediction of Useful Image Annotations for Recognition . . 39
  - 3.1.1 Multiple-instance Multi-label Learning . . . 41
    - 3.1.1.1 Multiple-instance Learning . . . 41
    - 3.1.1.2 Multiple-instance Multi-label Learning . . . 43
  - 3.1.2 Active Multi-level Selection of Multi-label Annotations . . . 49
    - 3.1.2.1 Illustration: Need for Comparing Multiple Types of Annotations 50
    - 3.1.2.2 Defining the Value of Information for an Annotation . . . 51
    - 3.1.2.3 Predicting the Informativeness of an Annotation . . . 54
    - 3.1.2.4 Predicting the Cost of an Annotation . . . 57
  - 3.1.3 Summary of the Algorithm . . . 62
- 3.2 Results . . . 63
  - 3.2.1 Datasets and Implementation Details . . . 63
  - 3.2.2 Multi-label Visual Category Learning with the MSK . . . 66
  - 3.2.3 Multi-level Active Selection for Learning Object Categories . . . 68
    - 3.2.3.1 Active Selection from MIL Data . . . 69
    - 3.2.3.2 Active Selection from MIML Data . . . 76
    - 3.2.3.3 Active Selection Examples . . . 80
    - 3.2.3.4 Effect of Initial Training Set Size . . . 81

3.2.4 Annotation Costs and Active Selection 85
    3.2.4.1 Annotation Cost Prediction 85
    3.2.4.2 Active Selection with a Learned Cost Function 88
3.2.5 Computation Time 90
3.3 Discussion 91

Chapter 4. Selection under a Budget for Multiple Simultaneous Annotators 94

4.1 Budgeted Batch Active Learning (BBAL) 96
    4.1.1 Problem Definition and Overview 96
    4.1.2 Formulation and Algorithm 99
    4.1.3 Summary: Using BBAL 105
4.2 Results 106
    4.2.1 Datasets and Implementation Details 107
    4.2.2 Learning Activities on a Budget 110
    4.2.3 Learning Objects on a Budget 111
    4.2.4 Effect of the Budget Size 113
    4.2.5 Comparison to State-of-the-Art Batch Selection 114
    4.2.6 Impact of Budgets versus Fixed-Size Batches 116
    4.2.7 Computation Time 117
4.3 Discussion 117

Chapter 5. Sub-linear Time Active Selection for Web-scale Data 120

5.1 Hashing Hyperplanes to Near Points 123
    5.1.1 Background: Locality-Sensitive Hashing 124
    5.1.2 Hyperplane Hashing based on Angle Distance (H-Hash) 125
        5.1.2.1 Proof of Locality-sensitivity for Hyperplane Hashing 127

5.1.2.2 Approximation Guarantees for Hyperplane Hashing 128
5.1.2.3 Improved Hyperplane Hashing based on Angle Distance (IH-Hash) 131
5.1.3 Embedded Hyperplane Hashing based on Euclidean Distance (EH-Hash) 132
5.1.4 Comparison of Approximation Guarantees 138
5.1.5 Recap of the Hashing Approaches 139
5.1.6 Application to Large-scale Active Learning 140
5.2 Results 141
5.2.1 Datasets and Implementation Details 141
5.2.2 Document Classification: Newsgroups Results 142
5.2.3 Object Recognition: CIFAR-10 Results 143
5.2.4 Minimizing both Selection and Labeling Times 144
5.2.5 Active Section from 1 Million Images 146
5.2.6 Comparison with the 59-trick Heuristic 147
5.3 Discussion 149

Chapter 6. A Large-scale System for Autonomous Online Visual Learning 152
6.1 Live Learning of Object Detectors 154
6.1.1 Object Representation and Linear Classifier 155
6.1.1.1 Window Descriptions 156
6.1.1.2 Feature Encoding 157
6.1.1.3 Relationship to Existing Detection Models 159
6.1.2 Generating Candidate Root Windows 160
6.1.3 Active Selection of Object Windows 161
6.1.4 Online Annotation Requests 164
6.1.5 Training the Detector 165

6.2 Results . . . 165
  6.2.1 Datasets and Implementation details . . . 167
  6.2.2 Comparison to State-of-the-Art Detectors . . . 168
  6.2.3 Active Detector Training on PASCAL . . . 171
  6.2.4 Online Live Learning on Flickr . . . 173
    6.2.4.1 Live Learning Applied to PASCAL Test Set . . . 174
    6.2.4.2 Live Learning Applied to Flickr Test Set . . . 175
  6.2.5 Computation Time . . . 177
6.3 Discussion . . . 178

Chapter 7. Future Work **181**

Chapter 8. Conclusion **183**

Bibliography **187**

Vita **205**

List of Figures

1.1 Traditional protocol vs. Proposed active framework 2
1.2 Cost-sensitive active learning 7
1.3 Budgeted batch active learning 9
1.4 Sub-linear time active learning 12

2.1 Different kinds of supervision on image data 18
2.2 Active learning example 27
2.3 Web annotation tool 33

3.1 Three important distinctions while choosing informative image data 37
3.2 Overview of multi-level active learning 39
3.3 Example scenarios where MIL is suitable for image classification 42
3.4 Our multi-label multiple-instance learning (MIML) scenario 44
3.5 Multi-label set kernel (MSK) 46
3.6 The three candidate annotation types (or “levels”) our approach chooses from 49
3.7 Illustration: need for comparing multiple types of annotations 51
3.8 Which image would you rather annotate? 58
3.9 Our interface on Mechanical Turk to collect annotation times 60
3.10 Summary of our multi-level active learning approach. 61
3.11 Example images from the SIVAL dataset 64
3.12 Example images from Google downloaded dataset. 64
3.13 Example images from the MSRC dataset 66

3.14 Sample learning curves on SIVAL dataset 71
3.15 Summary of average improvement over all 25 SIVAL categories 72
3.16 Multi-level results on the Google dataset 75
3.17 Active selection from MIML data 78
3.18 Annotation queries selected by our method in subsequent iterations 79
3.19 Cumulative number of labels acquired for each annotation type 80
3.20 Effect of the initial training set size on MSRC 82
3.21 Effect of the initial training set size SIVAL 83
3.22 Accuracy of our cost function in predicting "easy" vs. "hard" images 85
3.23 Easiest and hardest images to annotated predicted by our function 86
3.24 Predicted times vs. actual times 88
3.25 Active selection with a learned cost function 89

4.1 The budgeted batch active learning problem 97
4.2 Budgeted selection on Hollywood dataset 109
4.3 Budgeted selection on SIVAL dataset 111
4.4 Example batch selections 112
4.5 Effect of budget size 114
4.6 Comparison to fixed batch approach with variable cost data 116

5.1 Intuition behind hyperplane hashing (H-Hash) 126
5.2 Intuition behind H-Hash solution 127
5.3 Intuition behind embedded hyperplane hashing (EH-Hash) 133
5.4 Comparison of probability of collision for H-Hash and EH-Hash 135
5.5 Comparison of ρ for EH-Hash and H-Hash 139
5.6 Summary of sub-linear time active selection 140

5.7 Sub-linear time selection on Newsgroups 142
5.8 Sub-linear time selection on CIFAR-10 143
5.9 First 15 examples selected for labeling on CIFAR-10 Airplane 144
5.10 Minimizing both selection and labeling times 145
5.11 Sub-linear time active learning on 1 Million examples 147
5.12 Comparison with the 59-trick heuristic 148

6.1 Proposed part-based object representation. 156
6.2 Contrasts with related object detection models 158
6.3 Illustration of jumping window root candidates 161
6.4 MTurk interface to obtain bboxes on actively selected examples. 164
6.5 Summary of large-scale system for live learning 166
6.6 Example images from PASCAL VOC 2007 and Flickr 167
6.7 Example detections on the PASCAL dataset obtained by our detector 170
6.8 Active detector training on PASCAL. 172
6.9 Live learning results on PASCAL test set 173
6.10 Live learning results on Flickr test set 176
6.11 Images selected by our live approach 177

List of Tables

3.1 Comparison with state-of-the-art MIML approach 67

3.2 Region-level multi-class classification accuracies compared to state-of-the-art approaches 67

3.3 Savings in cost when using cost prediction 90

4.1 Distribution of manual effort costs on SIVAL and HOHA 107

4.2 Comparison to fixed batch approach on Corel 115

6.1 Statistics of Flickr dataset 166

6.2 Comparison of object detector with state-of-the-art 169

6.3 Our best results on PASCAL VOC 2007 175

6.4 Overlap scores of the windows selected by our approach 177

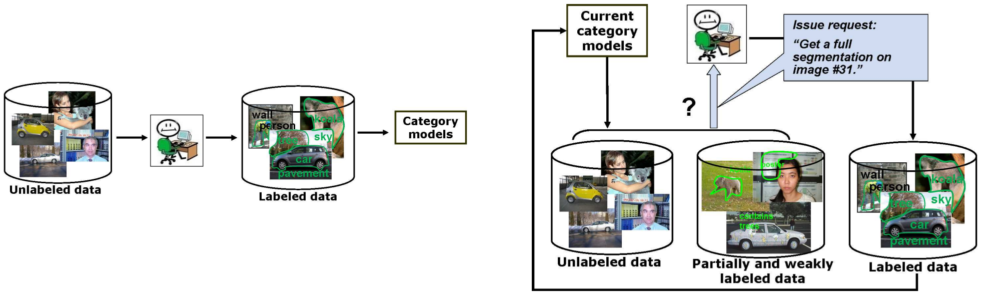
6.5 Run-time comparisons of our approach with state-of-the-art 178

Chapter 1

Introduction

Computer vision research is aimed at developing algorithms and representations that will enable a computer system to autonomously analyze visual information. One of the primary challenges in this research is the problem of recognizing generic object categories. It is challenging on a number of levels: objects of the same class may exhibit wide variability in appearance, real-world images naturally contain large amounts of irrelevant background “clutter”, and subtle context cues can in many cases be crucial to proper perception of objects. Nonetheless, recent advances have shown the feasibility of using statistical machine learning techniques to build accurate models for a number of well-defined object categories.

However, some form of human supervision is required to train such models. Figure 1.1(a) illustrates the traditional protocol for providing supervision for object recognition systems. Training data containing a large number of examples of the categories to be learned are first manually collected. Depending on the requirement of the model being learned, the collected examples are annotated in specific ways. This may range from tagging images with category labels to outlining object boundaries to marking landmark points on objects and cropping, aligning and normalizing the pose of objects. Unfortunately, the accuracy of most current approaches relies heavily on the amount of such labeled training examples available for each



(a) The current protocol for learning classifiers. Training data is manually collected, and annotated to train a classifier.

(b) The active learning protocol for learning classifiers. Training data is manually or automatically collected, and the system repeatedly chooses examples to request annotations from a human annotator.

Figure 1.1: Contrast between the traditional protocol of training classifiers for visual recognition (left) and the proposed active learning framework (right).

class of interest, which effectively restricts existing results to relatively few categories of objects (often on the order of 10s).

Considering that recognition research aims at learning visual representations of about 30,000 nouns (objects) notwithstanding the innumerable verbs (actions) and adjectives (attributes), the standard protocol of learning models from carefully gathered and annotated images is unsuitable as a means of transferring human knowledge. In particular,

- Collecting such hand-crafted training examples is an expensive endeavor in terms of manual effort. While recent work in vision has considered reducing this reliance on supervision, there is no direct measure of the amount of manual effort that is being expended.

- Allowing a human to select examples for training a system will inadvertently introduce biases in the collected data. In addition, there is an obvious disconnect between what a human considers as "useful" and what a vision system will consider as "useful".

- Data collection is treated as a one-time preprocessing stage. However, realistically any learning system must be expected to continuously perceive its environment and identify concepts that it finds unclear in order to expand its horizon of knowledge.

The vision community is well aware of the challenge of reducing the reliance of object recognition methods on well-annotated datasets. Recent methods have considered reducing the "level" of supervision by using weakly labeled images which are easier to obtain [119, 32, 73, 125, 115], re-using knowledge across categories [94, 40] and leveraging the free but noisy tagged images on the Web [31, 70, 106]. In an effort to reduce the "amount" of supervision, approaches have been devised for labeling and utilizing unlabeled examples effectively through active learning strategies [18, 75, 123, 54, 77, 48, 52, 21] or semi-supervised methods [16, 40, 43, 68]. Working in the other direction, some research seeks to ease the "effort" required to provide supervision by tempting users with games or nice datasets [116, 83] or compensation [96]. [1]

While the results are encouraging, existing techniques fail to address the following key insights about cost-effective large-scale transfer of human knowledge:

- The division of labor between the machine learner and the human labelers ought to respect any cues regarding which annotations would be easy (or hard) for either party to provide. This means that there should be a definitive notion of both how *informative*

---

[1] I will discuss these methods in more detail and provide contrasts with my approach in Chapter 2.

a particular annotation is to the learner and how much *effort* it requires from a human supervisor. Also, to use a fixed amount of manual effort most effectively may call for a combination of annotations at multiple supervision levels (e.g., a full segmentation on some images and a present/absent flag on others).

- With the availability of multiple human annotators working simultaneously the machine learner must be able to distribute the labor across different annotators in the most effective way. This means that the learning system must be able to generate large batches of annotation queries simultaneously that not only minimize the total annotation cost but also provide as much non-overlapping information as possible.
- Finally, since obtaining annotations from a human is an interactive process, any large-scale learning system must be able to quickly generate annotation questions without wasting the annotators' time. Therefore, the learner must also be able to pinpoint the most informative label requests among a large pool of unlabeled examples without having to exhaustively scan the entire dataset.

My thesis “Active Visual Category Learning” successfully incorporates these insights in order to provide a cost-effective approach for learning visual concepts such as object categories. Figure 1.1(b) illustrates the proposed active learning protocol where the learned model itself selects the most informative examples to obtain labels for from a human annotator. While traditional active learning has been shown to reduce the total number of labels to train classifiers, there are several distinct unsolved problems when choosing informative data to annotate for visual recognition. Through my thesis work I address these important problems:

- How can a classifier learn from annotations at multiple levels of granularity? (Chapter 3)
- How do we define a principled way to compare every candidate annotation both in terms of how much effort it might require and how much information it might provide? (Chapter 3)
- Can we predict an image's annotation costs directly from its features so as to compute its true worth? (Chapter 3)
- How do we utilize multiple annotators at once so that the total annotation cost is reduced? (Chapter 4)
- Can we design active selection methods that do not need to exhaustively scan all available annotations in order to pick a question to ask? (Chapter 5)

Solutions to these problems fundamentally enhance active learning as a cost-effective large-scale means for learning various visual concepts. I demonstrate this by developing the first system capable of scalable, automatic online learning of object detectors with crowd-sourced labels (Chapter 6).

Through extensive experiments on several challenging datasets I demonstrate that my approach can learn visual classifiers with a lower expenditure of manual effort when compared to the state-of-the-art and several relevant baseline techniques. In particular, I demonstrate a large-scale system for autonomous online learning that improves the state-of-the-art for the most difficult objects in the PASCAL dataset, a widely accepted challenging benchmark for object detection, thus showing that the proposed framework is a more effective protocol for learning visual models.

1.1 Overview of Thesis

This section previews the main components of my thesis, to provide the reader a compact summary of the chapters ahead. In the following subsections, I provide a summary of the problems I have addressed, expanding the themes introduced above with a bit more detail. The following chapters provide technical ideas and results for each component.

1.1.1 Cost-sensitive Active Learning with Multi-level Queries

Traditional active learning methods design selection criteria for choosing an unlabeled example on which to request a *single* type of label from an oracle (e.g., which category the example belongs to, or whether a particular object exists in the image example so as to improve the category model [124, 54, 21, 77, 107]). However, to use a fixed amount of manual effort resources most effectively the category learner must be allowed to choose from a mixture of annotations at different levels of granularity (strongly labeled and weakly labeled) depending on how confidently the model can explain the example. This is because strongly labeled examples are less ambiguous but more expensive in terms of manual effort, while weakly labeled examples are more ambiguous in their labels but easier to obtain. Take for example, a full segmentation on images as opposed to a present/absent flag. It might be easier to provide a present/absent flag on an image compared to object outlines; however, a full segmentation provides unambiguous information to the classifier. Similarly, the longer the video, the longer it will take to watch it and annotate its contents; the more sophisticated or time-consuming the image query, the more we may need to pay a human labeler to answer it.

The learning algorithm must therefore be able to accommodate the multiple levels of gran-

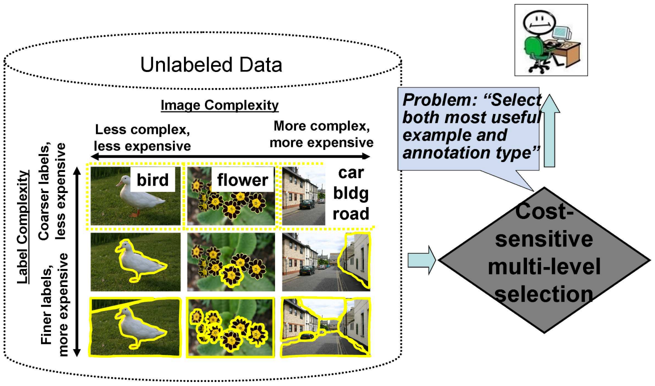

Unlabeled Data
Image Complexity
Less complex, less expensive
More complex, more expensive
Label Complexity
Coarser labels, less expensive
Finer labels, more expensive
bird
flower
car
bldg
road
Problem: "Select both most useful example and annotation type"
Cost-sensitive multi-level selection

Figure 1.2: The problem setting of our cost-sensitive active learning approach. Useful image annotations can occur at multiple levels of granularity, and each individual image example can require different amounts of manual effort. The main challenge is to design an active selection function that can compare annotations based on both how informative they are and how much manual effort they require.

ularity that may occur in image annotations, and to compute which item *at which of those levels* appears to be most fruitful to have labeled next. Additionally, while most approaches assume that all unlabeled examples require the same annotation cost, the actual manual effort required to provide an annotation varies both according to the annotation type as well as the particular image example. Hence, an active learner must not only incorporate the variable cost of an annotation into its selection function, but it should also be able to predict the cost of an image example even before obtaining it. Finally, most real-world images consist of multiple objects, and so should be associated with *multiple* labels simultaneously. Note that multi-label is thus more general than *multi-class*, where usually each example is assumed to represent an item from a single class. Therefore, ideally an active learner must be able to both learn from and assess the value of images containing some unknown combination of categories. Figure 1.2 illustrates the problem setting.

In order to handle these issues, I consider an active learning framework where the expected informativeness of any candidate image annotation is weighed against the predicted cost of obtaining it. I devise a *multiple-instance, multi-label learning (MIML)* formulation that allows the system itself to choose which annotations to receive, based on the expected benefit to its current object models. After learning from a small initial set of labeled images, my method surveys any available unlabeled data to choose the most promising annotation to receive next. After re-training, the process repeats, continually improving the models with minimal manual intervention.

Critically, our active learner chooses both which image example as well as what *type* of annotation to request: a complete image segmentation, a segmentation of a single object, or an image-level category label naming one of the objects within it. Furthermore, since any request can require a different amount of manual effort to fulfill, we explicitly balance the value of a new annotation against the time it might take to receive it. Even for the same type of annotation, some images are faster to annotate than others (e.g., a complicated scene vs. an image with few objects). Humans can easily glance at an image and roughly gauge the difficulty. But can we predict annotation costs directly from image features? Learning with data collected from annotators on the Web, we show that active selection gains actually improve when we account for the task's variable difficulty.

Our results demonstrate that (1) the active learner obtains accurate models with much less manual effort than typical passive learners, (2) we can fairly reliably estimate how much a putative annotation will cost given the image content alone, and (3) our multi-label, multi-level strategy outperforms conventional active methods that are restricted to requesting a single type of annotation.

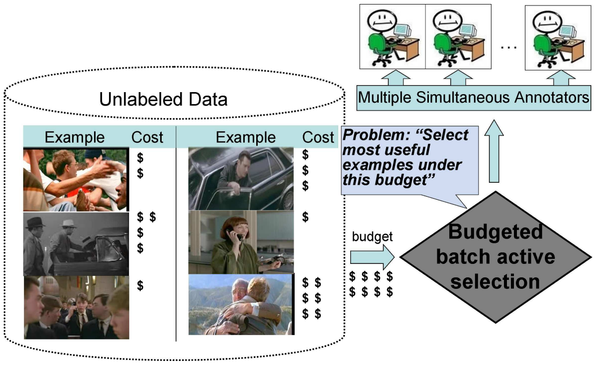

Unlabeled Data
Example
Cost
Example
Cost
Multiple Simultaneous Annotators
Problem: "Select most useful examples under this budget"
budget
$ $ $ $
$ $ $ $
Budgeted batch active selection

Figure 1.3: The problem setting for budgeted batch active selection. With the availability of multiple simultaneous annotators it is often preferable to farm out a batch of good queries at once. Since examples have variable annotation costs, the main challenge is to select the most useful set of examples without overspending the given annotation budget.

1.1.2 Selection under a Budget for Multiple Simultaneous Annotators

The previous section dealt with choosing a *single* annotation from a large pool of multiple types of annotation queries. Such an approach can be used to train classifiers when a single human annotator is available to interact with the system. However, in many applications multiple annotators are available simultaneously (e.g., Mechanical Turk[96], LabelMe[83]). An active learning system that needs to repeatedly go offline and compute the next annotation request cannot take advantage of such resources. Therefore, it may be preferable to farm out a *batch* of good queries at once. There are a few recent active learning methods that try to select a *fixed* batch of examples at each iteration [12, 57, 42]. However, such techniques typically assume that all examples require the same amount of manual effort to label, and thus aim to minimize the total number of queries made. In reality, the cost associated with labeling different examples often varies, sometimes significantly, discussed above.

Therefore, I formalize the problem of far-sighted active learning with a budget. Figure 1.3 shows the problem setting of this approach. At each iteration the active learner is allowed to choose a set of examples to get labeled, provided the total sum of costs associated with the selected examples is under a given budget. I propose a novel method for optimally selecting a set of examples for a support vector machine (SVM) classifier under these conditions. Given a large unlabeled pool of data where each example has an associated cost, we introduce a set of instance selection variables. I formulate an optimization problem to learn the maximum margin hyperplane along with the instance variables that minimize the empirical risk (on both the labeled data and selected unlabeled points), while satisfying the given budget constraint. We then relax it to a continuous optimization problem that can be decomposed into two strictly convex optimization problems loosely coupled in the hyperplane parameters and selection variables. We devise a monotonically convergent alternating minimization algorithm to compute the solution.

To my knowledge, the proposed approach is the first batch active selection strategy that is sensitive to the costs of labeling, and the first method to allow sets of training examples to be chosen so as to meet a prescribed budget. The efficiency of the component optimization steps also makes it rather scalable to large unlabeled data pools. Furthermore, in contrast to previous methods, our approach considers how much the classifier objective changes if we were to obtain the most probable labels on the candidate examples for selection. We find that this aspect is critical to performance, particularly in the practical scenario where one wants to set a large budget at each iteration.

I validate my method on benchmark datasets for three recognition applications: object recognition, activity recognition, and content-based image retrieval. I demonstrate the advantages of our approach compared to passive, myopic greedy, and batch selection baselines, and show its effectiveness across a range of budgets. Our results indicate that budgeted batch selection is crucial for efficient active learning in practical scenarios, clearly outperforming conventional myopic selection techniques.

1.1.3 Sub-linear Time Active Learning for Web-scale Data

So far, I considered actively selecting annotations from multiple types and for multiple annotators, assuming throughout that human effort is more expensive than machine cycles. However, when one considers applying active learning on very large "unprepared" unlabeled datasets the expense of selecting the annotation to request becomes equally important. Generally methods today are tested in somewhat canned scenarios: the implementor has a moderately sized labeled dataset, and simply withholds the labels from the learner until a given point is selected, at which point the "oracle" reveals the label. In reality, one would like to deploy an active learner on a *truly* massive unlabeled data pool (e.g., all documents on the Web) and let it crawl for the instances that appear most valuable for the target classification task. The problem is that a scan of millions of points is rather expensive to compute exhaustively, and thus defeats the purpose of improving overall learning efficiency.

Therefore, we consider the problem of performing active selection on large-scale datasets where the computational cost of selection outweighs other considerations as shown in Figure 1.4. For active selection, we use the "simple margin" selection criterion for a linear SVM classifier. Given a hyperplane classifier and an unlabeled pool of vector data, the point that minimizes the distance to the current decision boundary is selected for labeling. This is a widely used margin-based selection criterion [98, 85, 15] and it has been shown to substantially reduce total human annotation effort. However, for large-scale active learning, even

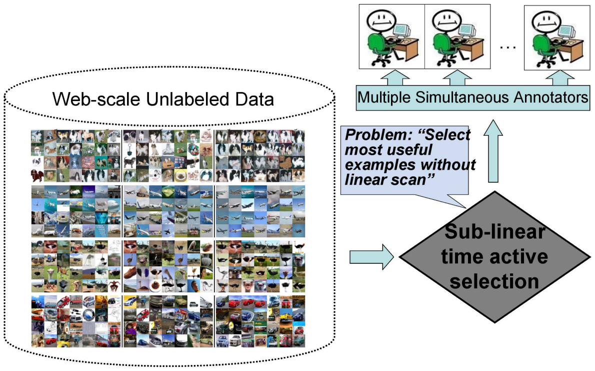

Web-scale Unlabeled Data
Multiple Simultaneous Annotators
Problem: "Select most useful examples without linear scan"
Sub-linear time active selection

Figure 1.4: The problem setting for our sub-linear time active learning approach. In order to deploy an active learner on the large amounts of unlabeled data that are available through web based resources, one must be able to select informative examples with minimal computational costs. The main challenge is therefore to design a selection algorithm that requires sub-linear selection time, i.e., it does not have to exhaustively scan the entire unlabeled pool.

with such a simple albeit effective criterion it is impractical to exhaustively apply the classifier to all unlabeled points at each round of learning. Therefore, to exploit massive unlabeled pools, a fast (sub-linear time) search method to identify the closest points to a given hyperplane is needed.

To this end, I consider the novel Nearest Neighbor to a Query Hyperplane (NNQH) problem and propose two approximate solutions. For each, I introduce randomized hash functions that offer query times sub-linear in the size of the database, and provide bounds for the approximation error of the neighbors retrieved. Our first approach devises a two-bit hash function that is locality-sensitive for the angle between the hyperplane normal and a database point. Our second approach embeds the inputs such that the Euclidean distance reflects the hyperplane distance, thereby making them searchable with existing approximate nearest neighbor algorithms for vector data. While the preprocessing in our first method is more efficient, our second method has stronger accuracy guarantees.

Our two NNQH solutions supply exactly the hash functions needed to rapidly identify the most uncertain examples for a linear SVM classifier according to the “simple margin” selection criterion. Therefore, our algorithms make it possible to benefit from *both* massive unlabeled collections as well as actively chosen label requests. I demonstrate our algorithms’ significant practical impact for large-scale active learning with SVM classifiers. Our results show that our method helps scale-up active learning for realistic problems with massive unlabeled pools on the order of millions of examples.

1.1.4 A Large-scale System for Autonomous Online Visual Learning

My goal in defining and solving the problems described in the previous sections is to enable large-scale transfer of human knowledge for learning visual concepts such as objects and activities. In order to demonstrate the effectiveness of our solution as a viable protocol for learning visual models, as the final component of this thesis, I built the first complete end-to-end system for scalable, autonomous online learning of object detectors.

I chose object detection as a suitable setting to demonstrate aspects of our solution because (1) object detection typically requires identifying a single tight-fitting window among thousands of windows within an image, an ideal setting for our large-scale selection approach; (2) state-of-the-art methods for detection typically require large numbers of training examples annotated using bounding boxes, an expensive process whose expense, we believe, can be significantly reduced using our active approach; (3) it is an extremely challenging problem where any progress is diligently recorded and encouraged [23, 30, 103, 62, 29].

I present an approach for *live learning* of object detectors, in which the system autonomously refines its models while iteratively feeding annotation requests to crowd-sourced human labelers. Rather than fill the data pool with some canned dataset, the system itself gathers possibly relevant images via keyword search. It repeatedly surveys the data to identify unlabeled sub-windows that are most uncertain according to the current model, and generates tasks on Mechanical Turk to get the corresponding bounding box annotations. After an annotation budget is spent, we obtain a category model which can be used to detect instances of the category on novel test images. Notably, throughout the procedure *we do not intervene with what goes into the system's data pool, nor the annotation quality from the hundreds of online annotators.*

In order to handle the technical challenges such a large-scale system entails, I propose a novel part-based detector amenable to linear classifiers, and show how to identify its most uncertain instances in sub-linear time with our hashing-based solution. Our detector strikes a good balance between speed and accuracy, with results competitive with and even exceeding the state-of-the-art on the PASCAL VOC. Most importantly, I show successful live learning in an uncontrolled setting. The system learns accurate detectors with much less human effort than strong baselines that rely on human-verified keyword search results.

1.2 Main Contributions

My thesis makes several important contributions for visual recognition and active learning. I provide a method to actively learn categories from a mixture of weakly and strongly labeled examples. We are the first to identify and address the problem of active visual category learning with multi-level annotations. My approach acknowledges the variable manual effort costs of different images and provides a unified framework for predicting both the information content and the cost of different types of image annotations for the multi-label learning setting. I also define a novel active learning problem, budgeted batch active learning, where there are multiple annotators and each example requires a different annotation cost depending on its difficulty, and propose an efficient solution for choosing a set of examples that fit under an annotation budget. I demonstrate the application of our approach for learning three different tasks: object recognition, activity recognition, and content-based image retrieval. I then provide two hashing-based solutions for the novel approximate nearest neighbor to a hyperplane query problem which enables large-scale pool-based active learning. I empirically demonstrate that our solutions make it practical to perform active selection with millions of unlabeled points. Tying all these together, I develop the first complete end-to-end solution for scalable, automatic online learning of object detectors in which the system autonomously refines its models by actively requesting crowd-sourced annotations on images crawled from the Web.

1.3 Road Map

In the following chapter I discuss some background material and related work to the thesis. In Chapter 3 I consider the problem of learning effectively from annotations occurring at multiple levels of granularity. In Chapter 4, I consider the problem of selecting a batch of examples that provide the most improvement in a classifier objective without overspending a given annotation budget. In Chapter 5, I consider the problem of retrieving the examples that are most informative to a hyperplane classifier without having to exhaustively scan the entire database. In Chapter 6, I explain our approach for *live-learning* of object detectors by autonomously querying annotations on select examples from crowd-sourcing services. Finally, Chapter 7 discusses the main contributions of my thesis.

Chapter 2

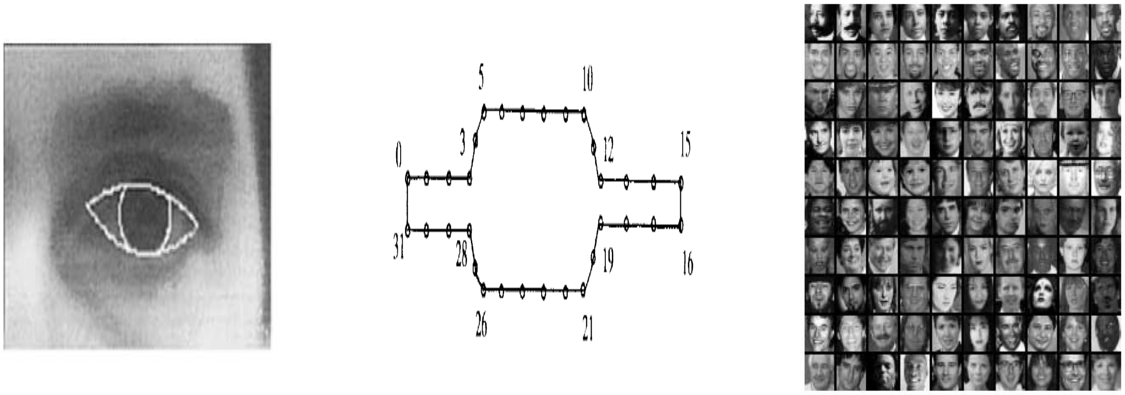
Related Work

In this chapter I review work related to the research presented in this thesis. The fundamental aim of this thesis is to provide a systematic way of introducing supervision while learning object models. Given the expense of labeled image data, researchers have explored various ways to reduce supervision requirements. The relevant previous work aimed at reducing supervision requirements can be broadly grouped into three different threads. The first thread aims at reducing the "level" of supervision, coarsely corresponding to the difficulty of providing a particular type of annotation. The second line of research tackles the "amount" of supervision in terms of the number of training examples that need to be annotated. A third line of research tackles the "effort" involved in providing annotations.

I describe each strategy in turn, and conclude the subsections with a brief overview of important relationships and contrasts with my work.

2.1 Reducing the Level of Supervision

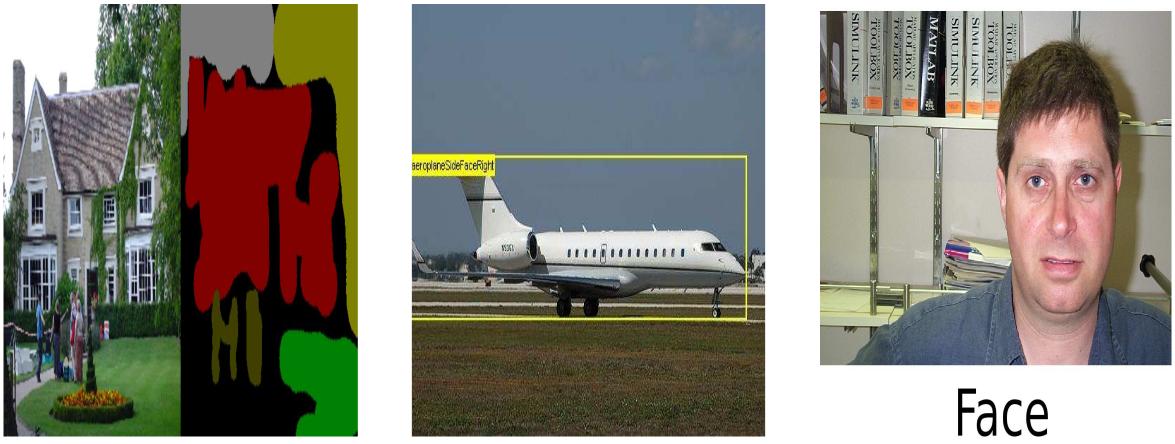
Most approaches to object recognition require some form of human supervision. This may range from identifying parts of objects [128, 22] to cropping and aligning images of objects to be learned [114, 100], to providing complete image segmentations [67] and bounding boxes



(a) Individual object parts (eyes) are outlined in training images of faces.

(b) Landmark points are marked on training objects (resistor).

(c) Images are cropped, aligned and normalized for training a face detector.



(d) All objects in the image are outlined and labeled.

(e) A bounding box is provided around the object of interest.

(f) Image level tag specifying the type of object present.

Figure 2.1: Figure illustrating the different kinds of supervision that can be provided on image data.

[28] to weak image level tags [14] and auxiliary data indicating an object's identity [94, 40] as shown in Figure 2.1 (approximately ordered in decreasing levels of supervision). In the following we review some of these approaches.

2.1.1 Weakly Supervised Recognition

Some of the earliest work in object recognition and detection require considerable human effort in collecting training examples for learning classifiers. Features of faces are detected and described using deformable templates in [128] using hand-crafted parts based on the knowledge of the object category being modeled (e.g. eyes, noses, ears, etc for describing faces). Active shape models are learned for different objects using hand-labeled landmark points on different views of training objects in [22]. In [114], a state-of-the-art face detector is built using training data that consists of cropped, frontal, normalized images of a large number of faces.

Given the expense of collecting such hand-labeled examples for generic objects, computer vision research has since explored the paradigm of *weakly supervised* recognition. These approaches assume that each image contains a single object of interest, and the only supervision required is an image level label saying whether it contains the object of interest or not. Robust models are then built using these "weak" labels to categorize and localize objects in novel images. Using weakly labeled images to learn categories was first proposed in [119, 32]. Similarly, in [5], a joint distribution of image regions and words is learned using only tags on images. These approaches learn to associate or emphasize features on certain image regions based on their repeated occurrence in a large number of training images. In contrast, “stronger” labels such as bounding boxes require more manual effort, but directly provide these regions of interest to the classifier.

Several researchers have since shown that multiple instance learning (MIL) can accommodate the weak or noisy supervision often available for image data [106, 73, 125, 115]. Multiple-instance learning considers a particular form of weak supervision where the learner is given a set of *positive bags*, which are sets of instances containing at least one positive instance and *negative bags*, which are sets of instances none of which are positive. Compared to traditional supervised learning, the labels available in the MIL setting are ambiguous or weak as they do not specify which among the instances in a positive bag are positive. The multiple-instance learning setting was first identified in [25], who represented ambiguously labeled examples using axis-parallel hyper-rectangles and demonstrated applications for drug activity prediction. More recently MIL has received various treatments within the machine learning community [132, 2, 37, 79, 13]. In [13] for example, a large-margin MIL formulation that addresses the possibility of very sparse positive bags is proposed, and it is demonstrated on several machine learning datasets.

There are several instances of MIL in vision targeting different tasks using a similar setting. In this setting, an image consisting of a set of regions/component blobs/segments is a positive bag, and only a subset of the component blobs are true positive examples (i.e., correspond to foreground). Labels are available only at the image level specifying if the image contains at least one component blob that belongs to a particular class. In [73], for example, MIL is used in this setting to classify natural scenes using very low resolution images. Content-based image retrieval is solved using an MIL based approach in [125, 130]. Weakly supervised object localization is performed using an MIL approach in [36] where multiple stable segmentations represent the set of instances in a bag.

Multi-label variants of MIL, Multiple-Instance Multi-Label Learning (MIML), are proposed in [133, 129], where instances within a bag can belong to any number of classes (as opposed to $\{+1,-1\}$ for MIL), but in a similar spirit to MIL only image level labels are available to the classifier. Hence, an image can be associated with any number of classes, which is the case for truly unprepared images. In [133], the MIML problem is transformed into a traditional supervised task by clustering instances within bags and computing a *bag of words* representation of the instances. More recently, the authors of [129, 131] model the relation between instances and labels more explicitly using hidden variables or class-specific feature representations, with the goal of exploiting category co-occurrence cues. In [77], the authors introduce an active approach to select sample-label pairs based on the idea that for multi-label data, only a part of labels need to be annotated while others can be inferred by exploring correlations between labels.

Discussion.

While it is commendable that a lot of research in computer vision is concentrated on low supervision recognition, we must exercise some caution before completely discarding stronger annotations such as bounding boxes or segmentations. The approach of [114] is still the state-of-the-art face detection method even though it utilizes stronger supervision. Therefore, it may be that a single strong annotation could be worth the value of a very large number of weak annotations in some cases. To further reinforce this thesis, the current best approaches on the PASCAL VOC dataset utilize the bounding box information provided around the objects in training images. Though on fairly regular datasets such as Caltech101 weak supervision (image tags) may provide more than sufficient information to learn accurate models, on harder datasets such as the PASCAL VOC stronger information seems essential.

Therefore, instead of completely ignoring stronger annotations, what we need is a principled way of comparing annotations at different levels both in terms of how much manual effort they consume and how much information they provide to the classifier to best utilize the manual effort resources that are available to us. This is one of the major problems that I tackle in this thesis.

In particular, MIL provides a way of learning simultaneously from strongly and weakly labeled examples, which I show in later chapters to be a crucial functionality for minimizing supervision requirements. As part of this thesis, I also propose a kernel-based approach for solving the MIML problem that produces state-of-the-art results in order to tackle object recognition in the multi-label setting. Further, I identify an additional setting where a weak form of supervision is available in object recognition and show how MIL can be used in conjunction with keyword based search results to obtain a method to learn object categories without any human supervision.

2.1.2 Unsupervised Recognition

Apart from utilizing weak forms of supervision, recent methods have also shown the possibility of learning visual patterns from unlabeled image collections [78, 94, 40, 66]. In this group of methods the only supervision the user is required to provide is to identify a single object's or a cluster's identity once the images have been automatically organized. Several of these approaches are based on image clustering techniques that accommodate local image feature representations. A number of authors have studied probabilistic clustering methods originally used for text—such as probabilistic Latent Semantic Analysis (pLSA), Latent Dirichlet Analysis, and Hierarchical Dirichlet Processes—to discover the hidden mixture of visual themes ("topics") in a collection of unorganized [94, 82] or semi-organized [31, 70] image data. Alternatively, several approaches have considered clustering algorithms such as normalized cuts and affinity propagation [40, 66, 26, 56]. Given an unlabeled collection of images these approaches typically organize the images into different groups that have a semantic meaning with respect to an underlying similarity matrix. These methods demonstrate excellent results on datasets such as Caltech101 where the images belonging to a category are highly regular.

However, clustering methods are most appropriate for mining image data, but not necessarily for learning categories: they may sometimes elicit themes associated with semantic categories, but there is no way to guarantee it. A common flaw of such methods, frequently referred to, is that they are themselves not equipped to provide models to classify novel examples. For example, pLSA requires some way to select which topic to use for each class model, and must resort to a "folding-in" heuristic. Many of the clustering approaches require a large number of constantly repeating patterns to produce effective clusters, and therefore may not be appropriate when such data is scarce.

2.1.3 Alternative Sources of Training Data

In addition to reducing the level of supervision, vision researchers have also identified innovative ways to take advantage of data sources where text naturally accompanies images.

These methods exploit the weak connection between the text surrounding images and the semantic meaning of the image to essentially obtain “free” training data with which they can learn classifiers. For example, in [8], news captions occurring alongside photographs are used to cluster faces occurring in the pictures. Annotated stock photo libraries have been used by a number of approaches such as [27] to automatically annotate novel examples. Several authors use the information present in generic web pages to filter and mine useful images for categories [127, 9, 86].

In particular, several methods try to download images of a category through keyword searches and learn visual category models straight from the automatically collected image data. Such approaches attempt to deal with the images’ lack of homogeneity indirectly, either by using clustering techniques to establish a mixture of possible visual themes [94, 31, 70], or by applying models known to work well with correctly labeled data to see how well they stretch to accommodate “noisily” labeled data [33, 86]. Unfortunately, the variable quality of the search returns and the difficulty in automatically estimating the appropriate number of theme modes make such indirect strategies somewhat incompatible with the task.

Discussion. With the advent of more search functionalities through several online portals such as Flickr which provide high quality image data, utilizing such “free” data becomes imperative if the goal is to minimize supervision requirements. As part of the thesis, I outline a method that exploits text-based indexing to gather image examples, however thereafter it learns categories from the image content alone. In contrast to previous approaches, this allows categories of interest to be directly specified, and produces a large-margin classifier to recognize novel instances. In later chapters, I also develop a large-scale system that autonomously learns category models by collecting image examples through keyword searches and obtains annotations using crowd-sourcing services.

2.2 Reducing the Amount of Supervision

A parallel line of research aimed at reducing the supervision requirements looks at minimizing the number of labeled examples. Semi-supervised learning methods accomplish this by supplementing the limited amount of labeled data with a large number of easily available unlabeled data. Active learning approaches, on the other hand, reduce the number of labels required to learn a classifier by allowing the classifier to query labels on only the most informative examples. Notably, prior to the research in my thesis, there is little work in visual recognition exploiting active learning strategies.

2.2.1 Semi-supervised Visual Learning

Semi-supervised learning approaches build classifiers by using a large amount of unlabeled data, together with a small amount of labeled data. The earliest work in semi-supervised learning utilize unlabeled data based on the “cluster” assumption that points of a class tend to form clusters. Based on this assumption, unlabeled data can then be used as an aid in finding the boundary of each cluster, following which the labeled points can be used to assign a class to each cluster [97]. Because semi-supervised learning requires less human effort and gives higher accuracy, several researchers have considered applying such techniques for problems in recognition.

In [16, 40] the possibility of learning visual patterns from partially labeled image collections is explored in a semi-supervised setting for image classification. Using partially annotated data, the authors augment the model with constraints to handle weak supervision and achieve performance comparable to the fully supervised setting.

In [70], a form of semi-supervised learning, self-training, is used to learn from a combination of labeled seed examples and unlabeled images downloaded from the web. In self-training the classifier is first trained using the small amount of labeled data available. The most confident unlabeled points, together with their predicted labels, are then added to the training set and the classifier is re-trained. Co-training, another form of semi-supervised learning where two or more independent classifiers train each other by classifying disjoint sets of unlabeled data, is explored for action recognition in [43] and for object detection in [68].

One known difficulty with semi-supervised learning is that bad matching of problem structure with the semi-supervised model assumption could lead to degradation in classifier performance [134]. For example, many semi-supervised approaches assume that the density of points near the decision boundary is low. When data is sampled from two overlapping Gaussian distributions, such an assumption becomes invalid which could lead to poor performance. Therefore, detecting the problem structure in advance and applying the right semi-supervised technique is important in such cases.

2.2.2 Active Learning

In contrast, active learning strategies use unlabeled points to instead select informative examples on which labels are obtained from a human oracle (see Figure 2.2). Therefore, active learning directly targets the number of labeled examples that need to be provided to the clas-

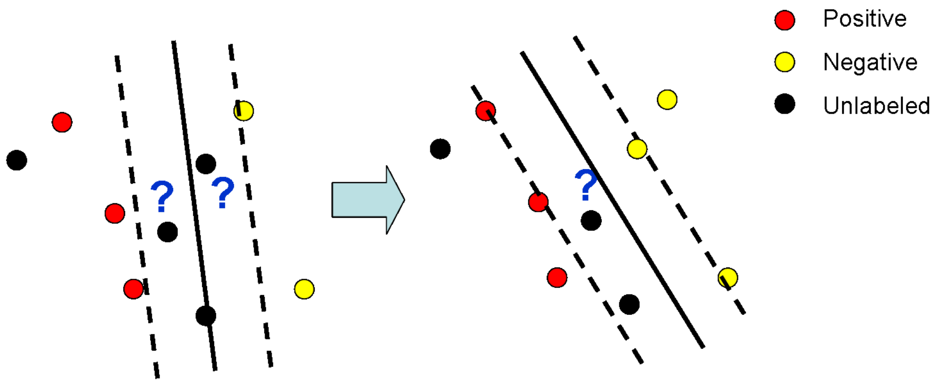


Figure 2.2: In active learning the classifier is initially trained on a small set of examples, and the classifier chooses informative examples from an unlabeled pool of data to request labels. In the figure points in black are unlabeled data points, and the classifier is most uncertain about the examples that fall between the positive and negative planes. Therefore, requesting labels on these examples would provide the most benefit to the classifier. Note that this figure shows one particular form of active learning based on margin-based selection [98] as an intuitive example.

sifier to learn a good model. There are several active learning strategies introduced in the machine learning literature for choosing the right example to query. See [87] for a detailed survey of active learning approaches.

Uncertainty sampling is a popular method that selects the examples over which the current classifier is least confident on a label assignment. A straightforward uncertainty sampling approach for probabilistic classifiers or support vector machines would be to choose points that have probabilities close to 0.5 [69], or that are along the margin of the separating hyperplane. For multi-label classifiers the classification entropy of an unlabeled instance provides a more general uncertainty measure and several approaches propose to select the point with the largest entropy for labeling [46].

A more theoretically motivated framework for active learning considers choosing points that reduce the version space of the classifier the most. The version space of a classifier is the set of all hypotheses that are consistent with the current set of labeled examples. If training a classifier is viewed as a search for the best model within the version space, then the goal of such methods is to constrain the size of the version space as much as possible using few labeled instances. In [98], a simple margin-based selection method for SVMs is proposed which attempts to minimize the version space of the SVM at every iteration. In [90], a query by committee approach for active learning is proposed, where a competing committee of models is initially trained on some labeled data, following which the unlabeled instance that disagrees most with the committee is selected for labeling. Theoretical justifications showing that such an approach reduces the version space is provided in [35].

Another general active learning approach is to query the instance that would impart the greatest change to the current model if its label were known. For example, decision-theoretic measures that compute the expected model change have been explored in [71, 55], where they were applied to classify synthetic data and voicemail. A common result of all these approaches is that random or passive learning tends to waste a large amount of labeled training data which are either uninformative or already correctly classified, while active learning strategies instead choose only a small subset of the training data to reach similar accuracies.

Given the advantages of such techniques and the expense of obtaining labeled image data, a few authors have begun to apply active learning methods for different problems in vision.

Relevance feedback in content based image retrieval was first identified as a possible application of active learning in [18, 75], where image retrieval results are refined to match a user's subjective query concept. In [123], relevance feedback is considered for video annotations in an active learning framework taking into account the higher expense of annotating video data.

Active learning with Gaussian Process classifiers is introduced in [54] for the task of image recognition. An active criterion for instance-level queries is suggested in [89] and applied within an MI learner for an image categorization task where an image is a weakly labeled bag of segments. More recently, [48, 52] consider active learning for multi-class object recognition using nearest neighbor and support vector machine classifiers. In [77] the authors propose a two dimensional active selection approach that selects along both the label dimension and the instance dimension, which applies to the multi-label setting where an example is associated with multiple labels. Dataset creation using active selection is explored in [21]. These approaches are examples of traditional active learning, which we also refer to as *flat-cost active learning* since they assume unifom cost for obtaining labels and try to reduce the total number of labels.

However, in reality, the amount of effort required to provide labels could vary significantly across different unlabeled examples and labels depending on a number of factors. A few studies in the learning community try and quantify manual effort on a per example basis for different learning tasks in order to perform *cost-sensitive active learning*. In [55, 4], the length of a voice mail or sentence is used to approximately identify examples that could take more or less manual effort to annotate. Budgeted learning for *active classifiers*, which work on constrained budgets while querying attributes on a test example, is explored in the work of [41] for medical diagnosis. In [80], regressors are learned based on sentence length, number of characters, etc to predict annotation time for document classification. While the length of a voice mail or the cost of a medical diagnosis directly provides the cost of an example, no such direct measure exists to quantify the effort involved in providing an image annotation. Thus far, no existing approaches in object recognition attempt to quantify or predict the amount of effort required to provide annotations on image examples.

The above approaches focus on selecting a single unlabeled instance to label, retraining the classifier at each iteration. However, retraining can be expensive and even disruptive to the learning process; in fact, when one has access to multiple "labelers" at once (e.g., on Mechanical Turk [96, 24]), a *batch* selection would be more effective. A few *batch-mode active learning* approaches have been proposed recently [12, 57, 42], including one that targets a computer vision application [45]. Batch selection calls for more than a selection of the N-best queries at a given iteration, since such a greedy strategy does not account for possible overlap in information. Instead, researchers design selection functions that balance informativeness with the so-called *diversity* among the selected set [12, 45]. In [57], the authors provide bounds on the advantage of myopic active learning over batch-mode selection methods when using the maximum entropy criterion, which is an important result for batch-mode selection approaches. Using this bound they design an algorithm that switches between sequential active learning (exploration) and batch-mode learning (exploitation) depending on the tightness of the bounds on the current classifier.

Discussion. Active learning approaches so far considered for object recognition are based on what we refer to as the traditional active learning paradigm for binary or multi-class classification. In traditional active learning each example has only one type of label associated with it, and the idea is to select the example that should be labeled next. We also refer to this type of active learning *single-level* active learning to emphasize the fact that labels are obtained at a single level of granularity. However, the active selection problem for object recognition is more complex. Most real world images typically contain multiple objects belonging to different categories, and users can provide annotations at different levels depending on the classifier's knowledge. Moreover, the amount of manual effort required to provide a particular annotation could vary significantly depending on both the type of annotation and the particular image example.

In addition, most approaches are myopic in the sense that they choose a single example to label at an iteration. The few batch-mode active learning approaches ignore the cost of labeling examples and rely on the current classifier to estimate uncertainty. As a result, these functions' performance can degrade with very large batches and when the examples have variable costs.

Finally, previous active learning methods in vision focus on image classification, and no work addresses the more complex problem of object detection where one needs to not only name objects but also localize their extent in the images. Existing approaches demonstrate results under the "sandbox" setting where fixed datasets of modest scale have already been selected and labeled. I address these issues in this thesis.

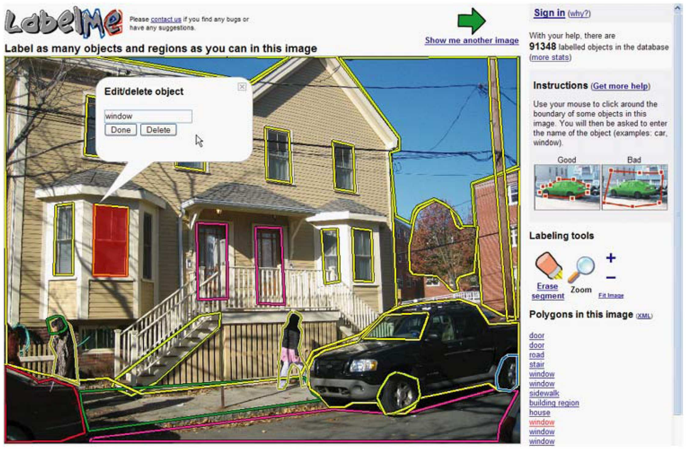
2.3 Reducing the Effort of Providing Supervision

Collecting benchmark datasets and training examples for testing various object recognition methods has been largely the responsibility of the vision researchers trying to build category models. The Caltech 101 dataset for 101 different categories was, for example, collected from the web and manually pruned by a group of researchers at Caltech. This places a huge burden on the researchers and could further introduce biases in the dataset. Thus, some research seeks to facilitate or ease the “effort” involved in collecting and providing annotated examples using innovative interfaces for images and videos.

2.3.1 Labeling Services

A few distributed labeling services that target the large number of users in the world wide web have been successfully utilized for obtaining image annotations efficiently. The Labelme dataset [83] targets vision researchers by providing cleanly labeled data in return for providing annotations using their free web annotation tool (see Figure 2.3). More recently, the possibility of directly compensating annotators with cash in a distributed framework has arisen with crowd sourcing services such as Mechanical Turk [96].

These methods provide nice web-based interfaces where the user is shown an image along with possibly one or more existing annotations, which are drawn on the image. The user has the option of annotating a new object by clicking along the boundary of the desired object and indicating its identity, or editing an existing annotation. The user may annotate as many objects in the image as they wish. Most users work either for direct compensation or as a constructive way of entertaining themselves.



Please contact us if you find any bugs or have any suggestions.

Label as many objects and regions as you can in this image

With your help, there are 91348 labelled objects in the database (more stats)

Instructions (Get more help)

Use your mouse to click around the boundary of some objects in this image. You will then be asked to enter the name of the object (examples: car, window).

Edit/delete object: window

Figure 2.3: A screenshot of the labeling tool in use. The user is shown an image along with possibly one or more existing annotations, which are drawn on the image. The user has the option of annotating a new object by clicking along the boundary of the desired object and indicating its identity, or editing an existing annotation. The user may annotate as many objects in the image as they wish. Image from [83].

2.3.2 Image Annotation Games

In the work of Luis Von Ahn [116, 117, 44], the process of providing annotations is posed as innovative games which are targeted at the large community of online users. When users play the game they help determine the contents of images and user preferences by providing meaningful labels for them. Briefly, the games are played by two partners and is meant to be played online by a large number of pairs at once. Partners are randomly assigned from among all the people playing the game.

In [116], the goal of the game is to guess what their partner is typing for each. Once both players have typed the same string, they move on to the next image. The words that two players agree on are treated as valid labels for the image. In Peekaboom[117], one of the partners (Boom) gets an image along with a word related to it, and must reveal parts of the image for the other partner (Peek) to guess the correct word. Similarly, Matchin [44] measures user preferences by making people guess which images their partners would prefer.

2.3.3 Interactive Segmentation

Interactive segmentation algorithms make it easier for an annotator to specify a region of interest. Such tools require very little user interaction to segment complex foreground objects without having to trace the outline of the entire object. Intelligent Scissors [74] allows a user to choose minimum cost paths by roughly tracing the object's boundary with the mouse. As the mouse moves, the minimum cost path from the cursor position back to the point is shown. In graph Cut [11], the interface allows users to mark certain pixels as belonging to the foreground or background. Graph cut algorithms are then used to find a globally optimal segmentation using boundary and region information. Grabcut [81] extends the graph-cut segmentation tool using a robust algorithm for "border matting" to estimate simultaneously the alpha-matte around an object boundary and the colors of foreground pixels.

Discussion. The above approaches provide nice interfaces for collecting annotations with the goal of making the annotation process less cumbersome or more interesting to the user. However, they select image queries in a random fashion without a notion of which example might be useful to a classifier. Integrating an active learning framework with these techniques could provide more direction to the image queries enabling the construction of small but extremely informative datasets for learning categories.

Having summarized related work in this area, I will next detail my approach to address these issues in the following chapters. The next chapter introduces one of the central ideas of my thesis, that of *multi-level active learning*.

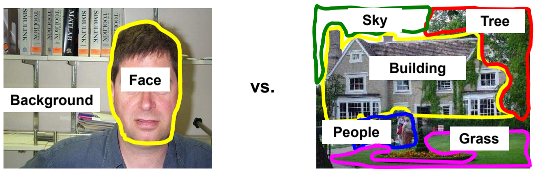
Chapter 3

Cost-sensitive Active Learning with Multi-level Queries

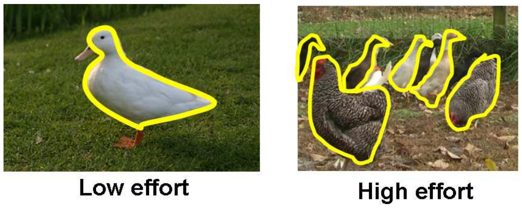
Active learning strategies provide a way to reduce the reliance on labeled training data by minimizing the number of labeled examples required to learn classifiers. They typically do this by allowing the classifier to choose which example needs to be labeled next from a large pool of unlabeled examples, reducing supervision without sacrificing much accuracy in the final model. The assumption is that while unlabeled examples can be collected with little or no effort, providing annotations on the examples entails non-trivial effort. Such methods are therefore appealing for object recognition because of the abundance of unlabeled images (available, for example, on the Web) and the substantial effort required to provide detailed annotations.

However, in the general case, visual category learning does not fit the mold of traditional active learning approaches, which primarily aim to reduce the number of labeled examples required to learn a classifier, and almost always assume a binary decision task. When trying to choose informative image data to label for recognition, there are three important distinctions we ought to take into account.

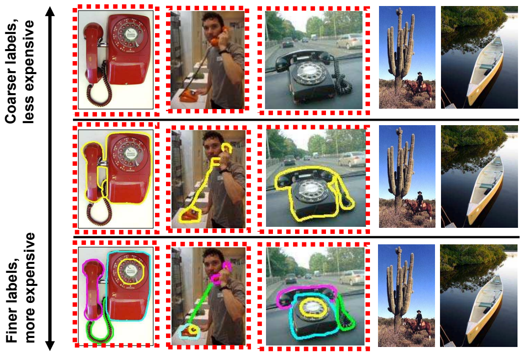
First, while many of today's manually collected datasets assume that the class to be learned occurs prominently in the foreground and therefore can be associated with a single label,



(a) Most real-world images contain multiple objects and can therefore be associated with multiple labels.



(c) The actual manual effort required to label varies according to annotation type and image example.



(b) Useful image annotations can occur at multiple levels of granularity. For example, a learner may only know whether the image contains a particular object or not (top row, dotted boxes denote object is present), or it may also have segmented foregrounds (middle row), or it may have detailed outlines of object parts (bottom row).

Figure 3.1: Three important problems that need to be addressed while choosing informative image data to label for recognition, none of which are considered by traditional active learning approaches.

most naturally occurring images consist of multiple objects. Therefore, an image can be associated with *multiple labels* simultaneously as shown in Figure 3.1(a).[1] This means that an active learner must assess the value of an image containing some unknown combination of categories.

Second, whereas in conventional learning tasks the annotation process consists of simply assigning a class label to an example, image annotation can be done at different levels—

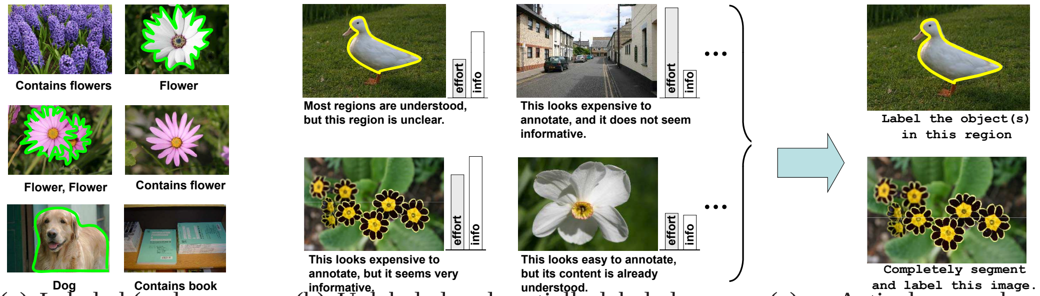
---

[1] Multi-label is thus more general than *multi-class*, where usually each example is assumed to represent an item from a single class.

by assigning class labels, drawing a segmentation of object boundaries, or naming some region (Figure 3.1(b)). Richer annotations such as segmentations provide more information from which to infer class membership, but require more effort on the part of the person providing supervision. While recent work has begun to explore ways to reduce the level of supervision ([119, 94, 78, 6, 31, 70, 116, 83, 105]), such techniques fail to address a key issue: to use a fixed amount of manual effort most effectively may call for a combination of annotation at different supervision levels. Therefore, instead of ignoring annotations such as segmentations, which require more effort to obtain, we need a principled way of predicting the tradeoff between the effort and information gain associated with any candidate image annotation. This means an active learner must be able to choose from annotations at multiple levels of granularity and specify not only which example but also what *type* of annotation is currently most helpful.

Third, while previous methods implicitly assume that all annotations cost the same amount of effort (and thus minimize the total number of queries), the actual manual effort required to label images varies both according to the annotation type as well as the particular image example. For example, completely segmenting an image and labeling all objects requires more time and effort than providing an image-level tag specifying object presence. Even for the same type of annotation, some images are faster to annotate than others (e.g., a complicated scene versus an image with few objects, as seen in Figure 3.1(c)).

In this chapter, I incorporate these insights and propose a unified framework for predicting both the information content and the cost of different types of image annotations, for the Multiple-instance Multi-label learning (MIML) setting. Figure 3.2 provides an overview of


Contains flowers
Flower
Flower, Flower
Contains flower
Dog
Contains book
effort
info
Most regions are understood, but this region is unclear.
This looks expensive to annotate, and it does not seem informative.
This looks expensive to annotate, but it seems very informative.
This looks easy to annotate, but its content is already understood.
...
Label the object(s) in this region
Completely segment and label this image.

(a) Labeled (and partially labeled) examples to build models

(b) Unlabeled and partially labeled examples to survey

(c) Actively chosen queries sent to annotators

Figure 3.2: Overview of the proposed approach. (a) We learn object categories from multi-label images, with a mixture of weak and strong labels. (b) The active selection function surveys unlabeled and partially labeled images, and for each candidate annotation, predicts the tradeoff between its informativeness versus the manual effort it would cost to obtain. (c) The most promising annotations are requested and used to update the current classifier.

our proposed approach. After learning from a small initial set of labeled images, our method evaluates all available unlabeled data using a novel Value of Information (VOI) based selection function in order to choose the most promising annotation to receive next. After re-training, the process repeats, continually improving the models with minimal manual intervention. [2]

3.1 Multi-level Active Prediction of Useful Image Annotations for Recognition

The goal of this work is to learn category models with minimum supervision under the real-world setting where each potential training image can be associated with multiple classes. Throughout, our assumption is that human effort is more scarce and expensive than machine

---

[2] The contents of this chapter were published in [106, 107, 108, 110].

cycles; thus our method prefers to invest in computing the best queries to make, rather than bother human annotators for an abundance of less useful labelings.[3]

We consider image collections consisting of a variety of supervisory information: some images are labeled as containing the category of interest (or not), some have both a class label and object outlines, while others have no annotations at all. We derive an active learning criterion function that predicts how informative further annotation on any particular unlabeled image or region would be, while accounting for the variable expense associated with different annotation types. Specifically, we show how to continually assess the value of three different types of annotations: a label on an image region, an image-level tag, and a complete segmentation of the entire image (see Figure 3.6). We also refer to these types as “levels”, since they correspond to different levels of detail in the annotation. As long as the information expected from further annotations outweighs the cost of obtaining them, our algorithm will request the next valuable label, re-train the classifier, and repeat.

In the following, I introduce the multiple-instance learning (MIL) and multiple-instance multi-label learning (MIML) frameworks and define a discriminative kernel-based classifier that can deal with annotations at multiple levels (Section 3.1.1). Then, I derive a decision-theoretic function to select informative annotations in this multi-label setting, leveraging the estimated costs (Section 3.1.2.3). Finally, I develop a novel method to predict the cost of an annotation (Section 3.1.2.4).

---

[3] Later in Chapter 5, I will return to the issue of how to also minimize the machine effort (selection time).

3.1.1 Multiple-instance Multi-label Learning

An arbitrary unlabeled image is likely to contain multiple objects. At the same time, typically the easiest annotation to obtain is a list of objects present within an image. Both aspects can be accommodated in the multiple-instance multi-label learning setting, where one can provide labels at multiple levels of granularity (e.g., image-level or region-level), and the classifier learns to discriminate between multiple classes even when they occur within the same example.

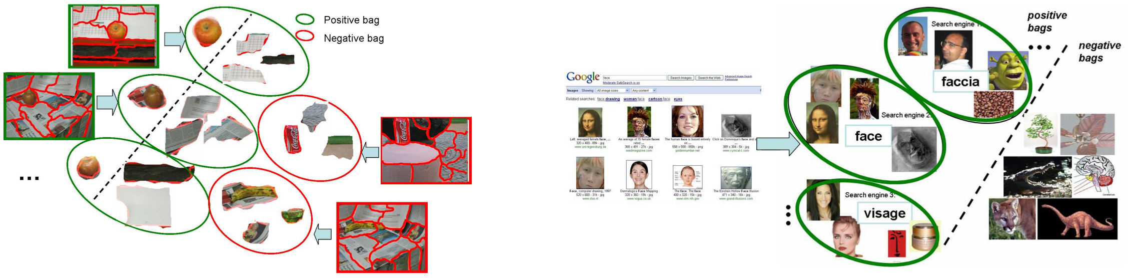
In the following, I extend support vector machine based MIL to the multi-label case. The main motivation of our design is to satisfy both the multi-label scenario as well as the needs of our active selection function. Specifically, we need classifiers that can rapidly be incrementally updated, and which produce probabilistic outputs to estimate how likely each label assignment is given the input.

3.1.1.1 Multiple-instance Learning

In the MIL setting, as first defined by [25], the learner is given *sets* (bags) of instances and told that at least one example from a positive bag is positive, while none of the members in a negative bag is positive. The goal of MIL is to induce the function that will accurately label individual instances such as the ones within the training bags in spite of the label ambiguity: the ratio of negative to positive instances within every positive bag can be arbitrarily high.

Specifically, there is a set of labeled training bags $\mathcal{X}_L$, which is itself comprised of a set of positive bags $\mathcal{X}_p$ and a set of negative bags $\mathcal{X}_n$. Let X be a bag of instances, and $\mathcal{X}_p =$

$\{X \in \mathcal{X}_p\}$ and $\tilde{\mathcal{X}}_n = \{x | x \in X \in \mathcal{X}_n\}$ be the set of positive bags and negative instances, respectively. The goal is to determine the function $f : \Re^{d^N} \rightarrow \{+1, -1\}$ that best predicts labels for new input patterns drawn from the same distribution as the training examples, such that the probability of error is minimized.



(a) (b)

Figure 3.3: Example scenarios where MIL is suitable for image classification. (a) In this scenario training images are represented by a bag of regions and a positive image contains at least one of the regions containing the object of interest. (b) Groups of images downloaded from keyword searches from multiple search engines are positive bags and individual images are the instances.

MIL is well-suited for the following two image classification scenarios as illustrated in Figure 3.3:

- In the first scenario, training images are labeled as to whether they contain the category of interest, but they also contain other objects and background clutter. Every image is represented by a bag of regions, each of which is characterized by its color, texture, shape, etc. [73, 125] as shown in Figure 3.3(a). For positive bags, at least one of the regions contains the object of interest. The goal is to predict when new image regions contain the object—that is, to learn to label regions as foreground or background. Since a positive instance is a positive bag containing a single instance, MIL can accommodate both region labels (instance-level) and image tags (bag-level). This is a form of “weak” supervision.

- In the second scenario, the keyword associated with a category is used to download groups of images from multiple search engines in multiple languages. Each downloaded group is a bag, and the images within it are instances (Figure 3.3(b)). For each positive bag, at least one image actually contains the object of interest, while many others may be irrelevant. The goal is to predict the presence or absence of the category in new images. We first proposed this scenario in [106].

In both cases, an instance-level decision is desirable, but bag-level labels are easier to obtain. While it has been established that MIL is a valuable classification paradigm in such cases, previous methods do not consider how to determine what labels would be most beneficial to obtain.

3.1.1.2 Multiple-instance Multi-label Learning

While in the MIL setting described so far each bag is labeled as positive or negative, in the more general MIML setting, each instance within a bag can be associated with one of C possible class labels; therefore each bag is associated with multiple labels—whichever labels at least one of its instances has.

Formally, let $\{(X_1, L_1), (X_2, L_2), \ldots, (X_N, L_N)\}$ denote a set of training bags and their associated labels. Each bag consists of a set of instances $X_i = \{x_1^i, x_2^i, \ldots, x_{n_i}^i\}$, and a set of labels $L_i = \{l_1^i, l_2^i, \ldots, l_{m_i}^i\}$, where n_i denotes the number of instances in X_i, and m_i

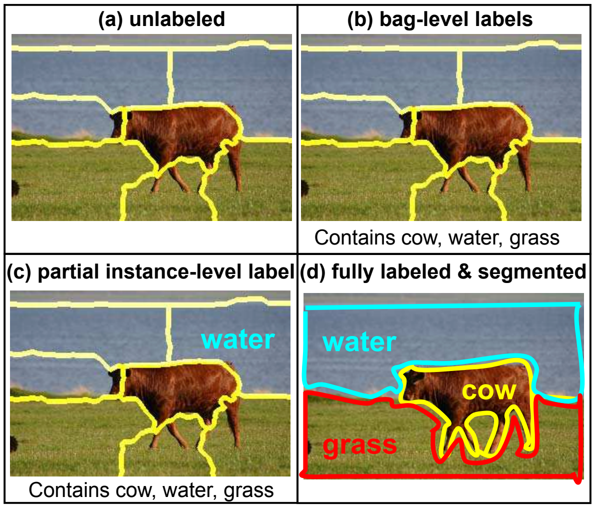

(a) unlabeled
(b) bag-level labels
Contains cow, water, grass
(c) partial instance-level label
(d) fully labeled & segmented
water
Contains cow, water, grass
water
cow
grass

Figure 3.4: In our MIML scenario, images are multi-label bags of regions (instances). Unlabeled images are oversegmented into regions (a). For an image with *bag-level* labels, we know which categories are present in it, but we do not know in which regions (b). For an image with some *instance-level* labels, we have labels on some of the segments (c). For a *fully annotated* image, we have true object boundaries and labels (d).

denotes the number of labels in L_i. Note that often a bag has fewer unique labels than instances ($m_i \leq n_i$), since multiple instances may have the same label. Every instance x_j^i is associated with a description $\varphi(x_j^i)$ in some kernel embedding space and some class label $l_k^i \in \mathbb{L} = \{1, \ldots, C\}$, but with only the bag-level labels it is ambiguous which instance(s) belongs to which label. A bag X_i has label l if and only if it contains at least one instance with label l. Note that a labeled instance is a special case of a bag, where the bag contains only one example ($n_i = 1$), and there is no label ambiguity.

We first consider the first scenario defined above: an image is a bag, and its instances are the oversegmented regions within it found automatically with a segmentation algorithm (see Figure 3.4). A bag's labels are tags naming the categories present within the image; a region (instance) label names the object in the particular region. Each region has a feature vector describing its appearance. This follows the common use of MIL for images ([73, 129, 107]), but in the generalized multiple-instance multi-label case.

Our MIML solution has two components: first, we decompose the multi-class problem into a number of binary problems, in the spirit of standard one-vs-one classification; second, we devise a *Multi-label Set Kernel* that performs a weighting in kernel space to emphasize different instances within a bag depending on the category under consideration.

Each one-vs-one binary problem is handled by an SVM trained to separate bags containing label l_i from those containing l_j, for all i, j. For the single-label case, one can average a bag's features to make a single feature vector summarizing all its instances: $\varphi(X_i) = \frac{1}{|X_i|} \sum_{j=1}^{n_i} \varphi(x_j^i)$, and then train an SVM with instances and bags; this is the Normalized Set Kernel (NSK) approach of [37]. The NSK is a kernel for sets, and is derived from the defi-

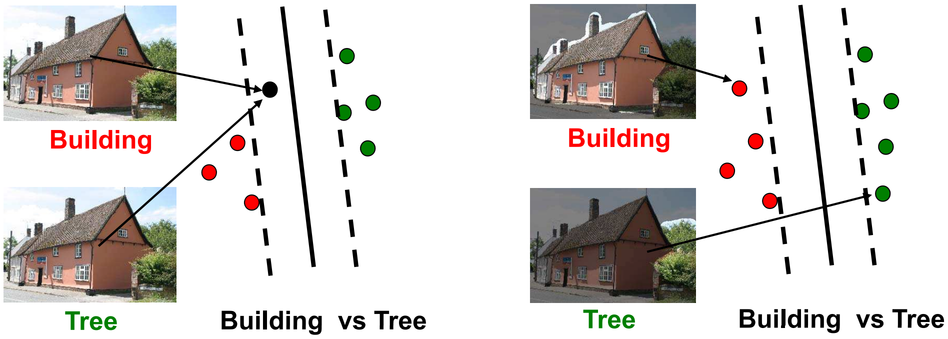


Figure 3.5: The intuition behind our multi-label kernel function. **Left:** In MIML, if an image's representation is independent of its label, two different labels could map to the same point in feature space. **Right:** Our Multi-label Set Kernel weighs instances based on the predicted class membership, thereby associating specific regions within the image to the provided labels. In the top image the region containing a building (lighter shading) contributes more to the overall image representation given the label "building", while in the bottom image the region containing a tree contributes more for the label "tree".

nition of convolution kernels using the set-membership function. In order to correct for the cardinality of the sets, a normalization factor based on the 1 or 2-norm is introduced. For the MIL setting, every instance in a bag can be seen as a member of the bag, and the NSK corresponds to an averaging process carried out in feature space. The NSK approach can be construed as a balancing constraint on the positive bags as shown in [13]. Intuitively, this means that *on average* we expect the label on a positive bag to be greater than zero.

However, in the multi-label case, some bags could be associated with *both* labels l_i and l_j. Simply treating the image as a positive example when training both classes would be contradictory (see Figure 3.5 (left)). Intuitively, when training a classifier for class l_i, we want a bag to be represented by its component instances that are most likely to have the label l_i, and to ignore the features of its remaining instances. Of course, with bag-level labels only, the assignment of labels to instances is unknown.

I therefore propose a Multi-label Set Kernel (MSK) that weights the feature vectors of each instance within the bag according to the estimated probability that the instance belongs to the class. That way if an instance has a high chance of belonging to the given class, then its feature vector will dominate the representation (Figure 3.5 (right)). To this end, we design a class-specific feature representation of bags. Let $X = \{x_1, \ldots, x_n\}$ be a bag containing labels $L = l_1, \ldots, l_m$ (where here we drop the example index i for brevity). We define the class-specific feature vector of X for class l_k as

$$\varphi\left(X^{(l_k)}\right) = \sum_{j=1}^{n} \mathrm{Pr}(l_k|x_j)\varphi(x_j), \tag{3.1}$$

which weights the component instances by their probability of being associated with the class label under consideration. Here $\mathrm{Pr}(l_k|x_j)$ denotes the *true* probability that instance x_j belongs to category l_k, which we approximate as $\mathrm{Pr}(l_k|x_j) \approx p(l_k|x_j)$, where $p(l_k|x_j)$ is the posterior probability output by the classifier using the training data seen thus far. For a single instance (or equivalently, a single-instance bag), there is no label ambiguity, so the instance is simply represented by its feature vector.

For generic kernels, we may not know the feature space mapping $\varphi(x)$ needed to explicitly compute Equation (3.1). Instead, we can apply the same feature weights via the kernel value computation. Let X_1 and X_2 be bags associated with labels l_1 and l_2, respectively, that are currently being used to construct a classifier separating classes l_1 and l_2. Then the MSK kernel value between bags X_1, X_2 is given by

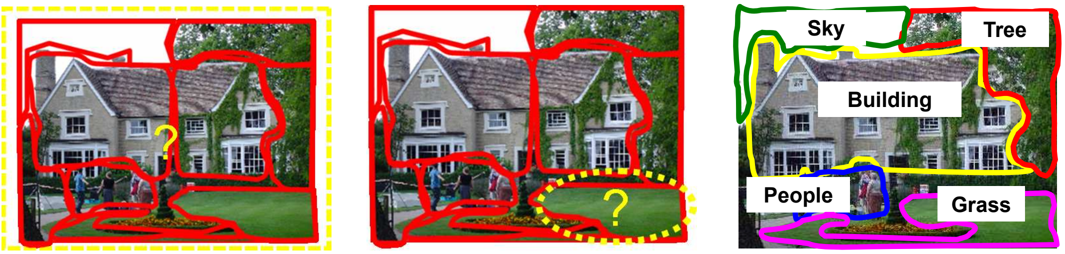
$$\mathcal{K}(X_1^{(l_1)}, X_2^{(l_2)}) = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} p(l_1|x_i^1)\; p(l_2|x_j^2)\; \mathcal{K}(x_i^1, x_j^2),$$

where $\mathcal{K}(x_i^1, x_j^2) = \varphi(x_i^1)^T\varphi(x_j^2)$ is the kernel value computed for instances x_i^1 and x_j^2, and $p(l_1|x_i^1), p(l_2|x_j^2)$ are the posteriors from the current classifiers. Note that because the kernel is parameterized by the label under consideration, a multi-label bag can contribute multiple different $\langle$feature,label$\rangle$ pairs to the training sets of a number of the one-vs-one classifiers.

Our Multi-label Set Kernel can be seen as a generalization of the NSK [37], which is restricted to single-label binary classification. It is also related to the kernel in [61], where weights are set using a Diverse Density function. In contrast, we estimate the class conditional probabilities using the classifier constructed with the currently available training data.

The proposed kernel is valid for both instances and bags, and thus can be used to build SVMs for all required component binary problems. Each SVM can accept novel instances or bags: the feature for an input instance is unchanged, while an input bag is weighted according to Equation (3.1). Given a new input X_{new}, we (a) run it through all $\frac{1}{2}C \times (C-1)$ classifiers, (b) compute the $\frac{1}{2}C \times (C-1)$ resulting two-class posteriors using the method of [76], and, finally, (c) map those posteriors to the multi-class posterior probabilities $p(l|X_{new})$ for each label $l \in \{1, \ldots, C\}$. For this last step we use the pairwise coupling approach of [122], where the pairwise class probabilities are used to solve a linear system of equations to obtain the multi-class probabilities.

While in our implementation we combine one-vs-one binary problems to obtain a multi-class classifier, our method is not restricted to this setting. Since our approach defines a kernel for the multi-label problem, it can be used with other kernel-based multi-class approaches, including one-vs-all SVMs.



(a) Name an object in the image (unlabeled bag).

(b) Label the specified region (unlabeled instance).

(c) Segment the image and name all objects (label all instances).

Figure 3.6: The three candidate annotation types (or "levels") that our approach chooses from when formulating a request.

3.1.2 Active Multi-level Selection of Multi-label Annotations

Thus far we have defined the multi-label learner, the basic classifier with which we want to actively learn. Next we describe our strategy to do active selection among candidate annotations.

There are three possible types of annotation request: the classifier can ask for a label on a bag, a label on an instance within a bag, or a label on all instances within a bag. A label on a bag serves as a "flag" for class membership, which is ambiguous because we do not know which of the instances in the bag are associated with the label. A label on an instance unambiguously names the class in a single image region, while labeling all instances within a bag corresponds to fully segmenting and labeling an image. Figure 3.6 illustrates each of these three types.

In the following subsections, I will first motivate the need for using multiple types of annotations (Section 3.1.2.1). Then I will define the value of information based criterion for scoring candidate annotations (Section 3.1.2.2) based on the expected reduction in risk (Section 3.1.2.3) and the manual effort cost (Section 3.1.2.4) and summarize the active learning algorithm (Section 3.1.3).

3.1.2.1 Illustration: Need for Comparing Multiple Types of Annotations

Traditional active learning methods assume equal manual effort per label, and thus try to minimize the total number of queries made to the annotator. In reality annotation costs will vary substantially from image to image, and from type to type. Thus, the standard “flat cost” implied by traditional active learners is inadequate.

To illustrate this idea more concretely, we ran an experiment where we measured both the reduction in misclassification risk produced by adding an annotation with its correct label from an unlabeled pool of images and the time to obtain the annotation. The misclassification risk is defined in the standard way, as the probability of classifying each example with an incorrect label, summed over all examples. Figure 3.7 shows this result for all examples in the unlabeled pool with the three annotation types (segmentations, image tags and region labels) for two different sizes of the initial training set (5 and 100 image tags respectively).

The figures suggest that neither more expensive nor less expensive examples are regularly more useful than the other. Similarly, the annotation that provides the best reduction in risk might not be the most effective in terms of the cost of obtaining it. For example, in Figure 3.7 (right) there are examples from all three annotation types with reductions in risk above 200 units. While a standard “flat cost” active learner would choose the more expensive segmentation (because of the marginally higher reduction in risk) a cost-sensitive learner might choose the less expensive one.

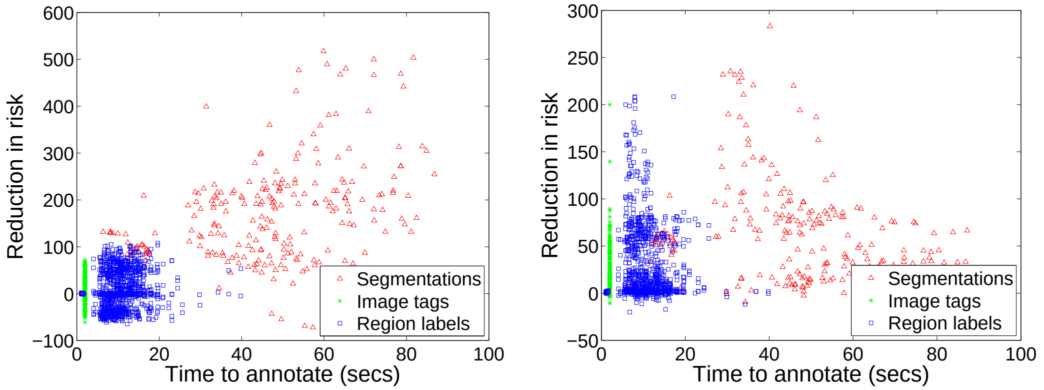


Figure 3.7: This figure shows the reduction in risk for each example in the unlabeled pool plotted against the time required to provide an annotation after training with 5 image tags (**left**) and 100 image tags (**right**). There is not an absolute correlation between the cost of an annotation and how informative it is, motivating the use of cost-sensitive active learning.

The figures also illustrate that while segmentations are indeed more expensive to obtain, the larger reductions in risk can effectively mitigate the cost for several examples. In addition, the relative risk reduction versus the annotation time required is a function that continually changes as more annotated data is acquired, as evident when we compare the total shape of the scatter plots on the left (where only 5 examples have been seen per class) and on the right (where 100 examples have been seen per class). Hence, to best reduce human involvement, the active learner needs a quantitative measure of the effort required to obtain any given annotation.

3.1.2.2 Defining the Value of Information for an Annotation

Thus, inspired by the classic notion of the *value of information* (VOI), and by previous binary single-label active learners ([55]), we derive a measure to gauge the relative risk reduction a new multi-label annotation may provide. The main idea is to evaluate the candidate images and annotation types, and predict which combination (of image+type) will lead to the greatest net decrease in risk for the current classifier, when each choice is penalized according to its expected manual effort. In contrast to previous VOI methods, our measure enables the multi-label setting and considers multiple types of annotations to select from.

At any stage in the learning process the dataset can be divided into three different pools: $\mathcal{X}_U$, the set of unlabeled examples (bags and instances); $\mathcal{X}_L$, the set of labeled examples; and $\mathcal{X}_P$, the set of partially labeled examples, which contains all bags for which we have only a partial set of bag-level labels (refer back to Figure 3.4). If the label on an image is considered to be a binary vector of length C, then the images in $\mathcal{X}_L$ are examples where the binary label vector is completely known. Images in $\mathcal{X}_U$ are examples where none of the labels are known, and images in $\mathcal{X}_P$ are examples where some of the elements in the vector and labels on some of its instances are known. An example is moved from $\mathcal{X}_U$ to $\mathcal{X}_P$ when any one of its unknown labels is requested. An example is moved from $\mathcal{X}_P$ to $\mathcal{X}_L$ only when the labels on all its instances have been obtained.

The total cost $T(\mathcal{X}_L, \mathcal{X}_U, \mathcal{X}_P)$ associated with a given snapshot of the data is the total misclassification risk, plus the cost of obtaining all the labeled data thus far:

$$T(\mathcal{X}_L, \mathcal{X}_U, \mathcal{X}_P) \;=\; \mathcal{R}(\mathcal{X}_L) + \mathcal{R}(\mathcal{X}_U) + \mathcal{R}(\mathcal{X}_P) + \sum_{X_i \in \mathcal{X}_B} \sum_{l \in L_i} \mathcal{C}(X_i^l), \qquad (3.2)$$

where $\mathcal{X}_B = \mathcal{X}_L \cup \mathcal{X}_P$, and $\mathcal{C}(\cdot)$ is defined in Section 3.1.2.4.

We measure the utility of obtaining a particular annotation by predicting the change in total cost that would result from the addition of the annotation to $\mathcal{X}_L$. Therefore, the value of information for an annotation $\mathbf{z}$ is:

$$\begin{aligned} VOI(\mathbf{z}) &= T\left(\mathcal{X}_L, \mathcal{X}_U, \mathcal{X}_P\right) - T\left(\hat{\mathcal{X}}_L, \hat{\mathcal{X}}_U, \hat{\mathcal{X}}_P\right) && (3.3) \\ &= \mathcal{R}(\mathcal{X}_L) + \mathcal{R}(\mathcal{X}_U) + \mathcal{R}(\mathcal{X}_P) - \left(\mathcal{R}(\hat{\mathcal{X}}_L) + \mathcal{R}(\hat{\mathcal{X}}_U) + \mathcal{R}(\hat{\mathcal{X}}_P)\right) - \mathcal{C}(\mathbf{z}) && (3.4) \end{aligned}$$

where $\hat{\mathcal{X}}_L, \hat{\mathcal{X}}_U, \hat{\mathcal{X}}_P$ denote the set of labeled, unlabeled and partially labeled data after obtaining annotation $\mathbf{z}$. Note that $\mathbf{z}$ could be any one among the three annotation types described in Figure 3.6. If all the labels on the example have been obtained through $\mathbf{z}$ then the example is moved to the labeled pool, i.e., $\hat{\mathcal{X}}_L = \mathcal{X}_L \cup \mathbf{z}$. On the other hand, if the example contains instances (regions) with no label information even after obtaining annotation $\mathbf{z}$ then the example is moved to the set of partially labeled data, i.e., $\hat{\mathcal{X}}_P = \mathcal{X}_P \cup \mathbf{z}$. Similarly, the example associated with $\mathbf{z}$ is removed from $\mathcal{X}_U$ or $\mathcal{X}_P$ as appropriate.

Thus, for each candidate, the selection function measures its expected informativeness and subtracts its predicted cost. A high VOI for a given input denotes that the total cost would be decreased by adding its annotation and is therefore most useful given its cost. So at every iteration our approach computes the VOI of all candidate image annotations present in $\mathcal{X}_P$ and $\mathcal{X}_U$ and chooses the example with the largest VOI for querying. Once the annotation is obtained it is moved from $\mathcal{X}_P/\mathcal{X}_U$ to $\mathcal{X}_L/\mathcal{X}_P$ as appropriate and the classifier is retrained.

Thus far we have defined our VOI based selection function to do active selection among candidate annotations. In the following sections, we first address how to predict informativeness (Section 3.1.2.3) followed by cost (Section 3.1.2.4).

3.1.2.3 Predicting the Informativeness of an Annotation

Let r_l denote the risk associated with misclassifying an example belonging to class l. The risk associated with $\mathcal{X}_L$ is:

$$\mathcal{R}(\mathcal{X}_L) = \sum_{X_i \in \mathcal{X}_L} \sum_{l \in L_i} r_l \left(1 - p(l|X_i)\right), \tag{3.5}$$

where $p(l|X_i)$ is the probability that X_i is classified with label l. Here, X_i is again used to denote both instances and bags and L_i its label(s). If X_i is a training instance it has only one label, and we can compute $p(l|X_i)$ via the current MIML classifier.

If X_i is a multi-label bag in the training set, we compute the probability it receives label l as follows:

$$p(l|X_i) = p\left(l|x_1^i, \ldots, x_{n_i}^i\right) = 1 - \prod_{j=1}^{n_i}(1 - p(l|x_j^i)). \tag{3.6}$$

For a bag to *not* belong to a class, it must be the case that none of its instances belong to the class. Thus the probability of a bag *not* having a label is equivalent to the probability that *none* of its instances have that class label.

The MIML classifier implicitly assumes that every image/instance can be classified into one of C labels. However, in the more general case, the dataset can also contain images that do not necessarily belong to the C classes. Such images are given a “negative” label, which specifies that none of the instances/regions in the image belong to any of the classes in $\{1, \ldots, C\}$, similar to the “negative” label in a standard MIL formulation. In this case, we weight $p(l|X_i)$ with the probability of X_i belonging to any one of the C classes as against the “negative” class, which is obtained by training a standard MIL classifier. Note that when $C = 1$, a single foreground class, the above reduces to the standard MIL solution since $p(l|X_i)$ is trivially 1. Similarly, in the absence of a "negative" class the above reduces to the MIML solution.

The corresponding risk for the unlabeled data is:

$$\mathcal{R}(\mathcal{X}_U) = \sum_{X_i \in \mathcal{X}_U} \sum_{l=1}^{C} r_l (1 - p(l|X_i)) \Pr(l|X_i), \tag{3.7}$$

where we compute the probabilities for bags using Equation 3.6, and $\Pr(l|X_i)$ is the true probability that unlabeled example X_i has label l, approximated as $\Pr(l|X_i) \approx p(l|X_i)$.

For the partially labeled data, the risk is:

$$\mathcal{R}(\mathcal{X}_P) \;\; = \;\; \sum_{X_i \in \mathcal{X}_P} \sum_{l \in L_i} r_l \, (1 - p(l|X_i)) + \sum_{l \in U_i} r_l \, (1 - p(l|X_i)) \, p(l|X_i), \tag{3.8}$$

where $U_i = \mathbb{L} \setminus L_i$.

The value r_l is the risk associated with misclassifying an example belonging to class l, specified in the same units as the cost function in Section 3.1.2.4. Intuitively, it should reflect the real cost of a classification mistake, as our algorithm directly trades off the cost of the manual labeling against the damage done by misclassification. While this can be set based on realistic system requirements, we interpret it as the cost of manually fixing a classification error (e.g., an average segmentation requires 50 seconds). If one preferred to avoid errors on a particular class, that could be encoded with variable r_l values per class label l. Note that r_l is not a parameter that needs to be optimized for performance; rather, it gives flexibility for situations that have real costs associated with the task.

The VOI function relies on estimates for the risk of yet-unlabeled data, so we must predict how the classifier will change given the candidate annotation, without actually knowing its label(s). We estimate the total risk induced by incorporating a candidate annotation $\mathbf{z}$ using the expected value:

$$\mathcal{R}(\hat{\mathcal{X}}_L) + \mathcal{R}(\hat{\mathcal{X}}_U) + \mathcal{R}(\hat{\mathcal{X}}_P) \approx E[\mathcal{R}(\hat{\mathcal{X}}_L) + \mathcal{R}(\hat{\mathcal{X}}_U) + \mathcal{R}(\hat{\mathcal{X}}_P)] \tag{3.9}$$

If the annotation $\mathbf{z}$ will label an unlabeled instance (Figure 3.6(b)), computing the expectation is straightforward, since that instance can simply be removed from $\mathcal{X}_U$ and added to $\mathcal{X}_L$ to evaluate the risk were it assigned each of the L possible labels in turn:

$$E[\mathcal{R}(\hat{\mathcal{X}}_L) + \mathcal{R}(\hat{\mathcal{X}}_U) + \mathcal{R}(\hat{\mathcal{X}}_P)] = \sum_{l \in \mathbb{L}} \left(\mathcal{R}(\mathcal{X}_L \cup \mathbf{z}^{(l)}) + \mathcal{R}(\{\mathcal{X}_U, \mathcal{X}_P\} \setminus \mathbf{z}) \right) \Pr(l|\mathbf{z}), \tag{3.10}$$

where $\mathbb{L} = \{1, \ldots, C\}$ is the set of all possible label assignments for $\mathbf{z}$. The value $\Pr(l|\mathbf{z})$ is obtained by evaluating the current classifier on $\mathbf{z}$ and mapping the output to the associated posterior, and risk is computed based on the (temporarily) modified classifier with $\mathbf{z}^{(l)}$ inserted into the labeled set. Similarly, if the candidate annotation $\mathbf{z}$ will add an image-level label to an unlabeled or partially labeled bag (Figure 3.6(a)), then $\Pr(l|\mathbf{z})$ is calculated using Equation 3.6.

However, if the annotation $\mathbf{z}$ entails fully segmenting and labeling an image with M automatically segmented regions (Figure 3.6(c)), we need to calculate the utility of obtaining the joint set of labels for all of a bag's instances. Since there are C^M possible labelings: $\mathbb{L} = \{1, \ldots, C\}^M$, a direct computation of the expectation is impractical. Instead we use Gibbs sampling to draw samples of the label assignment from the joint distribution over the M instances' descriptors. Let $\mathbf{z} = \{z_1, \ldots, z_M\}$ be the bag's instances, and let $\mathbf{z}^{(\mathbf{a})} = \left\{ (z_1^{(a_1)}), \ldots, (z_M^{(a_M)}) \right\}$ denote the label assignment we wish to sample, with $a_j \in \{1, \ldots, C\}$. To sample from the conditional distribution of one instance's label given the rest—the basic procedure required by Gibbs sampling—we re-train the classifier with the given labels added, and then draw the remaining label according to $a_j \sim \Pr(l|z_j)$, for $l \in \{1, \ldots, C\}$, where z_j denotes the one instance currently under consideration. For bag $\mathbf{z}$, the expected total risk is then the average risk computed over all samples:

$$\begin{aligned} E[\mathcal{R}(\hat{\mathcal{X}}_L) + \mathcal{R}(\hat{\mathcal{X}}_U) + \mathcal{R}(\hat{\mathcal{X}}_P)] &= \frac{1}{S}\sum_{k=1}^{S}(\mathcal{R}(\{\mathcal{X}_L \setminus \mathbf{z}\} \cup \{z_1^{(a_1)_k}, \ldots, z_M^{(a_M)_k}\}) \\ &+ \mathcal{R}(\mathcal{X}_U \setminus \{z_1, z_2, \ldots, z_M\}) + \mathcal{R}(\mathcal{X}_P)), \qquad (3.11) \end{aligned}$$

where k indexes the S samples. We compute the risk on $\mathcal{X}_L$ for each fixed sample by removing the bag $\mathbf{z}$ from the unlabeled or partially labeled pool, and inserting its instances with the label given by the sample's label assignment. Note that while computing the VOI of a candidate annotation we have no supervision information on that example, including the object outlines. Hence, the computation of VOI is performed using segments/regions generated using an automatic segmentation algorithm. Once we obtain a complete segmentation of an image from the annotator, we use the actual region outlines and labels to retrain the classifier.

Computing the VOI values for all unlabeled data, especially for the positive bags, requires repeatedly solving the classifier objective function with slightly different inputs; to make this manageable we employ incremental SVM updates [17].

3.1.2.4 Predicting the Cost of an Annotation

Given the expected reduction in risk defined in the previous section, we still need to define the cost of an annotation in order to compute its VOI as defined in Section 3.1.2.2. We define the cost of an annotation based on how much time a human annotator might require to provide

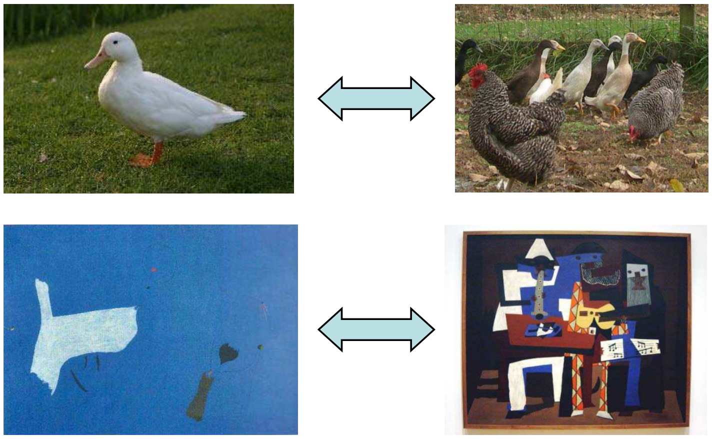


Figure 3.8: Which image would you rather annotate? Humans can easily glance at an image and roughly gauge the difficulty. This appears to be true even without prior knowledge about the specific objects present in the image (second row).

it. However, since we cannot directly obtain an annotation's cost without first obtaining the annotation itself, we require a method to predict the cost of an annotation given an image.

Thus, the goal in this section is to accurately predict annotation time based on image content alone—that is, without actually obtaining the annotation, we need to estimate how long it will take a typical annotator to complete it. As Figure 3.8 suggests, humans are able to predict the difficulty of annotating an image even without prior knowledge about the objects occurring in the image (second row) or other high-level cues. Therefore, it seems plausible that the difficulty level of an image could be predicted based on the image's low-level features. For an extreme example, if an image contains a single color it most likely contains only one object, and so it should not be difficult to segment. If the image has significant responses to a large number of filters, then it may be highly cluttered, and so it could take a long time.

Thus, we propose to use supervised learning to estimate the difficulty of segmenting an image. It is unclear what features will optimally reflect annotation difficulty, and admittedly high-level recognition itself plays some role. We select candidate low-level features, and then use multiple kernel learning to select those most useful for the task. Multiple kernel learning approaches automatically select the weights on the various features (kernels) by posing the problem as an optimization of the coefficients of such a combination. This reduces to a convex optimization problem known as a quadratically-constrained quadratic program (QCQP) as shown in [63]. In [3], a novel dual formulation of the corresponding QCQP as a second-order cone programming problem is proposed to yield a formulation for which the sequential minimal optimization (SMO) algorithm can be applied. We use this SMO algorithm to select cost-predictive features, since it allows efficient solutions for large-scale problems.

We begin with some generic features that may be decent indicators of image complexity: a histogram of oriented gradients, a gray-scale histogram, and two new features based on the edge density and color uniformity. The features are designed to exploit the fact that more objects could lead to more annotation time.

- The edge density feature divides the image into a hierarchical grid of cells and concatenates the edge density within each cell into a feature vector. We reason that edge density could be a good indicator of the number of objects, since with a larger number of objects in an image there are bound to be more edges separating them. The hierarchy, by capturing edge densities at multiple scales, helps in dealing with objects of different scales.
- The color uniformity feature computes the standard deviation of the r, g, b values of every pixel in the image based on a small neighborhood surrounding it, and obtains

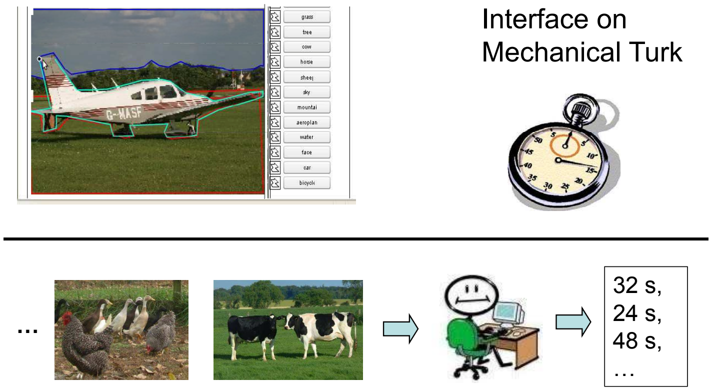


Figure 3.9: Our interface on Mechanical Turk to collect annotation times for segmenting images from anonymous users. The system times the responses as users use a polygon-drawing tool to superimpose object boundaries, and name and outline every major object.

a histogram of the standard deviations. With more objects we expect larger standard deviations in a neighborhood compared to a small number of smoothly varying regions such as sky or grass.

We gather the data online, using Amazon's Mechanical Turk system, where we can pay anonymous users to segment images of our choosing. The users are given a polygon-drawing tool to superimpose object boundaries, and are instructed to name and outline every major object (see Figure 3.9). The system times their responses. Thus the labels on the training images will be the times that annotators needed to complete a full annotation. To account for noise in the data collection, we collect a large number of user responses per image. Even if users generally have the same relative speeds (faster on easy ones, slower on harder ones), their absolute speeds may vary. Therefore, to make the values comparable, we normalize each user's times by his/her mean and use the average time taken on an image to be its target label.

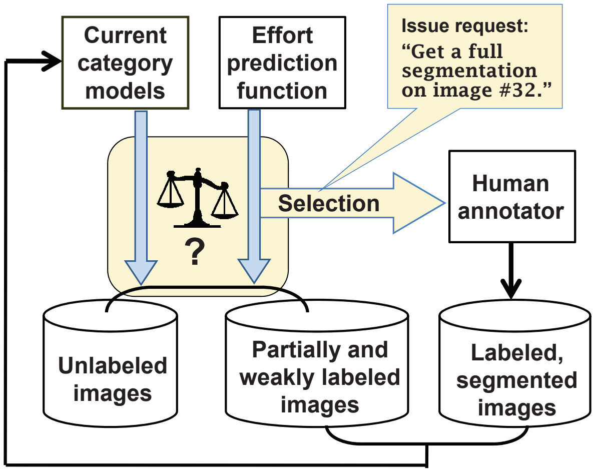

Current category models
Effort prediction function
Issue request:
"Get a full segmentation on image #32."
?
Selection
Human annotator
Unlabeled images
Partially and weakly labeled images
Labeled, segmented images

Figure 3.10: The summary of our multi-level active learning approach. After learning from a small initial set of labeled images, our method surveys any available unlabeled and partially labeled data. The VOI of every candidate annotation among three different types of annotations is computed using the expected change in risk and the predicted effort of obtaining the annotation given by our cost predictor. The annotation and example with the largest VOI is then selected and a human provides the annotation, after which the example is moved from the unlabeled/partially labeled pool to the partially/fully labeled pool as appropriate. The process repeats until there are no more examples with positive VOI, or once the allowed annotation cost limit has been reached.

We construct a χ^2 RBF kernel over the training examples per image feature. Based on the timing obtained from the anonymous users we divide the set of training images into a discrete range of “easy” and “hard” images using the mean time over all the images. We then use the MKL approach of [3] to learn the weights on the image features for the binary classification problem of classifying images into “easy” and “hard” categories. Using the obtained combined kernel, we also learn a cost predictor function using Support Vector Regression (SVR).

From this we can build a cost function $\mathcal{C}(\mathbf{z})$ that takes a candidate annotation $\mathbf{z}$ as input, and returns the predicted time requirement (in seconds) as output. When $\mathbf{z}$ is a candidate full segmentation, we apply the learned function to the image. When $\mathbf{z}$ is a request for a tag (bag-level label), we set $\mathcal{C}(\mathbf{z})$ as the cost estimated using similar time-based experiments. Finally, when $\mathbf{z}$ entails outlining a single object, we estimate the cost as the full image’s predicted time, divided by the number of segments in the image.

3.1.3 Summary of the Algorithm

We can now actively select multi-label, multi-level image annotations so as to maximize the expected benefit relative to the manual effort expended. The MIML classifier is initially trained using a small number of tagged images. To get each subsequent annotation, the active learner surveys all remaining unlabeled and partially labeled examples, computes their VOI, and requests the label for the example with the maximal value. After the classifier is updated with this label, the process repeats. Figure 3.10 provides a high-level summary of the approach. The final classifier can predict image- and region-level labels, in binary or multi-class settings.

3.2 Results

In the following subsections, we evaluate five aspects of our approach: (1) its accuracy when learning from multi-label examples, (2) its ability to accurately predict annotation costs, (3) its effectiveness as an active learner when selecting from three different types of annotations on both binary and multi-label problems, (4) the effect of introducing the cost predictor in the active selection function, and (5) the robustness of our approach with respect to the initial training set.

3.2.1 Datasets and Implementation Details

To validate our method we use three publicly available datasets, the SIVAL[4] dataset, the Google dataset [31] and the MSRC [5] dataset, since they have been used to evaluate previous MIL and MIML based approaches, which allows us to compare with state-of-the-art methods in the two settings. Additionally, the MSRC is a common benchmark for multi-class object segmentation.

- The SIVAL dataset contains about 1500 images from 25 objects. The cluttered images contain objects in a variety of positions, orientations, locations, and lighting conditions. See Figure 3.11 for examples. The images have been oversegmented into about 30

---

[4] http://www.cs.wustl.edu/accio/

[5] http://research.microsoft.com/en−us/projects/objectclassrecognition/

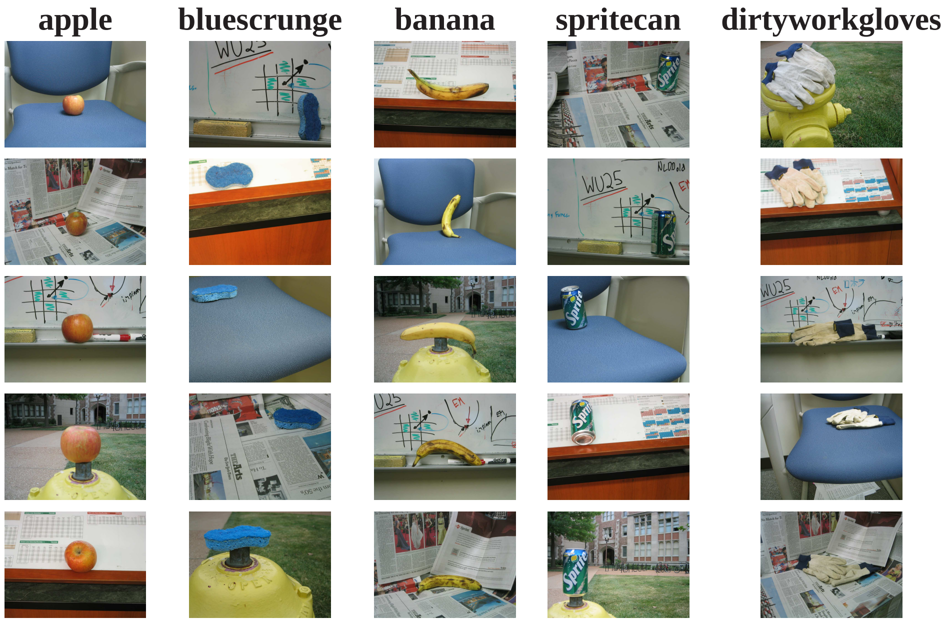


Figure 3.11: Example images from the SIVAL dataset. Each column illustrates one of the 25 objects.

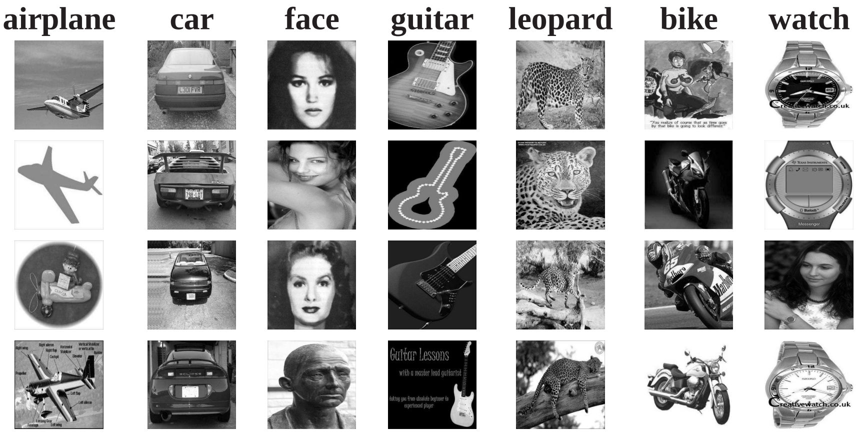


Figure 3.12: Example images from Google downloaded dataset. Each column shows images downloaded using a particular category name. Since the images are from keyword search not all images belong to the category of interest (e.g. row 4, column 1 is not an airplane).

regions (instances) each, each of which is represented by a 30-dimensional feature capturing the average color and texture values of the segment and each of its cardinal neighbors. These features are provided with the SIVAL dataset [6].

- The MSRC v2 contains 591 images from 21 classes and a variable number of objects per image, with 240 images and 14 classes in the (subset) v1. See Figure 3.13 for examples. In all MSRC experiments we use an RBF kernel with $\gamma = 10$, and set the SVM parameters (including the sigmoid parameters for the SVM probabilistic outputs given by the method of [76]) based on cross-validation. We ignore all "void" regions in the MSRC images. We segment the images with Normalized Cuts into a small number of segments (10 in our experiments). For each segment we then obtain texton and color histograms, as in [92]. We learn a dictionary of textons by convolving the images with a 38-dimensional filter bank and running K-means clustering to obtain 420 textons. For color histograms we obtain a 120-dimensional vector by concatenating a 40-dimensional histogram of each channel of the LUV representation of the image.

- The Google dataset [31] contains on average 600 examples each for seven object categories. Since the images are from a keyword search, the true number of training examples for each class are sparse: on average 30% contain a "good" view of the class of interest, 20% are of "ok" quality (extensive occlusions, image noise, cartoons, etc.), and 50% are completely unrelated "junk", as judged in [31]. Some example images from this dataset are shown in Figure 3.12; each column is a sample of images from a different keyword. To form positive bags from these images, we must group them into

---

[6] http://www.cs.wustl.edu/$\sim$sg/accio/SIVAL.html

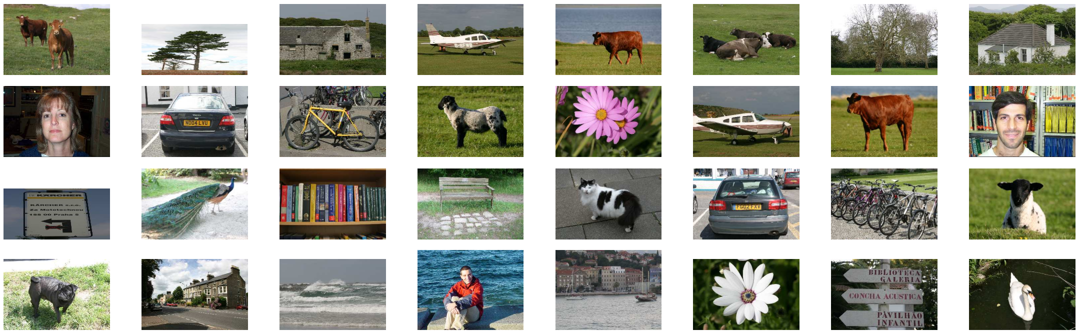


Figure 3.13: Example images from the MSRC dataset. The MSRC dataset contains 21 categories, and most images have multiple categories in them (e.g.: “building”, “road”, “sky”, “tree”).

multiple sets. Given the percentage of true positives, random selections of bags of size 25 are almost certain to contain at least one.

The SIVAL and MSRC datasets handle the main scenario discussed above; multi-label object segmentation and recognition. The Google dataset handles the second MIL scenario (see Section 3.1.1.1).

3.2.2 Multi-label Visual Category Learning with the MSK

In our first experiment, we evaluate our proposed Multi-label Set Kernel (MSK) classifier’s effectiveness in learning using only image-level labels on images containing multiple objects. We divide the MSRC v2 into five folds containing about an equal number of images, as is done by [129]. We choose one part as the test set, one to set parameters, and train on the rest.

Each image is a bag, and each segment is an instance. To learn the MIML classifier, we use only image-level (bag-level) labels, i.e., we withhold all the pixel-level labels during classifier

Approach	Ave. AUROC (img)	Ave. AUROC (region)
Ours	0.896 ± 0.00	0.91 ± 0.01
Zha et al. 2008 [129]	0.902	0.863

Table 3.1: Five-fold cross-validation accuracy when training with only image-level labels.

training. We first compare against the approach of [129], who provide state-of-the-art results on the MSRC dataset while learning from image-level labels.

Table 3.1 shows the average AUROC when predicting labels on new *images* (second column) or new *regions* (third column). We use AUROC to evaluate accuracy because it is the most appropriate measure for binary classification and it allows us to compare our results with existing state-of-the-art methods such as [129]. For image-level prediction our results are comparable to the state-of-the-art for MIML [129], whereas for region-level prediction we achieve a notable improvement (0.91 vs. 0.86). This appears to be a direct consequence of our Multi-label Set Kernel, which weighs the region descriptors so as to represent an image by its most relevant instances for each image-level label. As a result, we are able to directly separate novel regions from each class within a new image, and not just name objects that occur in it.

Approach	Supervision	Accuracy (%)
Winn et al. 2005 [121]	Pixel-level	67.6
Shotton et al. 2006 [92]	Pixel-level	70.5
Ours	Image-level	64.1 ± 2.9
Ours	Region-level	66.3

Table 3.2: Region-level multi-class classification accuracies compared to state-of-the-art approaches that use pixel-level information for training.

Next we compare against the approaches of [92] and [121], which use pixel-level labels (full segmentations) to train multi-class classifiers. We evaluate all approaches using the region-level multi-class classification accuracy. While [121] learn a generative model of texton histograms, [92] learn a discriminative CRF model incorporating appearance, shape and context information. A comparison with these approaches would tell us how important stronger supervision is on this dataset and how effectively our multi-label classifier is able to utilize image-level labels. Table 3.2 compares the region-level multi-class accuracies obtained over five trials of approximately equal train-test splits. Thus with much less manual training effort (image tags), our method performs quite competitively with methods trained with full segmentations; this illustrates the advantage of the multi-label multi-instance learner in effectively utilizing weaker supervision. Using both region- and bag-level labels we obtain an accuracy of 66.3%. This seems to suggest that selectively obtaining region-level labels should further improve our classifier.

Finally, using the NSK ([37]), which essentially removes our kernel weight mapping, the accuracy for this test would only be $55.95\% \pm 1.43$. This result indicates that the proposed method to map different regions to the image-level labels is more effective.

3.2.3 Multi-level Active Selection for Learning Object Categories

In this section we demonstrate our approach to actively learn visual categories for both the binary setting, where an image contains a single object of interest in a cluttered background as well as the multi-label setting, where an image contains multiple familiar objects that must be segmented and classified. We test all three datasets described above.

We provide results by simulating the active learning process: when the system requests an annotation on an image example, we satisfy the request using the ground truth labels. For instance, when the request is to outline all the objects in the image, we use the ground truth segmentation provided with the dataset (SIVAL/MSRC) to obtain all the objects and their labels. However, recall that when calculating the VOI of a region/image, the system uses an automatic low-level segmentation of the image.

3.2.3.1 Active Selection from MIL Data

For the binary MIL setting, we provide comparisons with single-level active learning (with both the method of [89], and where the same VOI function is used but is restricted to actively label only instances), as well as passive learning. For the passive baseline, we consider random selections from amongst both single-level and multi-level annotations, in order to verify that our approach does not simply benefit from having access to more informative possible labels.

For Gibbs sampling, we generate $S = 25$ samples with an initial burn-in period of 50 samples. This number was set arbitrarily; later experiments increasing the sample size to 50 did not improve results significantly, though in general larger samples should yield more accurate VOI estimates. The risk parameter (r_l) and the cost of labeling a single instance are all set to 1, meaning we have no preference for false positives or false negatives, and that we view a misclassification to be as harmful as requiring a user to label one instance.

We evaluate our approach for the two MIL classification scenarios explained in Figure 3.3. To recall, in the first scenario training images are represented by a bag of regions and a positive image contains at least one of the regions containing the object of interest. The goal is to predict when new image regions contain the object—that is, to learn to label regions as foreground or background. In the second scenario, groups of images downloaded from keyword searches from multiple search engines are positive bags and individual images are the instances. The goal is to predict the presence or absence of the category in new images.

Actively Learning Visual Objects and their Foreground Regions. We use the SIVAL dataset for evaluating the first MIL scenario. Thus each image is a bag containing both positive and negative instances (segments). Labels on the training data specify whether the object of interest is present or not, but the segments themselves are unlabeled (though the dataset does provide ground truth segment labels for evaluation purposes). We again report accuracy using the AUROC measure since classification on SIVAL is a binary task.

As the SIVAL dataset contains exactly one object per image (see Figure 3.11), we do not expect the segmentation costs to vary on a per example basis. Therefore, for this dataset we attribute a single cost to all annotations of a particular type. To determine how much more labeling a positive bag costs relative to labeling an instance, we performed a user study. Users were shown oversegmented images and had to click on all the segments belonging to the object of interest. The baseline task was to provide a present/absent flag on the images. For segmentation, obtaining labels on all positive segments took users on average four times as much time as setting a flag. Thus we set the cost of labeling a positive bag to 4 for the SIVAL data. The value agrees with the average sparsity of the dataset: the SIVAL set contains about 10% positive segments per image. The users who took part in the experiment were untrained but still produced consistent results.

The initial training set is comprised of 10 positive and 10 negative images per class, selected

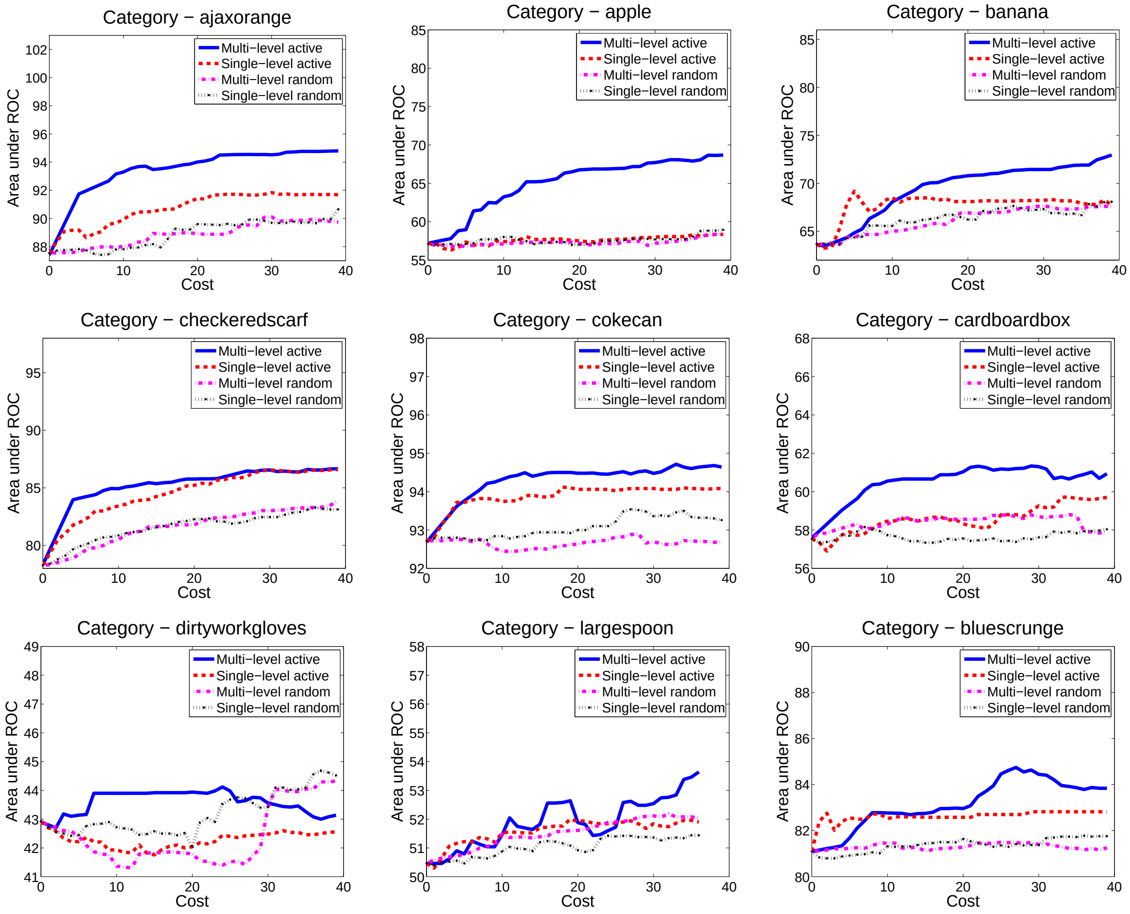
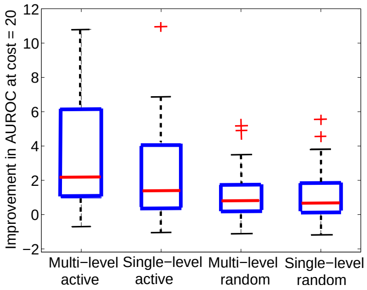


Figure 3.14: Results on the SIVAL dataset. Sample learning curves per class, each averaged over five trials. Our method corresponds to the “Multi-level active” curves. First six are best examples, last three are worst. For the same amount of annotation cost, our multi-level approach learns more quickly than both traditional single-level active selection as well as both forms of random selection.



Cost	**Our Approach**			**Settles et al. [89]**		
	Random	Multi-level Active	Gain over Random (%)	Random	MIU Active	Gain over Random (%)
10	+0.0051	+0.0241	372	+0.023	+0.050	117
20	+0.0130	+0.0360	176	+0.033	+0.070	112
50	+0.0274	+0.0495	81	+0.057	+0.087	52

Figure 3.15: **Left:** Summary of the average improvement over all 25 SIVAL categories after half of the annotation cost is used. **Right:** Comparison with [89] on the SIVAL data, as measured by the average improvement in the AUROC over the initial model for increasing labeling cost values.

at random. Our active learning method must choose its queries from among 10 positive bags (complete segmentations), 300 unlabeled instances (individual segments), and about 150 unlabeled bags (present/absent flag on the image). We use a quadratic kernel, $K(x, y) = (1 + \alpha\varphi(x)^T\varphi(y))^2$, with a coefficient of $\alpha = 10^{-6}$, and average results over five random training partitions.

Figure 3.14 shows representative (best and worst) learning curves for our method and the three baselines, all of which use the same MIL classifier (NSK-SVM). Note that the curves are plotted against the cumulative *cost* of obtaining labels—as opposed to the number of queried instances—since our algorithm may choose a sequence of queries with non-uniform cost. All methods are given a fixed amount of manual effort (40 cost units) and are allowed to make a sequence of choices until that cost is used up. Recall that a cost of 40 could correspond, for example, to obtaining labels on $\frac{40}{1} = 40$ instances or $\frac{40}{4} = 10$ positive bags, or some mixture thereof. Figure 3.15 (left) summarizes the learning curves for all categories, in terms of the average improvement at a fixed point midway through the active learning phase.

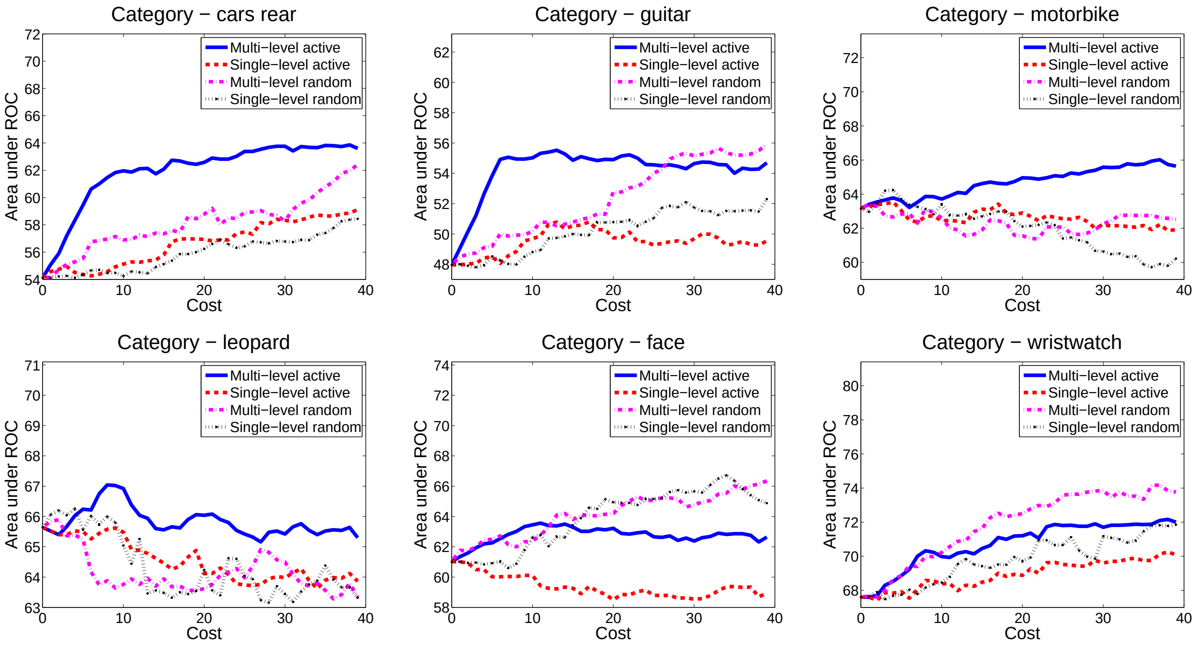
All four methods steadily improve upon the initial classifier, but at different rates with respect to the cost. (All methods fail to do better than chance on the ‘dirty glove’ class, which we attribute to the lack of distinctive texture or color on that object.) In general, a steeper learning curve indicates that a method is learning most effectively from the supplied labels. Our multi-level approach shows the most significant gains at a lower cost, meaning that it is best suited for building accurate classifiers with minimal manual effort on this dataset. As we would expect, single-level active selections are better than random, but still fall short of our multi-level approach. This is because single-level active selection can only make a sequence of greedy choices while our approach can jointly select bags of instances to query. Interestingly, multi- and single-level random selections perform quite similarly on this dataset (see boxplots in Figure 3.15 (left)), which indicates that having more unambiguous labels alone does not directly lead to better classifiers unless the right instances are queried.

At a cost of 24 units the mean AUROC over all 25 classes for active selection turned out to be 0.723, which is 92% of the accuracy achievable if using *all* the labels and examples in the unlabeled pool. To reach the same accuracy random selection requires 44 units of cost. This means that to reach 92% of the upper-bound accuracy, active selection requires 45.5% less annotation cost than the passive learner.

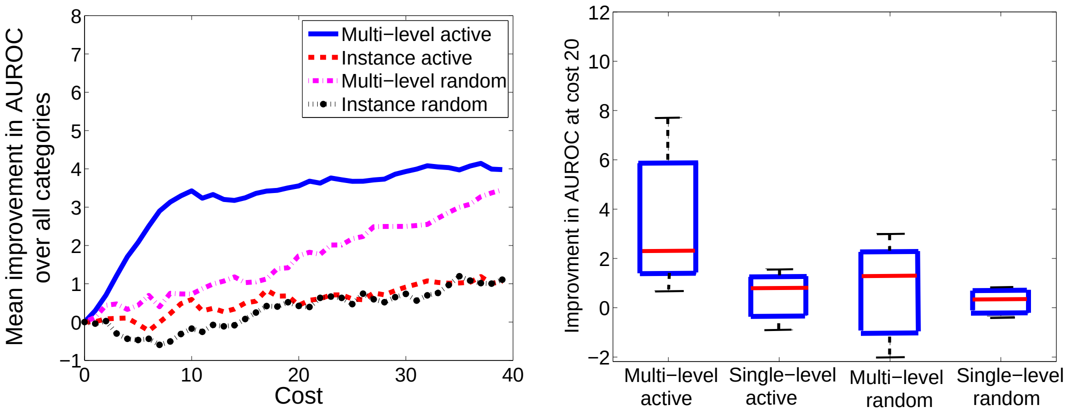
The table in Figure 3.15 compares our results to those reported in [89], in which the authors train an initial classifier with *multiple-instance logistic regression*, and then use the MI Uncertainty (MIU) to actively choose instances to label. To our knowledge this is the only other existing approach to perform active selections with MIL data, making it a useful method to compare to. Following [89], we report the average gains in the AUROC over all categories at fixed points on the learning curve, averaging results over 20 trials and with the same initial training set of 20 positive and negative images. Since the accuracy of the base classifiers used by the two methods varies, it is difficult to directly compare the gains in the AUROC. The NSK-SVM we use consistently outperforms the logistic regression approach using only the initial training set; even before active learning our average accuracy is 68.84, compared to 52.21 in [89]. Therefore, to aid in comparison, we also report the percentage gain relative to random selection, for both classifiers. The results show that our approach yields much stronger relative improvements, again illustrating the value of allowing active choices at multiple levels (the method of [89] only allows active queries for instance-level labels). For both methods, the percent gains decrease with increasing cost; this makes sense, since eventually (for enough manual effort) a passive learner can begin to catch up to an active learner.

Actively Learning Visual Categories from Web Images. Next we evaluate the scenario where each positive bag is a collection of images, among which only a portion are actually positive instances for the class of interest. Previous methods have shown how to learn from noisy Web images, with results rivaling state-of-the-art supervised techniques [106, 31, 70]. We show how to boost accuracy (AUROC) with these types of learners while leveraging minimal manual annotation effort.

To re-use the publicly available dataset from [31], we randomly group Google images into bags of size 25 to simulate multiple searches as in [106], yielding about 30 bags per category. We randomly select 10 positive and 10 negative bags (from all other categories) to serve as the initial training data for each class. The rest of the positive bags of a class are used to construct



(a) Example learning curves per class



(b) Summary: all classes

Figure 3.16: Results on the Google dataset, in the same format as Figure 3.14. Our multi-level active approach outperforms both random selection strategies and the single-level active method.

the test sets. All results are averaged over five random partitions. We represent each image as a bag of "visual words", and compare examples with a linear kernel. Our method makes active queries among 10 positive bags (complete labels) and about 250 unlabeled instances (images). There are no unlabeled bags in this scenario, since every downloaded batch is associated with a keyword.

Figure 3.16 shows the learning curves and a summary of our active learner's performance. Our multi-level approach again shows more significant gains at a lower cost relative to all baselines, improving accuracy with as few as ten labeled instances. On this dataset, random selection with multi-level annotations actually outperforms random selection on single-level annotations (see the boxplots). We attribute this to the distribution of bags/instances: on average more positive bags were randomly chosen, and each addition led to a larger increase in the AUROC.

3.2.3.2 Active Selection from MIML Data

In the previous section we considered active selection in the binary setting when the image contains a single object among background clutter or with sets of noisy images obtained by keyword searches. Next we use the MSRC dataset to demonstrate the impact of using our multi-label active selection function in the more general multi-label setting, where an image contains multiple objects of interest plus clutter, and selections can be made from different types of annotations.

We divide the examples into five folds containing an equal number in each and use the first part for training and the rest for testing. We construct the initial training set such that each class appears in at least five images, and use image-level labels. The rest of the training set forms the unlabeled pool of data. The active learner can request either complete segmentations or region-level labels from among the initial training examples, or image-level labels from any unlabeled example. We set $r_l = 50$ for all classes, which means that each misclassification is worth 50 *s* of user time. The parameter r_l should reflect the real cost of a classification mistake. Our choice of the value of r_l is based on the fact that an error made by the automatic labeling would take around 50 *s* to manually fix for the average image. For this experiment we fix the costs per type using the mean times from real users: 50 *s* for complete segmentations, 10 *s* for a region outline, and 3 *s* for a flag. We compare our approach to a “passive” selection strategy, which uses the same classifier but picks labels to receive at random, as well as a single-level active baseline (traditional active learning) that uses our VOI function, but only selects from unlabeled regions. All methods are given a fixed cost and allowed to make a sequence of label requests until the cost is used up.

Figure 3.17 shows the resulting learning curves for the MSRC v2. Accuracy is measured as the average value of the diagonal of the confusion matrix for region-level predictions on the test set since the task is multi-class classification. All results are averaged over five random trials. The proposed multi-level active selection yields the steepest learning curves. Random selection lags behind, wasting annotation effort on less informative examples. As before, single-level active is preferable to random selection, yet we get best results when our active learner can choose between multiple types of annotations, including segmentations or image flags. The total gains after 1800 secs are significant, given the complexity of the 21-way classification problem with a test set containing 1129 image regions. Note that the random selection curve is probably an over-estimate of its quality; since we limit the unlabeled pool

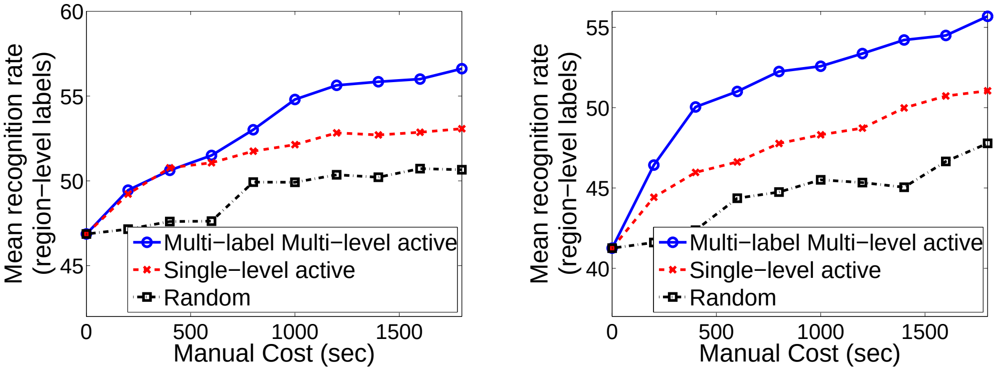


Figure 3.17: Learning curves when actively or randomly selecting multi-level and single-level annotations. **Left:** Region-level accuracy for the 21-class MSRC v2 dataset plotted against ground truth cost. **Right:** Region-level accuracy when 80 random images were added to the unlabeled pool. Our multi-level active selection approach yields the steepest learning curves while random selection lags behind, wasting annotation effort on less informative examples. When 80 random images are added to the unlabeled pool, random selection lags even further, since there are more uninformative images that it can choose.

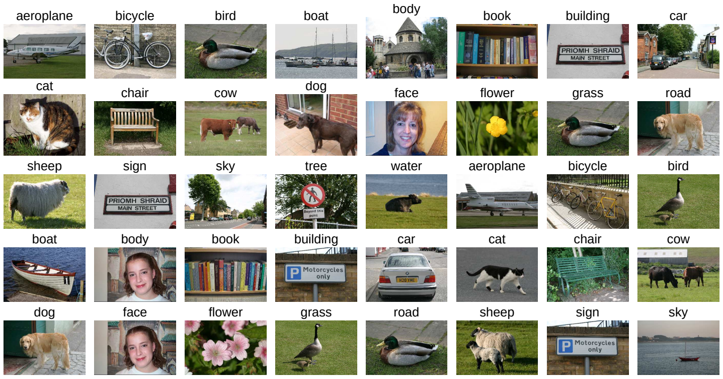
to only images from the MSRC, any example it requests is going to be fairly informative. Figure 3.17 (right) shows results for the same setting when 80 random images are added to the unlabeled pool with the “negative” class label, indicating that the more uninformative images that are present, the more random selection will lag behind.[7]

When active and random selection are run to completion on all labels, both methods reach an accuracy of 59.5% [8]; random selection requires 5776 units of manual effort to reach the upper-bound while active selection requires only 3075 units. Thus with active selection we reach the upper bound using 46.7% less cost than the passive learner requires.

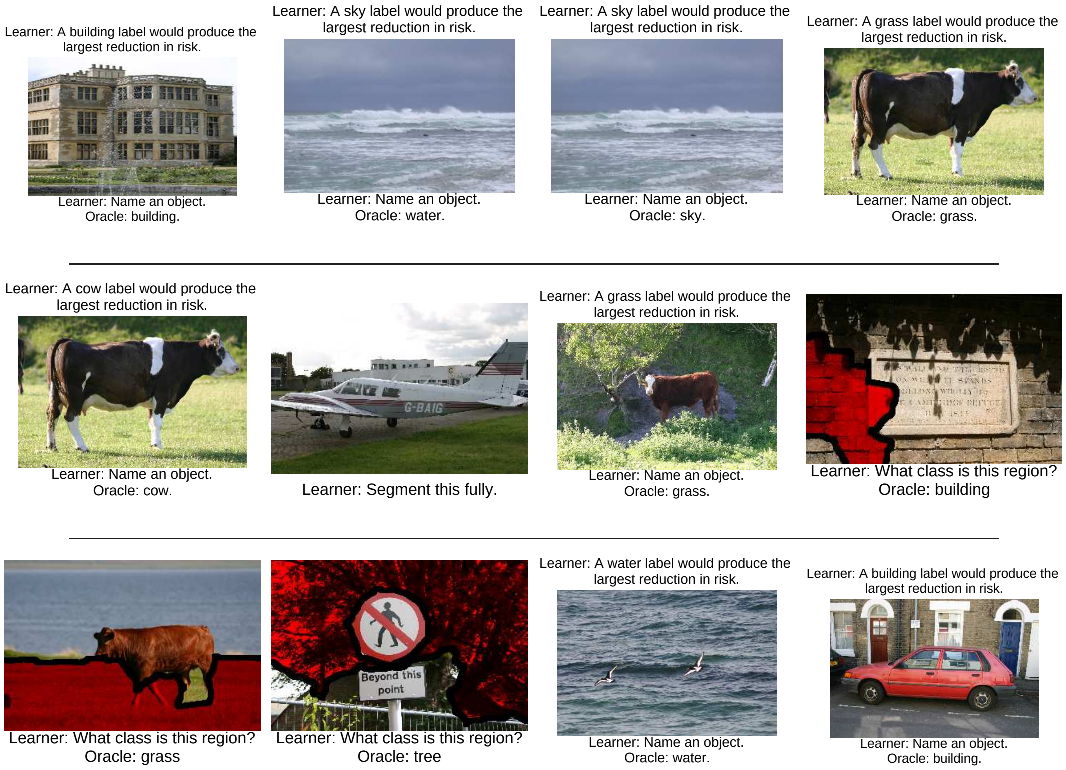
---

[7] In Chapter 6 I explore this notion further with live learning experiments in which training images, obtained automatically by querying web-based photo collections, naturally contain a mixture of useful and uninformative examples.

[8] Note that since we use a different train-test split for experiments in this section, this upper-bound is not comparable to the accuracy reported in Section 3.2.2



(a) Initial training set.


Learner: A building label would produce the largest reduction in risk.
Learner: Name an object.
Oracle: building.
Learner: A sky label would produce the largest reduction in risk.
Learner: Name an object.
Oracle: water.
Learner: A sky label would produce the largest reduction in risk.
Learner: Name an object.
Oracle: sky.
Learner: A grass label would produce the largest reduction in risk.
Learner: Name an object.
Oracle: grass.
Learner: A cow label would produce the largest reduction in risk.
Learner: Name an object.
Oracle: cow.
Learner: Segment this fully.
Learner: A grass label would produce the largest reduction in risk.
Learner: Name an object.
Oracle: grass.
Learner: What class is this region?
Oracle: building
Learner: What class is this region?
Oracle: grass
Learner: What class is this region?
Oracle: tree
Learner: A water label would produce the largest reduction in risk.
Learner: Name an object.
Oracle: water.
Learner: A building label would produce the largest reduction in risk.
Learner: Name an object.
Oracle: building.

(b) Annotations selected by the active learner in order (row major).

Figure 3.18: (a) Initial training set containing two examples per class. (b) Annotation queries selected by our method in subsequent iterations. Each image (from left to right) represents the example with the largest VOI as selected by our active learner on a sequence of iterations. The active learning query (one among a region label, an image tag, or a complete segmentation) is displayed at the bottom of the image along with the oracle's answer. For a query on a region, the corresponding region is highlighted in the image; for an image tag, the text on the top of the image represents what label is expected to produce the best reduction in risk.

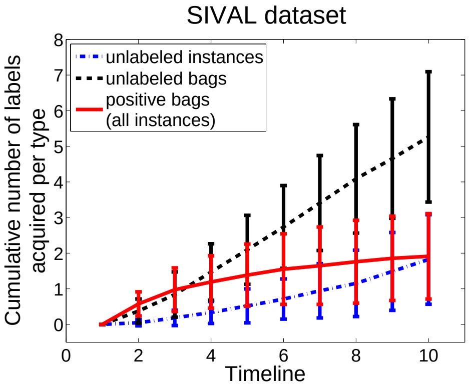


Figure 3.19: The cumulative number of labels acquired for each type with increasing number of queries. Our method tends to request complete segmentations or image labels early on, followed by queries on unlabeled segments later on. This agrees with the intuition that fewer segmentations are worth their higher annotation costs as the classifier becomes stronger.

3.2.3.3 Active Selection Examples

In this section we look at the types of annotation queries that our approach requests based on some qualitative and quantitative results. Figure 3.18 shows annotation queries selected by our approach during the first 12 iterations of an example run starting from a small training set consisting of two image tags per class. The initial training set is displayed in Figure 3.18(a), and Figure 3.18(b) shows the first 12 queries selected by our approach in row major order. The type of query and the result from the oracle are displayed at the bottom of the image. We also highlight the region being queried in the case of a region label; text on the top of the image shows which image tag our approach thinks would produce the biggest reduction in the risk (the l with the largest value in the summation in Equation 3.10).

The annotations requested by our approach are dominated by image tags, which is reasonable considering they are the least expensive labels among the three types. At the same time, the images for which tags are requested appear to consist of a small number of clearly defined objects ('sky', 'water' in the second and third images, 'water', 'building' in the first image, etc.). On more complex images, such as the sixth image of the airplane, a complete segmentation is requested. Also a region label on the ‘tree’ region is requested on the tenth image, even though a tree image tag is already available on the same image in the training set. This illustrates that in some cases stronger annotations might be required, even when the classifier already contains weaker information about a class.

The examples selected by our approach are also diverse in their appearance and class labels. For example, in the images selected by our approach that contain the region ‘sky’, the appearance of the region is distinct from the examples of ‘sky’ already available in the training set. This is also the case for classes ‘building’ and ‘water’.

Figure 3.19 shows the cumulative number of labels acquired for each type of annotation with increasing number of queries on the SIVAL dataset for the case of binary classification. Our previous observation on the larger proportion of image tags holds true in this dataset too. In addition, on this dataset our approach appears to select complete segmentations early on, followed by queries on unlabeled segments later on. Intuitively, as the classifier becomes stronger it may be that fewer segmentations can provide adequate risk reductions to mitigate their higher costs, and hence the less expensive image tags become favorable.

3.2.3.4 Effect of Initial Training Set Size

A well-known concern when performing active selection is that a faulty initial model might select uninformative examples to label and thus never converge to the most general hypothesis. Thus, we next consider the robustness of our approach by varying the number of training labels used to train the initial classifier. For the MSRC dataset we train the initial classifier with two, four, and eight image tags per class (42, 84, and 125 image tags overall) and

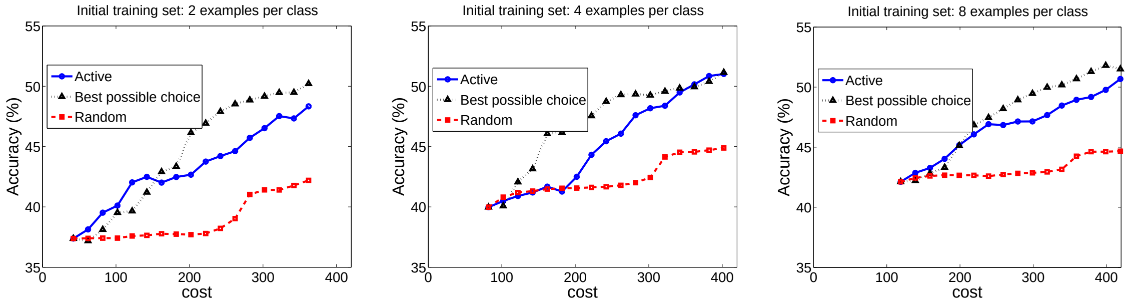


Figure 3.20: Effect of the initial training set size on the active selection on the MSRC dataset. The classifier is initialized with two (left), four (middle), and, eight (right) image tags per class, and active selection is compared with a random baseline and the best possible selection criterion based on the actual VOI. On the MSRC dataset our active selection criterion is robust to the initialization and performs much better than random selection on all three initial training sets. Nonetheless, we can expect the quality of the initial model to influence the reliability of the VOI in general.

then perform active selection with each model. In Figure 3.20, we compare our multi-level active selection approach against a multi-level random baseline and the best possible selection criterion. The best possible selection is obtained by computing the actual VOI of an example using its ground truth label. This is to compare how closely our expected VOI can approximate the actual VOI. We average results over five random trials.

On all three initializations, particularly for the smaller sets, our active selection approach has a larger slope than random selection. In addition, our active selection follows the trend of the best possible selection criterion. This illustrates the robustness of the approach to the initialization on this particular dataset. Also, since our multi-class classifier is an ensemble of a large number of binary classifiers, even with two image tags per class the final classifier could have enough examples to discriminate between the classes.

We show results for the same experiment for binary classification on the SIVAL dataset in

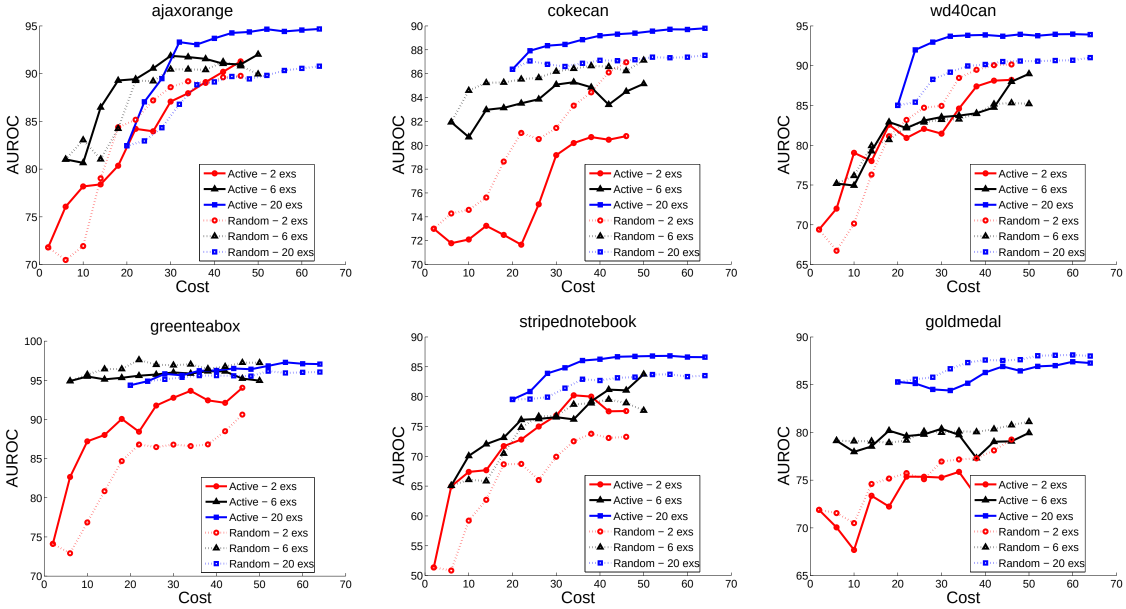


Figure 3.21: Effect of the initial training set size on active selection on the SIVAL dataset. We initialize the classifier with two, six, and twenty image tags equally distributed across positive and negative classes. The figure shows some representative (best and worst) learning curves for our active selection approach and a random baseline. On this dataset a small training set composed of only two examples produces sub-optimal selections for some classes.

Figure 3.21. The figure shows some representative (best and worst) learning curves comparing our selection function and a random baseline starting with two, six and twenty examples equally distributed across the positive and negative classes. The results are averaged over six random trials. Note that the three curves start at different points on the cost axes because they start with a different number of training examples. However, accuracies at a particular cost on the different curves are not necessarily comparable since the random initialization selects an equal number of positive and negative examples, while active and random selection approaches select from an unbalanced pool of positive and negative examples due to the one-vs-all binary setting.

The more variable results, as seen in the figure, could point to a harder dataset or the extremely low number of examples used in the binary setting as compared to the multi-class setting. The first row of learning curves show examples where a good initialization (larger number of examples) helps the active selection criterion. On these examples it appears that with smaller number of examples the active selection criterion could be misled into regions of the hypothesis space that do not necessarily correspond to the most general solution for the given training set. The first two curves in the second row are examples where even with very few training examples the active selection criterion is able to match results with a larger initial set. The final curve in the second row shows an example where active selection performs worse than random on all three initializations.

These results suggest that active learning could be affected by the initialization on certain problems. However, note that we deliberately chose an extremely small initial training set (two, six examples) to illustrate this point. Arguably, for most real applications one can

User	**Number of images**	**Accuracy (%)**
User 1	160	68.75
User 2	188	72.34
User 3	179	70.95
User 4	151	72.85
User 5	167	59.88
User 6	164	63.41
User 7	169	67.46
User 8	179	79.33
All users	210	**73.81**

Figure 3.22: Accuracy of our cost function in predicting “easy” vs. “hard”, both for user-specific and user-independent classifiers.

reasonably expect to initialize the model with at least 10’s of labeled examples.

3.2.4 Annotation Costs and Active Selection

In the following sections we evaluate how well we can learn to predict the difficulty of segmenting images using image features and the impact of using the predicted cost when making an active selection.

3.2.4.1 Annotation Cost Prediction

First, we isolate how well we can learn to predict the difficulty of segmenting images based on image features. To train our cost function, we gather data with Amazon’s Mechanical Turk. Users are required to completely segment images from the 14-class MSRC v1 dataset while a script records the time taken per image. We collect 25-50 annotations per image from different users. Users could skip images they preferred not to segment; each user was

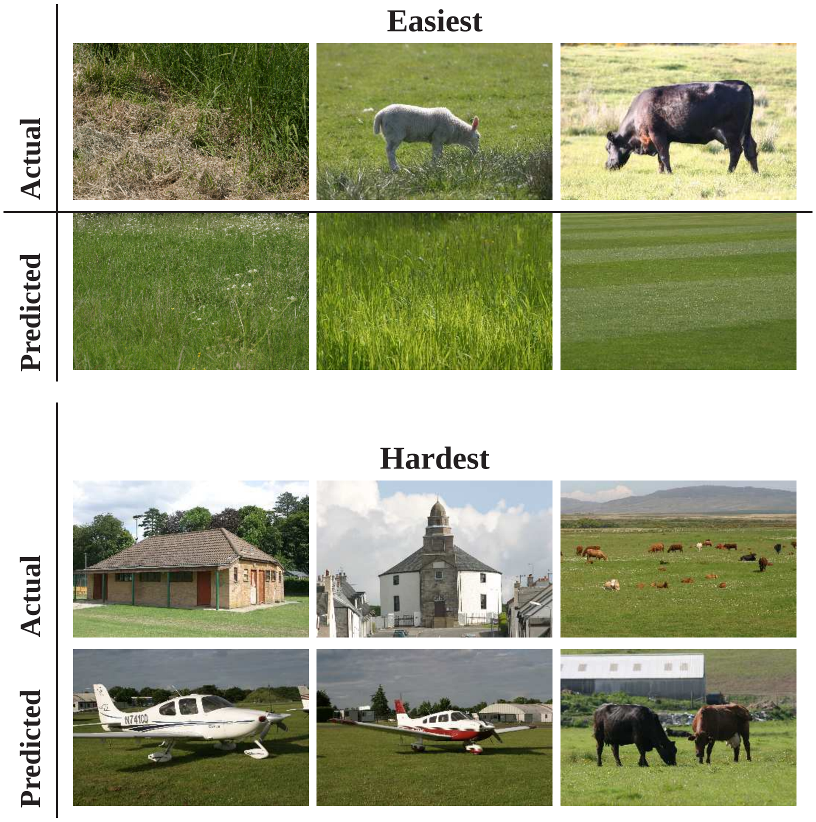


Figure 3.23: The easiest and hardest images to annotate based on actual users' timing data (top), and the predictions of our cost function on novel images (bottom).

allowed to label up to 240 images. However, no user completed all 240 images. The fact that most users skipped certain images (Figure 3.22, column: Number of images) supports our hypothesis that segmentation difficulty can be gauged by glancing at the image content.

We train both classifiers that can predict "easy" vs. "hard", and regressors that can predict the actual time in seconds. To divide the training set into easy and hard examples, we simply use a threshold at the mean time taken on all images. Using the feature pool described in Section 3.1.2.4, we perform multiple-kernel learning [3] to select feature types for both the user-specific data and the combined datasets. The edge density measure and color histograms received the largest weights (0.61, 0.33 respectively), with the rest near zero.

Figure 3.22 shows the leave-one-out cross validation (loo-cv) result when classifying images as easy or hard, for the users for whom we had the most data. For the majority, accuracy is well above chance. Most of the errors may largely be due to our arbitrary division between what is easy or hard based on the mean.

To train a regressor we use the raw timing data and the same set of features. Figure 3.23 shows examples that were easiest and hardest to segment, as measured by the ground truth actual time taken for at least eight users. Alongside, we show the examples that our regressor predicts to be easiest and hardest (from a separate partition of the data). These examples are intuitive, as one can imagine needing a lot more clicks to draw polygons on the many objects in the "hardest" set. Figure 3.24 (left) plots the actual time taken by users on an image against the value predicted by our cost function, as obtained with loo-cv for all 240 images in the MSRC v1 dataset. The root mean square difference between the actual and predicted times is 11.1 *s*, with an average prediction error of 22%. In comparison, predicting a constant value of 50 *s* (the mean of the data) yields an average prediction error of 46%. Given that the actual times vary from 8 to 100 *s*, and that the average cross-annotator disagreement was 18 *s*, an average error of 11 *s* seems quite good.

In order to verify that we were not simply learning a category-based level of effort, we looked at the actual and predicted times split across different classes. Figure 3.24 (right) shows a plot of the actual and predicted times broken across the different scene settings in the MSRC dataset. The x-axis shows the most dominant foreground class label in that particular scene

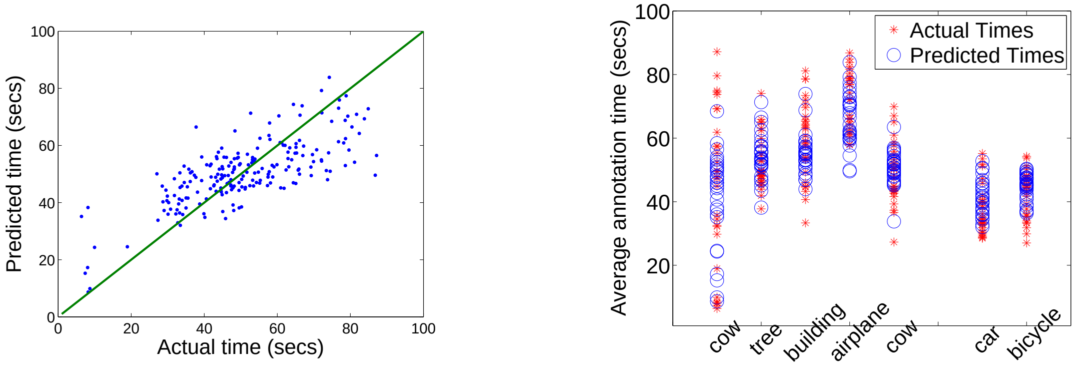


Figure 3.24: **Left:** Scatter-plot of the actual time taken by users to segment an image vs. the value predicted by our cost function, for the 240 images in the MSRC v1. The predicted and actual times are highly correlated, implying that our cost predictor has learned how difficult an image is to segment using only low-level image features. **Right:** The actual and predicted times split across the different categories of images in the MSRC dataset. The plot shows that most classes have images with varying difficulties, and assures that the difficulty measure we have learned is not class-specific.

layout. This figure shows that every class/scene layout contains images with varying difficulty in terms of the annotation effort required by users. While some categories have more variation than others (cow vs car), there is no direct connection between the image class and the time taken to provide annotations. The plot also shows that for most of the examples our cost predictor provides fairly accurate predictions of the annotation costs.

3.2.4.2 Active Selection with a Learned Cost Function

Thus far we have fixed the costs assigned per annotation type; now we show the impact of using the predicted cost while making active choices. We train a binary multi-instance classifier for each MSRC category using image labels on $\frac{4}{5}$-th of the data per class, in five different runs. The rest is used for testing. We compare two MIL active learners: one using

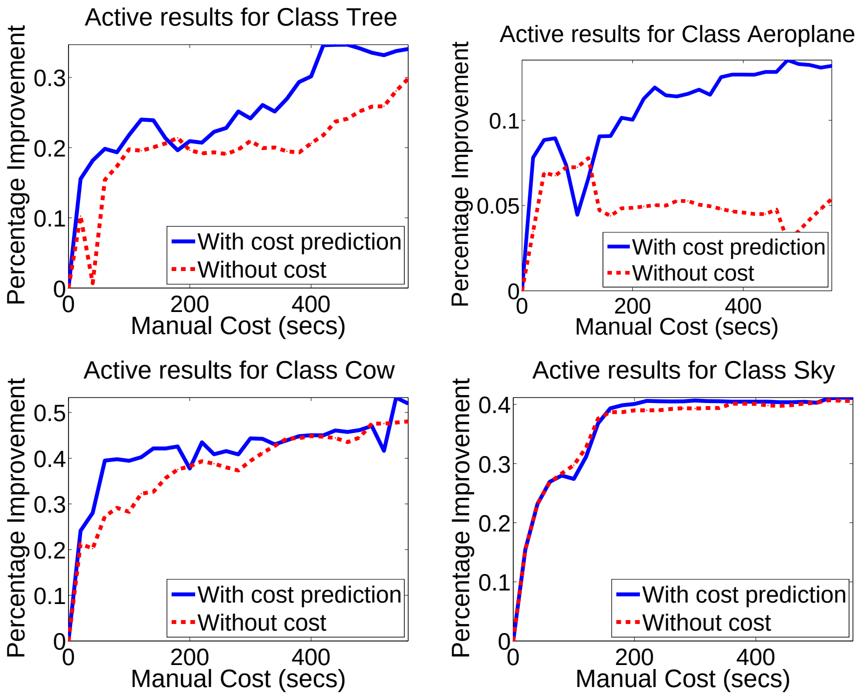


Figure 3.25: Representative learning curves when using active selection with the learned cost predictor, as compared to a baseline that makes active selections using a flat cost value. For classes like Tree, Cow, and Airplane (shown here), the cost prediction produces more improvement per unit cost, while for a few like Sky there is no significant difference—most likely because the images within the class are fairly consistent and equally informative and easy to label.

cost prediction, and one assigning a flat cost to annotations. At test time, both learners are "charged" the ground truth cost of getting the requested annotation.

Figure 3.25 shows representative (good and bad) learning curves, with accuracy measured by the AUROC value. For Tree, Cow, and Airplane, using the predicted cost leads to better accuracies at a lower cost, whereas for Sky there is little difference. This may be because most 'sky' regions look similar and take similar amounts of time to annotate.

% acc imp.	**Cost(secs)**		**% Cost saved**
	Cost prediction	Flat cost	
5	11.40	11.52	+1.07
10	24.52	31.41	+21.94
15	45.25	63.24	+28.45
20	165.85	251.10	+33.95
25	365.73	543.69	+32.73

Table 3.3: Savings in cost when using cost prediction within the active learner. Overall, our active selection takes less effort to attain the same level of accuracy as a cost-blind active learner.

Table 3.3 shows the cost required to improve the base classifier to different levels of accuracy. The fourth column shows the relative time savings our cost prediction enables over a cost-blind active learner that uses the same selection strategy. For larger improvements, predicting the cost leads to noticeably greater savings in manual effort—over 30% savings to attain a 25% accuracy improvement.

3.2.5 Computation Time

With our implementation of the incremental SVM technique of [17] it takes on average 0.5 secs to evaluate a single region and 20 secs to evaluate a bag (image) on a 1.6 GHz PC. This corresponds to about 15 minutes to choose which annotation to request when the dataset contains ~ 100 bags (images) for ~ 20 classes. Once an annotation is selected it takes less than 0.1 secs to retrain the classifier. The most expensive step in selecting an annotation is the Gibbs sampling procedure coupled with the need to update a large number of classifiers in the one-vs-one setting.

Since the complexity of the Gibbs sampling procedure depends on the number of segmented regions within an image one way to reduce computational costs would be to avoid oversegmenting an object into multiple regions. In [109], we proposed a novel top-down segmentation approach by defining pairwise potential functions for agglomerative grouping that measure how the classification entropy of the object-level classifiers changes when considering the combined appearance description of adjacent regions. Starting from an initial oversegmentation, we then iteratively merge regions which are better classified together. Such a technique could significantly improve the running time of our active selection scheme since it would avoid splitting adjacent regions if they are better classified when they are merged.

3.3 Discussion

This chapter addressed a new problem: how to actively choose not only which instance to label, but also what type of image annotation to acquire in a cost-effective way. Through extensive experiments I have validated several aspects of my initial thesis.

I showed that compared to traditional active learning, which restricts supervision to yes/no questions, a richer means of providing supervision and a method to effectively select supervision based on both information gain and cost to the supervisor is better-suited for building classifiers with minimal human intervention. Specifically, on the MSRC dataset we can save up to 50% of manual effort by choosing from multiple levels of annotations. Interestingly, our approach appears to select stronger supervision near the beginning of the active learning phase when the classifier is weak and switches to weaker supervision as it improves with more labels.

I showed that annotation effort, computed in terms of time to provide annotations, varies widely across images and annotators, from as little as 10 seconds to more than 2 minutes. By utilizing such user responses collected through an online crowd-sourced labeling service, I then showed that one could learn a function to predict image-specific cost based on image features alone to a fair degree (22% prediction error). A large portion of the error could be attributed to disagreement across annotators due to differing skill and attention levels. This suggests that extending the approach to target specific annotators and build user-specific cost functions could provide more accurate predictions.

I also showed that our approach is fairly robust to the initial classifier as long as the model is initialized with 10's of examples. With fewer than 6 labels the results obtained for active learning are more variable. It appears that with smaller number of examples to start with, the selection function could be misled into regions of the hypothesis space that do not necessarily correspond to the most general solution for the given training set. Although, note that we deliberately chose an extremely small initial training set (two, six examples) to illustrate this point. Nonetheless, this is a common concern of all active selection approaches and results depend on the exact data distribution and the difficulty of the task being learned.

Our method is general enough to accept other types of annotations or classifiers, as long as the cost and risk functions can be appropriately defined. Specifically, this would require defining classifiers that can provide posterior probabilities for different types of annotations being considered and a way to measure the cost of each type of annotation. For example, for a part-based object detector/classifier one could use annotations at the part-level or treat object parts as latent variables when given object-level annotations and apply my technique to learn from a combination of the two.

While we have concentrated mostly in the domain of object recognition, the problem of comparing different types of annotations in a unified framework is potentially applicable to several other domains both in vision and machine learning such as video annotation, tracking, or document classification. For example, in document classification a web document would be a bag of paragraphs and one could obtain annotations on documents as a whole or on individual paragraphs while learning a particular concept. Similarly, videos could be considered as bags of frames where annotations can be obtained either on a video as a whole or on its individual frames.

So far, I considered selecting a single annotation from multiple levels in order to reduce total human effort. In the next chapter I will address the problem of selecting examples for multiple simultaneous annotators so as to improve the speed of the annotation process by posting annotation questions in parallel.

Chapter 4

Selection under a Budget for Multiple Simultaneous Annotators

The previous section dealt with choosing a *single* annotation from a large pool of multiple types of annotation queries. Such methods can be used to train classifiers when a single human annotator is available to interact with the system. However, in some real applications we may have access to multiple simultaneous annotations. For example, systems such as Mechanical Turk or LabelMe provide access to a large number of annotators on the web. An active learning system that needs to repeatedly go offline and compute the next annotation request cannot take advantage of such resources. Therefore, it may in some cases be preferable to farm out a *batch* of good queries at once.

In this chapter, I formalize the problem of *far-sighted active learning under a budget*. At each iteration the active learner is allowed to choose a set of examples to get labeled, provided the total sum of costs associated with the selected examples is under a given budget. The technical problem of selecting a good set of examples at once is challenging, since one must take care to avoid overlapping information, i.e., it is wasteful to ask a batch of similar questions. Furthermore, it is risky to formulate a large selection based only on the current model's view of the data: some examples within large sets may lead to significant changes to the classifier that ultimately invalidate the perceived value of others that were selected.

The result of [57] is of particular interest for any non-myopic or batch-mode active learning algorithm that tries to make a large selection based on the current classifier. Central to their analysis is a theoretical bound which quantifies the performance difference between sequential active learning (myopic) and a priori design strategies or batch-mode selection methods. Using Gaussian processes (GP) and a maximum entropy based selection scheme, they show that if there is low uncertainty over the classifier parameters the predictive distribution should be independent of additional observed values and there should be *almost* no benefit from sequentially (one at a time) obtaining label information on unlabeled examples. However, the result requires that the probability distribution of the classifier parameters is highly peaked, which is not always true in active selection where the initial classifier is trained on a small number of examples and is therefore quite uncertain.

While a few “batch-mode” active learning strategies have been proposed in the machine learning literature [85, 12, 45, 42], none consider how to balance the joint selection with cost requirements. Meanwhile, current active selection approaches that do account for labeling cost lead to a myopic selection of a single request at a time [55, 4, 41, 107, 108].

I propose a novel method for optimally selecting a set of examples for a support vector machine (SVM) classifier under these conditions. Given a large unlabeled pool of data where each example has an associated cost, we introduce a set of instance selection variables. We formulate an optimization problem to learn the maximum margin hyperplane along with the instance variables that minimize the empirical risk (on both the labeled data and selected unlabeled points), while satisfying the given budget constraint. We then relax it to a continuous optimization problem that can be decomposed into two strictly convex optimization problems loosely coupled in the hyperplane parameters and selection variables. We devise a monotonically convergent alternating minimization algorithm to compute the solution.

In the following, I will formally define the problem setting for our approach and explain our alternating optimization solution and provide an algorithm for the same. I then provide validation for our approach on benchmark datasets for three recognition applications: object recognition, activity recognition, and content-based image retrieval. I demonstrate the advantages of our approach compared to passive, myopic greedy, and batch selection baselines, and show its effectiveness across a range of budgets.[1]

4.1 Budgeted Batch Active Learning (BBAL)

Given a preliminary recognition model and a budget for annotations to improve the training set, our method considers all the available unlabeled image data and computes the set of recommended requests that are jointly most informative and fall within the budget. Below, we first formally define the problem of budgeted selection, and overview the main idea of our approach. In Section 4.1.2 we present the detailed formulation and algorithm.

4.1.1 Problem Definition and Overview

We consider the problem of actively selecting a batch of examples to label, where the contents of the batch must be constrained by some budget. Formally, let $L = \{(x_1, y_1), (x_2, y_2), ...(x_l, y_l)\}$ denote a set of *l* initially labeled examples, where $y_i \in \{+1, -1\}$. Let $U = \{x_{l+1}, x_{l+2}, ...x_{l+u}\}$

---

[1] The contents of this chapter were published in [112].

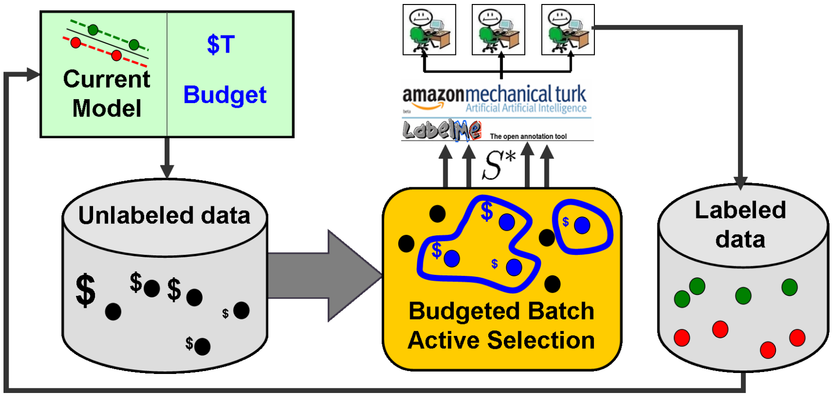


Figure 4.1: The problem setting of budgeted batch active learning. At each iteration the active learner can select a set of examples to label whose total cost meets a given budget of supervision for the iteration. The selected examples can then be labeled by multiple annotators working in parallel (for example, by using services such as Mechanical Turk).

denote an unlabeled pool from which examples can be selected and given to human labelers. Each unlabeled example x_i is associated with a cost c_i, which measures the manual effort required to obtain a label for x_i. Note that the cost varies per example, as in Chapter 3.

At each iteration, a set of examples $S = \{x_{k_1}, x_{k_2}, ...x_{k_n}\} \subseteq U$ can be selected for labeling, as long as the total cost of the selection does not exceed a specified budget T. That is, $\sum_{j=1}^{n} c_{k_j} \leq T$. Since costs vary, the number of selected examples n is not fixed. The goal is therefore to maximally utilize the given budget T by selecting the set S that is expected to produce the most gain in the classifier's performance. After obtaining labels for the chosen set, the classifier will be retrained, and the process can repeat, one batch at a time. Figure 4.1 provides an overview of the problem setting.

A naive approach to this problem, which we refer to as *Myopic Active Batch Learning*, would be to greedily choose the top most uncertain examples according to the current classifier that fit under the given budget—in other words, to rank the points in descending order by their uncertainty, and start adding them to the set S until the total budget is exhausted. However, such an approach ignores the information overlap between the selected examples.

Existing methods counter this problem by choosing a set that contains both examples that are uncertain and that are mutually diverse [12, 45]. Aside from needing good heuristics to balance the two properties, estimating uncertainty based on the *current classifier* (e.g., using the distance from the margin for an SVM) also fails to capture how uncertainty will change once the selected examples are added to the labeled set and the model's parameters are retrained. For large batches of examples this can be especially problematic. In addition, existing methods are specifically targeted at choosing a fixed number of examples at each iteration, but a variable-sized batch may be able to more optimally use labeling resources (i.e., a fixed-size batch must take n total examples, whereas a more effective selection might entail choosing a couple of the more expensive examples together with a set of $<< n$ cheaper ones).

Therefore, we propose an approach that directly targets the amount of reduction in the SVM objective that is to be expected by choosing a given set of examples under a budget. We call this *budgeted batch selection*. The main idea is as follows: we introduce an indicator variable over the unlabeled examples, and formulate a continuous optimization problem to determine which subset of possible queries should maximize the improvement to the classifier's objective, without overspending the budget. When fixing the selection variables, the optimization reduces to that of a standard SVM objective function, which can be solved efficiently; when fixing the model parameters, the selection variables are computed via linear programming.

Because we incorporate the predicted change in the model that the candidate examples will induce, the method is “far-sighted” in terms of the effects of the entire batch.

4.1.2 Formulation and Algorithm

Given a set of labeled examples L, the SVM objective seeks the optimal separating hyperplane defined by parameters (w, b):

$$\begin{aligned} \operatorname*{argmin}_{w,b,\epsilon} \;& \frac{1}{2}||w||^2 + C \sum_{(x_i,y_i)\in L} \epsilon_i, \\ \text{s.t. } & y_i(w^T x_i + b) \geq 1 - \epsilon_i,\ (x_i, y_i) \in L, \\ & \epsilon_i \geq 0, \end{aligned} \tag{4.1}$$

where each ϵ_i denotes the hinge loss on x_i, and C denotes the constant regularization penalty. This familiar SVM objective simultaneously minimizes the classification error on the training examples while maximizing the margin of separation between the positives and negatives.

Let A and B be two (possibly distinct) sets of labeled examples. To aid in notation below, we define an intermediate cost function, which takes parameters f_A and B:

$$g(f_A, B) = \frac{1}{2}||w_A||^2 + C\hat{R}_B^A, \tag{4.2}$$

where f_A denotes the SVM hyperplane parameters $f_A = (w_A, b_A)$ obtained by training on set A, and $\hat{R}_B^A = \sum_{(x_i,y_i)\in B} \epsilon_i^A$ denotes the empirical loss incurred by model f_A over the set B, and

$$\epsilon_i^A = \max(0, 1 - y_i(w_A^T x_i + b_A)) \tag{4.3}$$

denotes the hinge loss on x_i resulting from the model f_A. Note that the cost measured by $g(f_A, B)$ evaluates a margin term $\frac{1}{2}||w_A||^2$ (which reflects generalization ability) using the solution according to labeled data A, whereas it evaluates losses (which reflect misclassifications) on examples in B using the model f_A.

We want both the labels on the candidate selection sets as well as the existing labeled data to simultaneously influence the batch selection. As the points in a candidate set S are as yet unlabeled, we can only estimate the most "optimistic" cost reduction by maximizing over all possible labels on S. In the following, we use the term *optimistic labels* (borrowed from [42]) to refer to a label assignment for unlabeled points under which cost is maximally reduced.

Let $Y^* = \{y_{k_1}, \ldots, y_{k_n}\}$, be the set of optimistic labels associated with the examples in the optimal selection $S^* \subseteq U$, where $Y^* \in \{+1, -1\}^n$, for $n = |S^*|$. We want to select (S^*, Y^*) such that together they yield the maximal cost reduction, as measured by the cost produced *before* their addition to the labeled set versus the cost produced *after* they are added. Specifically, we want:

$$\begin{aligned}(S^*, Y^*) = \arg \min_{S \subseteq U, Y} \; & g\left(f_{L'}, L'\right) - g\left(f_L,\ L \cup (S, Y_L)\right), \\ \text{s.t.} \; & \sum_{x_i \in S} c_i \leq T, \end{aligned} \tag{4.4}$$

where $L' = L \cup (S, Y)$—that is, the labeled set expanded with some label assignment on S—and Y_L denotes the labels obtained by classifying S using f_L. The last inequality reflects the budget constraint limiting total annotation cost among selected examples to T. Note that the first term in the above objective measures the classification error on $L \cup (S, Y)$ and the margin when training using both L and (S, Y), while the second term measures both the margin and the classification error for the selected examples under the "old" model f_L, which is trained only on L. Thus, the optimal (S^*, Y^*) results in the maximal reduction in the SVM objective when considering optimistic labels.

To solve this optimization problem, we first expand the representation of the unlabeled set so that each unlabeled example appears as two examples labeled with both possible classes. Formally, we expand U to also include:

$$\begin{aligned} x_i &= x_{i-u}, \text{for } i \in [l+u+1, \ldots, l+2u], \\ y_i &= +1, \quad \text{for } i \in [l+1, \ldots, l+u], \\ y_i &= -1, \quad \text{for } i \in [l+u+1, \ldots, l+2u]. \end{aligned} \tag{4.5}$$

From here on, U represents the expanded unlabeled set. We then introduce a vector of indicator variables $q \in [0,1]^{2u}$, where $q_j = 1$ denotes that example $x_{l+j} \in S$, and $q_j = 0$ denotes that it is not. Let Y_U denote the set of labels on all unlabeled examples, which includes the labels Y for selection S. Now redefining $L' = L \cup (S, Y_U)$ we can rewrite the first g term from Equation 4.4 as:

$$\begin{aligned} g(f_{L'}, L') &= \frac{1}{2}||w_{L'}||^2 + C\hat{R}^{L'}_{L'} \\ &= \frac{1}{2}||w_{L'}||^2 + C\hat{R}^{L'}_{L} + C_u\hat{R}^{L'}_{(S,Y_U)}, \\ &= \frac{1}{2}||w_{L'}||^2 + C\hat{R}^{L'}_{L} + C_u\sum_{j=1}^{2u} q_j\epsilon^{L'}_{l+j}, \end{aligned} \tag{4.6}$$

where C_u is a constant regularization penalty for the selected unlabeled examples. Here $f_{L'}$ is obtained by solving the optimization problem in Equation 4.1 with the set $L \cup (S, Y)$, and the values for each ϵ_i are also based on this model $f_{L'}$ (and hence the labels Y_U), as denoted by the $\epsilon^{L'}_{l+j}$ terms.

Substituting $g(f_{L'}, L')$ from Equation 4.6 into Equation 4.4, the desired selection problem can now be written as:

$$\begin{aligned}
&\min_{w,b,q}\ \frac{1}{2}||w||^2 + C\sum_{x_i\in L}\epsilon_i + C_u\sum_{j=1}^{2u}\epsilon_{l+j}q_j - C_u\sum_{j=1}^{2u}\epsilon_{l+j}^{L}q_j,\\
&\text{s.t.}\ \ y_i(w^Tx_i+b)\geq 1-\epsilon_i, \epsilon_i\geq 0,\ 1\leq i\leq l+2u,\\
&\qquad \sum_{j=1}^{2u}q_jc_j\leq T,\\
&\qquad q_j+q_{u+j}\leq 1,\ \ 1\leq j\leq u,\\
&\qquad q_j\in\{0,1\},\ \ 1\leq j\leq 2u,
\end{aligned}\tag{4.7}$$

where $L' = L\cup(S, Y_U)$. Note that our encoding of the indicator means that q^* itself represents (S^*, Y^*) from Equation 4.4, and similarly the expanded labeled set L' is a function of q. We drop the superscripts L' for the ϵ_i variables for clarity since L' is now a parameter that we are optimizing over. Note that the first two terms of Equation 4.6 for $g(f_L, L\cup(S, Y))$ are constant w.r.t. the optimization variables and thus are ignored. The last term reflects the loss incurred for examples in S using a model L that does *not* account for labels Y_U, whereas the middle two reflect errors after its inclusion.

Although Equation 4.7 includes a constraint for every unlabeled example $x_{l+j}\in U$, since the penalty for the corresponding slack variable ϵ_{l+j} is zero whenever q_j is zero, the constraint only affects the cost for examples with non-zero q_j, that is, for $x_{l+j}\in S$. Finally, the constraint on pairs of q variables ($q_j + q_{u+j}$) reflects that only one of the labels ($+1$ or -1) can be chosen for an unlabeled example.

The optimization problem defined above is an integer programming problem which in general is NP-hard. Hence, we first relax it to a continuous optimization problem by allowing the q variables to take values between $(0,\ 1)$. Now the above objective can be seen as two different optimization problems loosely coupled by the term $C_u \sum_{j=1}^{2u} \epsilon_{l+j} q_j$: one on (w, b) and the other on q, both of which are convex. Fixing q, the minimization over w can be done by standard convex quadratic programming. Fixing w, the minimization over q is a convex linear programming problem.

To solve the relaxed problem, we devise an iterative alternating minimization procedure that is guaranteed to converge to a local optimum of the objective function. Assuming q to be constant, Equation 4.7 reduces to

$$\begin{aligned}(w^*, b^*) = \operatorname*{argmin}_{w,b} \;& \frac{1}{2}||w||^2 + C \sum_{(x_i,y_i)\in L} \epsilon_i + C_u \sum_{j=1}^{2u} \epsilon_{l+j} q_j, \\ \text{s.t.} \;& y_i(w^T x_i + b) \geq 1 - \epsilon_i, \epsilon_i \geq 0, \;\; (x_i, y_i) \in L, \\ & y_i(w^T x_{l+j} + b) \geq 1 - \epsilon_{l+j}, \\ & \epsilon_{l+j} \geq 0, \;\; x_{l+j} \in U, \;\; y_{l+j} \in Y_U. \end{aligned} \tag{4.8}$$

Note that this objective has a very similar form to that of the transductive SVM, as first proposed in [50]. Importantly, unlike the transductive SVM, in this case the inclusion of the indicator vector q means we penalize labeling errors on unlabeled data in S only, which is a subset of all unlabeled examples. Moreover, for a fixed q the problem reduces to that of the standard SVM problem, where the cost terms for the unlabeled examples are a function of the q variables. Hence, for a given q, we can efficiently optimize the SVM parameters and their optimistic labels for the selected batch.

On the other hand, fixing the model parameters (w, b) and relaxing the indicator vector as

$q \in (0,1)^{2u}$, Equation 4.7 reduces to

$$\begin{aligned} q^* = \underset{q}{\operatorname{argmin}} \;\; & C_u \sum_{j=1}^{2u} \epsilon_{l+j} q_j - C_u \sum_{j=1}^{2u} \epsilon_{l+j}^{L} q_j, \\ \text{s.t.} \;\; & \sum_{j=1}^{2u} q_j c_j \leq T, \\ & q_j + q_{u+j} \leq 1, \;\; 1 \leq j \leq u, \\ & 0 \leq q_j \leq 1, \;\; 1 \leq j \leq 2u. \end{aligned} \tag{4.9}$$

The above problem is a linear programming problem in q and can be solved using standard methods like an interior point method.[2] The ϵ_{l+j} variables depend on the current solution for (w^*, b^*) from Equation 4.8, whereas ϵ_{l+j}^{L} is a function of the parameters (w_L, b_L)—which are obtained by training on L alone—and Y_L, the true labels obtained so far.

Finally, by alternating between Equation 4.8 and 4.9, we can compute the batch selection meeting the given budget that is expected to most improve the classifier. We always initialize the ϵ values to 0, which corresponds to initializing our method with the myopic solution. We form S^* by choosing the examples with the largest q_i that fit the given budget T. Algorithm 1 provides pseudo-code for the procedure.

Note that the constraints on $\{w, b, \epsilon\}$ in Equation 4.7 are independent of q. Similarly, constraints on q are independent of $\{w, b, \epsilon\}$. Hence, fixing q and optimizing for $\{w, b, \epsilon\}$ decreases the objective function (Step 5 in Alg. 1). Similarly, Step 6 also decreases the objective function. Hence, our algorithm converges monotonically. In fact, with a stronger analysis, it

---

[2] In the implementation, we need to add slack on T since with varying costs per example one can only hit the budget T as closely as possible, but for clarity of presentation we omit it in the notation above.

Algorithm 1 Budgeted Batch Active Learning (BBAL)

```
Require: Labeled data - $L$, Unlabeled data - $U$,
         Current loss on unlabeled data - $\epsilon_i^L$,
         Labeling costs - $c = [c_1, \dots, c_u]$, Budget - $T$,
         Parameters - $C$, $C_u$, $\zeta$.
Initialize $\epsilon_{l+j}$ = $0$, for $j = 1, \dots, 2u$.
$Y_U = \{y_1, \dots, y_{2u}\}$, set as in Equation 4.5.
$\mathcal{C}(q_{old})$ = $\infty$, where $\mathcal{C}(\cdot)$ denotes objective in Equation 4.7
repeat
    $q$ = solve_linear_program ($\epsilon$, $\epsilon^L$, $c$, $T$)      // Equation 4.9
    $[w, b]$ = svm($L \cup (U, Y_U)$, $C \cup qC_u$)      // Equation 4.8
    Compute $\epsilon$ using $Y_U$, $w$, and $b$.
    $\mathcal{C}(q_{new})$ = $q$.
until convergence.
Set $q_j$ = max($q_j$, $q_{j+u}$), for $j = 1, \dots, u$.
return Set $S^* = \cup_{q_j>0}\ x_{l+j}$, for $j = 1, \dots, u$.
```

is easy to show that our algorithm converges to a local optimal of the objective function. In our experiments the algorithm converges quickly, requiring typically only 10-15 iterations.

4.1.3 Summary: Using BBAL

Our approach can be used for active training of any SVM classification problem. The inputs are an initial training set containing some labeled examples of the categories of interest, the number of selection iterations, an unlabeled pool of data, and the available budget. In practice, one would set this budget according to the resources available—for example, the money one is willing to spend on Mechanical Turk to get a training set for the next object recognition challenge. We construct the initial classifier, and then for each iteration, solve for the indicator vector specifying which set of unlabeled data objects should be annotated next. For unlabeled data with non-uniform costs, each resulting request will consist of a variable number of items (images, video clips). Once these tasks are completed (either sequentially, or in parallel by a team of annotators), the labeled set is expanded accordingly, the classifier is updated, and budgeted selection repeats. The final output is the trained classifier.

4.2 Results

I demonstrate my approach for active budgeted selection within multiple visual recognition applications. The main goal of our experiments is to demonstrate the advantage of maximally utilizing budgets of any size, and to validate the importance of using the change in the classifier objective when choosing large batches. To show these things, we consider three baselines:

- **Passive selection**: randomly chooses examples to label. To implement this on a budget, we randomly draw from the unlabeled pool until the budget is exhausted.
- **Myopic active batch** learner (MAB): greedily takes the top most informative examples whose summed costs come in under budget.
- **Batch-mode active** learner (BMAL): a state-of-the-art approach that selects batches of a fixed size [45]. Like our method, it considers an SVM objective, but unlike ours it does not include the model's expected change during selection, and it ignores per-example costs.

We emphasize that, to our knowledge, no existing method allows batch selection on a budget, making these the best three baselines to analyze.

Dataset	Cost (secs)				
	min	max	mean	median	total
SIVAL	4	202	31.9	32	6752
Hollywood	1.64	92.7	15.4	8.7	2476.7

Table 4.1: Distribution of manual effort costs on SIVAL and HOHA.

4.2.1 Datasets and Implementation Details

We use three publicly available benchmark datasets: SIVAL for object recognition, Hollywood for activity recognition, and Corel for CBIR. The first two consist of examples that require variable effort to annotate, allowing us to study the advantages of selecting requests to meet a budget. The third allows us to make direct comparisons with a state-of-the-art batch selection method for image retrieval.

- The SIVAL dataset contains 1500 images, each labeled with one of 25 object labels (e.g., gloves, apple, etc.). The cluttered images contain objects in a variety of poses and lighting conditions. We use the color and texture features provided by the dataset creators[3], which gives a 30-dimensional descriptor for each of 30 regions per image. For this dataset, the annotation cost per image is the time required for manual segmentation; we use the cost data provided by [88], though one could predict annotation costs using image features alone with sufficient training as we showed in Chapter 3. Note that this data was also used in the VOI experiments in Chapter 3.

- The Hollywood dataset (HOHA) contains 444 video samples with human actions from 32 movies [65]. Each sample is labeled according to one or more of 8 action classes

---

[3] http://www.cs.wustl.edu/accio/

(e.g., AnswerPhone, GetOutCar, HandShake, etc.). We use the "clean" training set. For features, we use the authors' code[4] to extract HoG-HoF descriptors around space-time Harris interest points. The space-time Harris detector [64] detects local structures in a video where the image values have significant local variations in both space and time. The HoG-HoF descriptors compute histograms of oriented gradients and optical flow inside a space-time volume surrounding the interest points. We then convert each action clip into a bag-of-words representation with 1000 words. For this dataset, we use the length of a video-clip to measure the annotation effort, since a human will watch the entire clip in order to identify which of the actions are performed in it. Table 4.1 shows the distribution of manual effort costs on the two above datasets.

- The Corel dataset contains 5,000 images from 50 different categories (e.g., antelope, butterfly, car, cat), as selected by the authors of [45]. Each category contains 100 images. We use the features provided on the authors' website[5], which consist of color moments, edge histograms, and a wavelet-based texture feature.

For SIVAL, we use an RBF kernel with the coefficient of 10^-5, which we set based on the feature space dimension. For HOHA, we use a χ^2 RBF kernel on HoG and HoF, with parameters as specified in [65]. We set the SVM penalty parameters as $C = 100$ and $C_u = 100$ for all approaches, a large value intended to emphasize correct classification of the selected examples. We train and test all approaches in the one-vs-all binary classification setting, and use the standard provided train-test splits. For SIVAL and Hollywood datasets, we use the

---

[4] http://www.irisa.fr/vista/Equipe/People/Laptev/download.html

[5] http://www.cais.ntu.edu.sg/$\sim$chhoi/SVMBMAL/

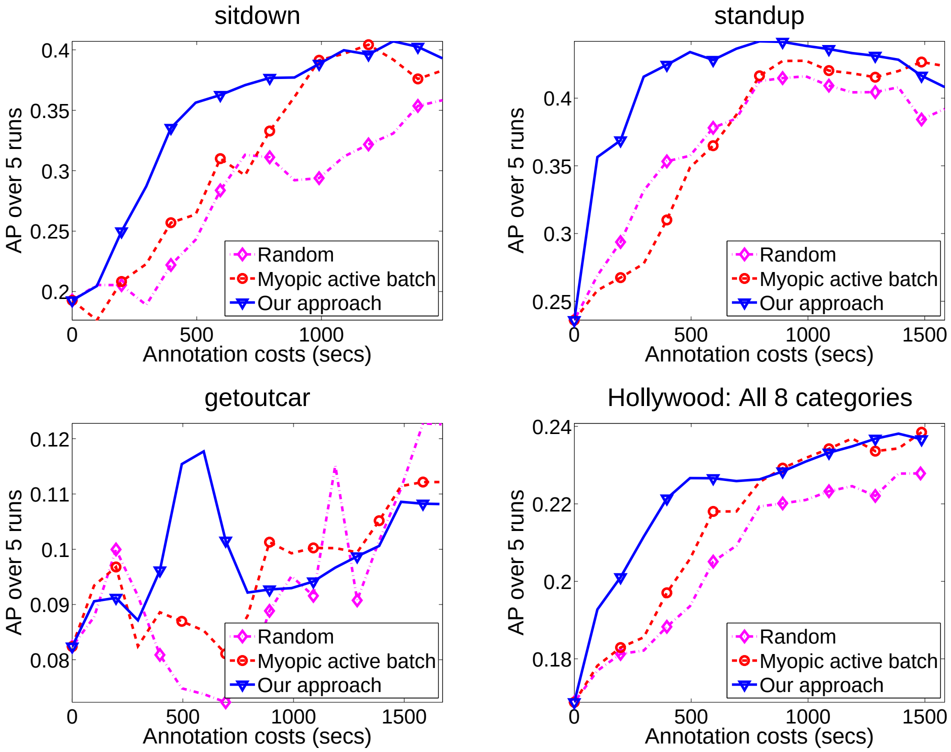


Figure 4.2: Results on the Hollywood dataset: example per-category learning curves (first two are best, third is worst) and the average results over all eight categories (bottom right plot) when actively learning categories of human activity from video clips.

area under the ROC curve (AUROC) as the evaluation metric as it is the most appropriate metric for binary classification. For COREL, we report results using precision and recall since the task is image retrieval and it will also allow us to compare our results with other state-of-the-art approaches such as [45].

For SIVAL and HOHA, our active learner's initial training set consists of five positive and five negative images per class, selected at random; we use the remainder as the unlabeled pool. We average all results over five such random selections.

4.2.2 Learning Activities on a Budget

Figure 4.2 shows representative (best and worst) learning curves for our method and the passive and myopic baselines plotted against the cost (annotation time) of the selected examples on the Hollywood dataset. The budget T is set such that all the unlabeled examples would be exhausted in about 20 batch iterations. About 10-15 examples on average get chosen per iteration. Note that average precision (AP) accuracy is plotted against the effort required to obtain annotations on the selected examples—not the number of queries—since the videos vary in length and require variable time to annotate.

All three methods steadily improve upon the initial classifier, but at different rates with respect to the cost. In general, a steeper learning curve indicates that a method is learning most effectively from the supplied labels. For most classes, our approach shows the most significant gains at a lower cost, meaning that it is best suited for maximally utilizing a budget. The myopic active batch baseline (MAB) is a bit better than random selection for most cases, but is weaker than our method due to its failure to account for the examples' cost and potential redundancy. Our results on some actions (e.g., "get out of car") are more variable than others, which we attribute to the fact that the training and test clips are from distinct movies, and therefore vary a lot in terms of lighting, appearance, characters, etc. Overall, however, our approach consistently produces better accuracy for lower annotation cost, and outperforms the baselines on average over all eight actions (bottom right plot in Figure 4.2).

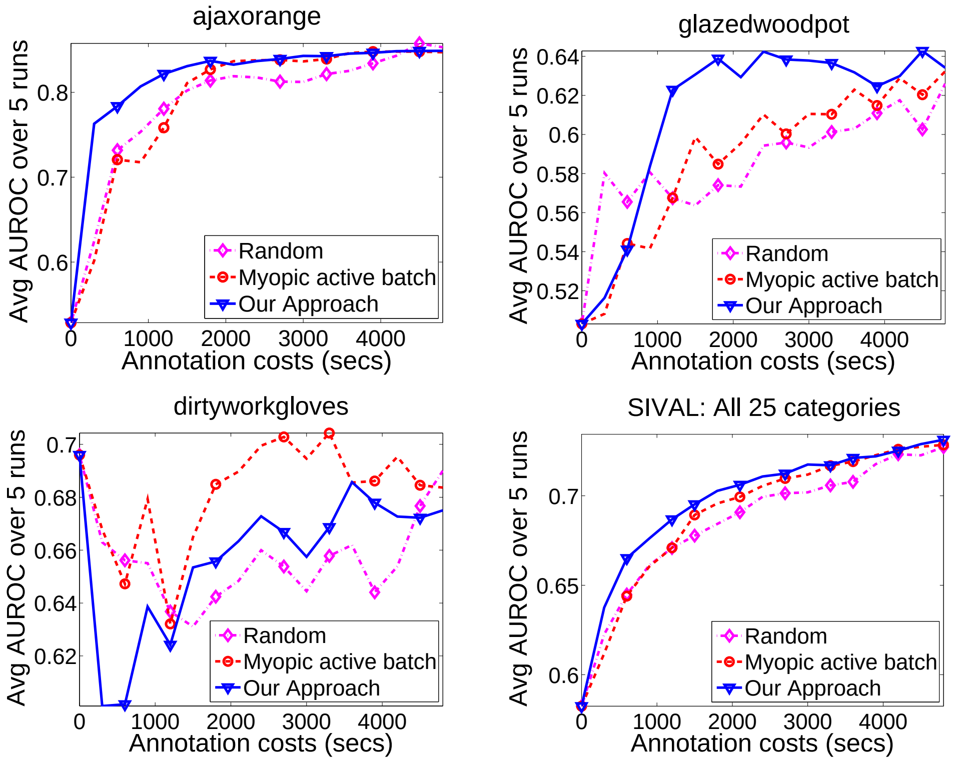
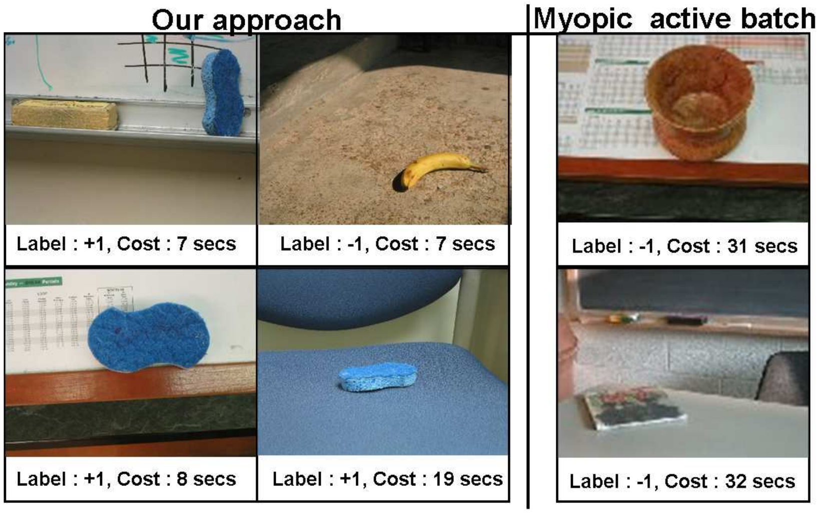


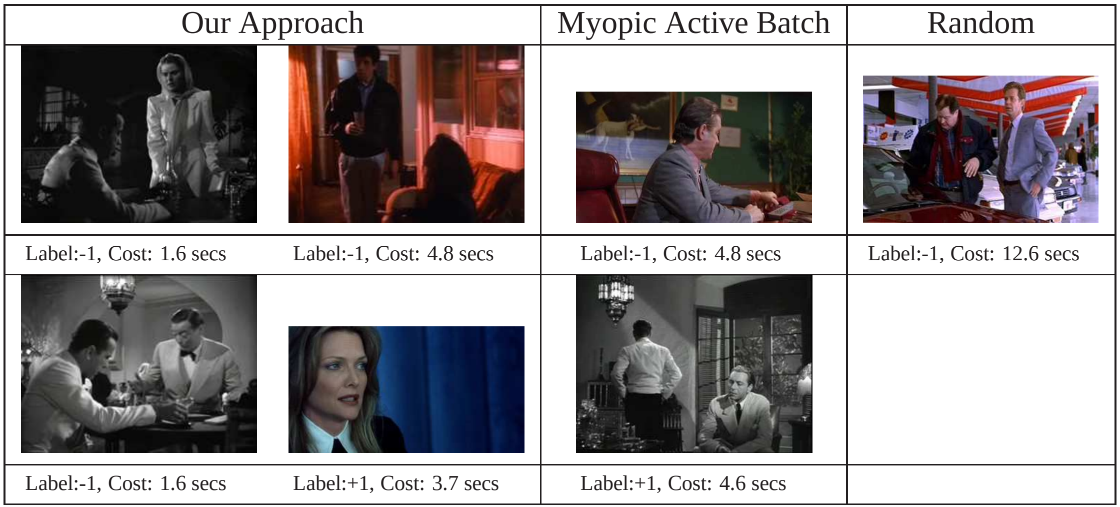
Figure 4.3: Results on the SIVAL dataset: example learning curves (first two are best, third is worst) and the average over all 25 categories (bottom right plot) when actively learning object categories from image examples.

4.2.3 Learning Objects on a Budget

Figure 4.3 shows corresponding results on the SIVAL dataset. The budget T is set to 300 secs, again so that all unlabeled data would be exhausted in $\sim$20 iterations. Our approach is consistently better than both baselines, as seen in the bottom right plot above. For some categories (such as “dirtyworkgloves”), none of the approaches improve with more labels, apparently due to those objects’ non-descript texture/color. While the differences between the approaches may appear to be smaller that what we see for HOHA, they are consistent and


Our approach
Myopic active batch
Label : +1, Cost : 7 secs
Label : -1, Cost : 7 secs
Label : -1, Cost : 31 secs
Label : +1, Cost : 8 secs
Label : +1, Cost : 19 secs
Label : -1, Cost : 32 secs

(a) SIVAL dataset, category - "bluescrunge"



(b) Hollywood dataset, category - "stand up"

Figure 4.4: (a) Example batch selections made by our approach (left) and the myopic baseline (right) for the SIVAL "bluescrunge" object on the first iteration, with a budget of 60 secs. (b) Selections made by our approach and the two baselines on Hollywood "stand up" category for a budget of 13 seconds. Our approach is able to select both less expensive and more informative examples, while sticking within the allowed budget as closely as possible.

significant considering that the results are averaged over five random initializations and 25 categories. Moreover, to achieve about 90% of the ultimate accuracy level possible on this dataset (0.7 AUROC), our method requires notably less cost: about 43% less annotation cost than the passive selector, and 20% less than the myopic selector.

Figure 4.4 shows example batch selections made by our approach and the baseline techniques on the two datasets. The examples illustrates the main advantage of our approach: we are able to select both less expensive and more informative examples, while sticking within the allowed budget as closely as possible.

4.2.4 Effect of the Budget Size

Next we study the impact of increasing budget sizes. We expect the far-sightedness of our approach to offer particular advantages for larger budgets. This is because when a large number of examples is selected we expect the classifier to change a lot and baselines techniques that depend on the unchanged current classifier to choose examples might not be able to incorporate this change. For this experiment, we vary the size of the budget, and measure the accuracy of our method and the baselines at a fixed cost for each budget (approximately $\frac{1}{4}$ of the total unlabeled pool's cost). The range of budget sizes tested was set so as to exhaust all unlabeled data in about $10, \ldots, 40$ iterations.

Figure 4.5 shows the results, for two example categories from SIVAL and HOHA. We include a minimal budget size to illustrate that for a budget allowing only $\approx$a single item to be selected, MAB and our approach would be almost equivalent (see leftmost points on both plots). As expected, for larger budgets, the myopic choices drop in accuracy, sometimes be-

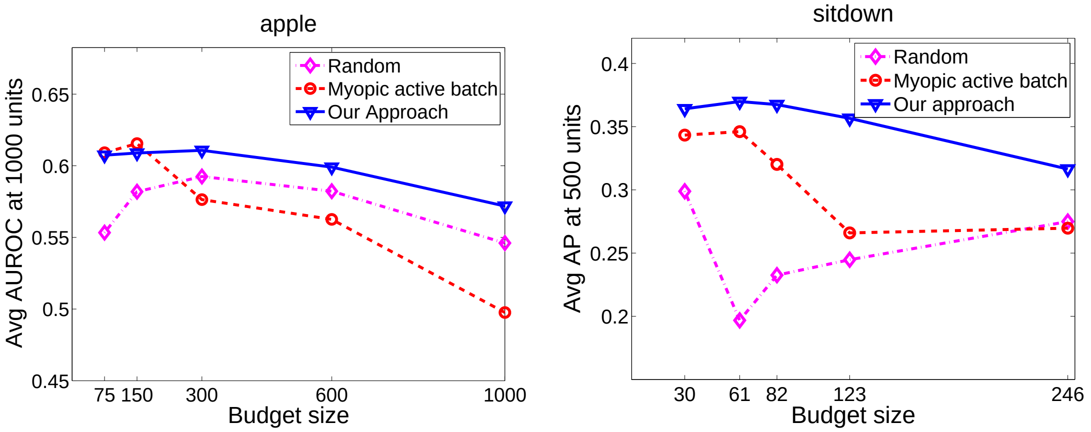


Figure 4.5: Active learning performance as a function of increasing budget size. The quality of our far-sighted selections remains more stable for larger budgets.

low the random baseline. Passive selection's accuracy is stable across budget sizes since it is simply random. Our approach shows the least degradation—a consequence of considering how the classifier changes if we were to obtain the most probable labels on the candidate examples for selection. This is a key result, given that real recognition systems drawing on a pool of annotators must be able to pick a large batch of jobs wisely in order to farm them out in parallel.

4.2.5 Comparison to State-of-the-Art Batch Selection

Next we provide comparisons with the state-of-the-art batch-mode active learning (BMAL) method of [45] on a CBIR task with Corel. The approach of [45] considers a min-max view of the SVM objective function and derives a selection criterion for batches that minimizes both the total classification uncertainty and the redundancy among the selected examples. The two BMAL variants use quadratic programming ($\mathrm{SVM}_{BMAL}^{SS(QP)}$) and combinatorial opti-

Precision	Batch Size					
	5	10	15	20	25	30
Ours	0.620	0.734	0.809	0.853	0.888	0.905
$\text{SVM}_{BMAL}^{SS(QP)}$	0.640	0.718	0.798	0.835	0.860	0.886
$\text{SVM}_{BMAL}^{SS(CO)}$	0.622	0.717	0.776	0.835	0.868	0.889

Recall	Batch Size					
	5	10	15	20	25	30
Ours	0.321	0.371	0.417	0.452	0.477	0.503
$\text{SVM}_{BMAL}^{SS(QP)}$	0.332	0.373	0.423	0.452	0.468	0.490
$\text{SVM}_{BMAL}^{SS(CO)}$	0.321	0.377	0.412	0.447	0.471	0.493

Table 4.2: Corel results. **Top:** The average precision of the top 20 retrievals with different batch sizes. **Bottom:** The average recall of the top 100 retrievals with different batch sizes (evaluation done as prescribed in [45]).

mization ($\text{SVM}_{BMAL}^{SS(CO)}$). While their approach is intended for fixed-size batches, and ours allows variable-sized batches, we can still test our method in this setting since it is a special case (i.e., budget=batch size). We replicate the experimental setup given by the authors, using 200 random queries, and applying the same kernel [93], SVM parameters, and scoring criteria (see [45] for details).

Table 4.2 shows the results. Our results are comparable, if not better, than the state-of-the-art, and the gains are a bit more apparent with larger batch sizes. We attribute our gains to our method's inclusion of the expected classifier change. See [45] for more results from other active selection baselines (including [98, 12]), all of which generally underperform BMAL, and thus our method, for this data. Moreover, our method provides a more general solution for the batch selection problem since it can better handle variably sized batches given a fixed budget as we show in the next section.

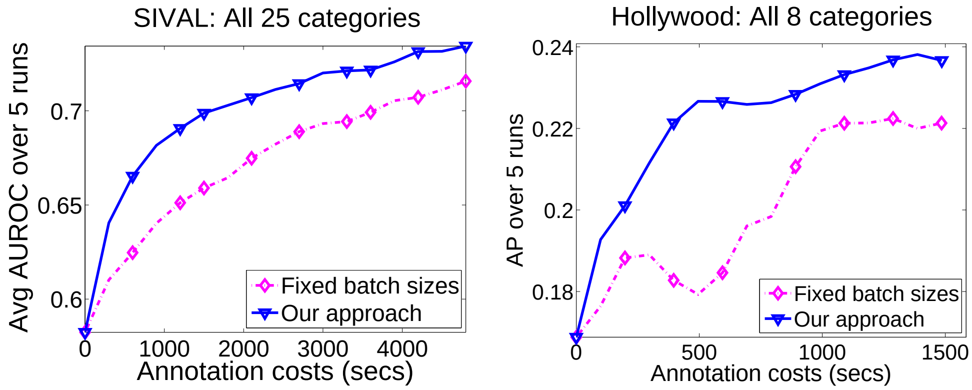


Figure 4.6: Comparison of active batch selections when using our budgeted approach versus restricting selections to a fixed batch size.

4.2.6 Impact of Budgets versus Fixed-Size Batches

Finally, we examine the impact of being able to select variable-sized batches according to a fixed budget, as compared to fixing a batch size. We implement a QP-solver for the BMAL approach ([45]) and run experiments on SIVAL and HOHA. Since the BMAL baseline must choose k examples at each iteration (regardless of the cost), we set k to the budget T divided by the dataset's mean cost. We set the BMAL regularization parameter as $\lambda = 1$, as suggested by the authors [45]. We found this to be a reasonable choice to illustrate the advantage of being able to choose variable-sized batches under budgeted learning.

Figure 4.6 shows the results. On both datasets, our budgeted selection performs better than a fixed-batch choice. This reinforces our claim that the higher the cost variability among the unlabeled data, the more crucial it is to optimize selections for the given budget. Our method essentially picks a mixture of less/more expensive examples so as to best utilize the allowed annotation budget, whereas a method limited to choosing fixed-size batches is misled into choosing a seemingly informative batch that may be overly expensive in reality.

4.2.7 Computation Time

Our solution is quite efficient since it uses an LP and QP for which several efficient solvers exist. In our experiments, convergence typically occurs in $\sim 10 - 15$ iterations. Our Matlab code takes about 0.6 seconds per batch selection for 200 unlabeled examples, and 4 minutes for 5000 examples on a 2.8 GHz processor.

4.3 Discussion

In this chapter, I formalized the problem of far-sighted active learning on a budget, and proposed a new method for optimally selecting a set of examples for a support vector machine classifier under these conditions. I provided an efficient iterative minimization technique that balances candidate examples' costs and value when selected in batches. Experiments on two benchmark datasets show the practical advantages when compared to passive and greedy myopic alternatives, as well as an existing active batch selection baseline. Overall the results are quite encouraging and suggest that the proposed approach enables wise use of budgeted supervision.

In contrast to previous methods, our approach considers how much the classifier objective changes if we were to obtain the most probable labels on the candidate examples for selection. We find that this aspect is critical to performance, particularly in the practical scenario where one wants to set a large budget at each iteration. Note that the VOI based criterion used in Chapter 3 also considered the change in classifier objective by computing the expected value of reduction in risk over all possible labels for a single unlabeled example. In contrast, in this chapter we considered the most *optimistic* change in the classifier objective possible by a set of labels. This is because when selecting a large batch of examples the expected value would be more noisy since it would incorporate VOI values from a combinatorial number of possible labels, out of which only a small subset will be useful.

Furthermore, the proposed approach is the first batch active selection strategy that is sensitive to the costs of labeling, and the first method to allow sets of training examples to be chosen so as to meet a prescribed budget. The efficiency of the component optimization steps also makes it rather scalable to large unlabeled data pools since it solves an LP and a QP for which several efficient solvers exist.

While I have shown applications of our budgeted active learning approach for visual recognition tasks, our method is general enough to apply to several other domains both in vision and machine learning since we directly learn an SVM classifier under a given budget of supervision. For example, one could use our approach for learning to classify news documents into high-level categories given an annotation budget. In this setting the cost of annotating a news document could similarly be obtained based on the total length of the document or by setting up user experiments. Similary, we could train classifiers under an annotation budget to differentiate between urgent versus non-urgent voicemails, where the cost of providing annotations is proportional to the length of the voicemail.

Like most existing active selection methods our batch selection approach has a time complexity that is cubic in the size of the unlabeled pool. In the next chapter, I address the problem of large-scale active selection where there are millions of unlabeled examples and even approaches that are linear in the size of the unlabeled pool are not practical.

Chapter 5

Sub-linear Time Active Selection for Web-scale Data

A practical paradox with pool-based active learning algorithms is that their intended value—to reduce learning time by choosing informative examples to label first—conflicts with the real expense of applying them to very large unlabeled datasets. Generally methods today are tested in somewhat canned scenarios: the implementor has a moderately sized labeled dataset, and simply withholds the labels from the learner until a given point is selected, at which point the "oracle" reveals the label. In reality, one would like to deploy an active learner on a massive *truly* unlabeled data pool (e.g., all documents on the Web) and let it crawl for the instances that appear most valuable for the target classification task. The problem is that a scan of millions of points is rather expensive to compute exhaustively, and thus defeats the purpose of improving overall learning efficiency.

In this chapter, I consider the problem of performing active selection on large-scale datasets where the computational cost of selection outweighs other considerations. To exploit such massive unlabeled pools, a fast (sub-linear time) search method to identify the most informative points to a given classifier is required. Approximate heuristic solutions such as the '59-trick' [95], where 59 examples are randomly chosen and ranked using the selection function, exist for lowering the computational cost of active selection. However, such techniques do not provide approximation guarantees except for trivial situations such as when the examples are uniformly distributed and it is unclear if they would work for any general data distribution.

Thus, I address the following *hyperplane-to-point* search problem: given a database of points, which examples are nearest to a novel hyperplane query? We call this the *nearest neighbor to a query hyperplane* (NNQH) problem. The NNQH search problem ties in with pool-based active learning through the simple-margin selection criterion for linear SVM classifiers. In this active selection scheme, given a hyperplane classifier and an unlabeled pool of vector data $\mathcal{U} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, the most informative point (in terms of reduction in version space) is the one that minimizes the distance to the current decision boundary: $\boldsymbol{x}^* = \mathrm{argmin}_{\boldsymbol{x}_i \in \mathcal{U}} |\boldsymbol{w}^T \boldsymbol{x}_i|$. This is a widely used margin-based selection criterion [98, 85, 15] and it has been shown to substantially reduce total human annotation effort.

A large number of existing algorithms provide efficient data structures for point-to-point retrieval tasks with various useful distance functions, producing either exact or approximate near neighbors while forgoing a brute force scan through all database items, e.g., [34, 101, 38, 1, 19, 120, 59]. By comparison, much less work considers how to efficiently handle hyperplane to point search, which is useful for active learning in the context of the margin-based selection criterion of [98].

Locality-sensitive hashing (LSH) methods devise randomized hash functions that map similar points to the same hash buckets, so that only a subset of the database must be searched after hashing a novel query [38, 1, 19]. A related family of methods design Hamming space embeddings that can be indexed efficiently (e.g., [91, 84, 120]).

However, in contrast to our goal, all such techniques are intended for vector/point data. A few researchers have recently examined approximate search tasks involving subspaces. In [7], a Euclidean embedding is developed such that the norm in the embedding space directly reflects the principal angle-based distance between the original subspaces. Another method to find the nearest subspace for a point query is given in [72], though it is limited to relatively low-dimensional data due to its preprocessing time/space requirements.

Therefore, I propose two solutions for approximate hyperplane-to-point search. For each, I introduce randomized hash functions that offer query times sub-linear in the size of the database, and provide bounds for the approximation error of the neighbors retrieved. The first approach devises a two-bit hash function that is locality-sensitive for the angle between the hyperplane normal and a database point. The second approach embeds the inputs such that the Euclidean distance reflects the hyperplane distance, thereby making them searchable with existing approximate nearest neighbor algorithms for vector data. While the preprocessing in our first method is more efficient, our second method has stronger accuracy guarantees.

The two NNQH solutions supply exactly the hash functions needed to rapidly identify the most uncertain examples for a linear SVM classifier according to the simple-margin selection criterion. Therefore, our algorithms make it possible to benefit from *both* massive unlabeled collections as well as actively chosen label requests.

In the following, I will formally define the problem and provide background definitions for Locality-Sensitive Hashing (LSH), which is critical to our solution. Then I will provide two hashing based solutions for the NNQH problem and explain how these can be applied for pool-based active learning. Finally, in Section 5.2, I demonstrate our algorithms' significant practical impact for large-scale active learning with SVM classifiers. Our results show that our method helps scale-up active learning for realistic problems with massive unlabeled pools on the order of millions of examples.[1]

5.1 Hashing Hyperplanes to Near Points

We consider the following retrieval problem. Given a database $\mathcal{D} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$ of N points in $\mathbb{R}^d$, the goal is to retrieve the points from the database that are closest to a given *hyperplane* query whose normal is given by $\boldsymbol{w} \in \mathbb{R}^d$. We call this the *nearest neighbor to a query hyperplane* (NNQH) problem. Without loss of generality, we assume that the hyperplane passes through origin, and that each $\boldsymbol{x}_i$, $\boldsymbol{w}$ is unit norm. We see in later sections that these assumptions do not affect our solution.

The Euclidean distance of a point $\boldsymbol{x}$ to a given hyperplane $h_{\boldsymbol{w}}$ parameterized by normal $\boldsymbol{w}$ is:

$$d(h_{\boldsymbol{w}}, \boldsymbol{x}) = \|(\boldsymbol{x}^T\boldsymbol{w})\boldsymbol{w}\| = |\boldsymbol{x}^T\boldsymbol{w}|. \tag{5.1}$$

Thus, the goal for the NNQH problem is to identify those points $\boldsymbol{x}_i \in \mathcal{D}$ that minimize $|\boldsymbol{x}_i^T\boldsymbol{w}|$. Note that this is in contrast to traditional proximity problems, e.g., nearest or farthest neighbor retrieval, where the goal is to *maximize* $\boldsymbol{x}^T\boldsymbol{w}$ or $-\boldsymbol{x}^T\boldsymbol{w}$, respectively. Hence, existing approaches are not directly applicable to this problem.

We formulate two algorithms for NNQH. Our first approach maps the data to binary keys that are locality-sensitive for the angle between the hyperplane normal and a database point, thereby permitting sub-linear time retrieval with hashing. Our second approach computes a

---

[1] The contents of this chapter were published in [49].

sparse Euclidean embedding for the query hyperplane that maps the desired search task to one handled well by existing approximate nearest-point methods.

In the following, I will first provide necessary background on Locality-Sensitive Hashing (LSH). The subsequent two sections describe each approach in turn, and Section 5.1.5 reviews their trade-offs. Finally, in Section 5.1.6, I will explain how either method can be applied to large-scale active learning.

5.1.1 Background: Locality-Sensitive Hashing

Informally, LSH [38] requires randomized hash functions guaranteeing that the probability of collision of two vectors is inversely proportional to their "distance", where "distance" is defined according to the task at hand. Since similar points are assured (w.h.p.) to fall into the same hash bucket, one need only search those database items with which a novel query collides in the hash table.

Formally, let $d(\cdot,\cdot)$ be a distance function over items from a set S, and for any item $p \in S$, let $B(p, r)$ denote the set of examples from S within radius r from p.

Definition 5.1.1. *[38] Let $h_{\mathcal{H}}$ denote a random choice of a hash function from the family $\mathcal{H}$. The family $\mathcal{H}$ is called $(r, r(1+\epsilon), p_1, p_2)-$sensitive for $d(\cdot,\cdot)$ when, for any $q, p \in S$,*

- *if* $p \in B(q, r)$ *then* $\Pr[h_{\mathcal{H}}(q) = h_{\mathcal{H}}(p)] \geq p_1$,
- *if* $p \notin B(q, r(1+\epsilon))$ *then* $\Pr[h_{\mathcal{H}}(q) = h_{\mathcal{H}}(p)] \leq p_2$.

For a family of functions to be useful, it must satisfy $p_1 > p_2$. A k-bit LSH function computes a hash "key" by concatenating the bits returned by a random sampling of $\mathcal{H}$: $g(p) = \left[h_{\mathcal{H}}^{(1)}(p), h_{\mathcal{H}}^{(2)}(p), \ldots, h_{\mathcal{H}}^{(k)}(p)\right]$. Note that the probability of collision for close points is thus at least p_1^k, while for dissimilar points it is at most p_2^k. During a preprocessing stage, all database points are mapped to a series of l hash tables indexed by independently constructed $g_1, \ldots, g_l$, where each g_i is a k-bit function. Then, given a query q, an exhaustive search is carried out only on those examples in the union of the l buckets to which q hashes. These candidates contain the (r, ϵ)-nearest neighbors (NN) for q, meaning if q has a neighbor within radius r, then with high probability some example within radius $r(1 + \epsilon)$ is found.

In [38] an LSH scheme using projections onto single coordinates is shown to be locality-sensitive for the Hamming distance over vectors. For that hash function, $\rho = \frac{\log p_1}{\log p_2} \leq \frac{1}{1+\epsilon}$, and using $l = N^\rho$ hash tables, a $(1 + \epsilon)$-approximate solution can be retrieved in time $O(N^{\frac{1}{(1+\epsilon)}})$. Related formulations and LSH functions for other distances have been explored (e.g., [19, 1, 47]). Our contribution is to define two locality-sensitive hash functions for the NNQH problem.

5.1.2 Hyperplane Hashing based on Angle Distance (H-Hash)

Recall that we want to retrieve the database vector(s) $\boldsymbol{x}$ for which $|\boldsymbol{w}^T\boldsymbol{x}|$ is minimized. If the vectors are unit norm, then this means that for the "good" (close) database vectors, $\boldsymbol{w}$ and $\boldsymbol{x}$ are almost perpendicular (see Figure 5.1). Let $\theta_{\boldsymbol{x},\boldsymbol{w}}$ denote the angle between $\boldsymbol{x}$ and $\boldsymbol{w}$. We define the distance $d(\cdot, \cdot)$ in Definition 5.1.1 to reflect how far from perpendicular $\boldsymbol{w}$ and $\boldsymbol{x}$ are:

$$d_\theta(\boldsymbol{x}, \boldsymbol{w}) = (\theta_{\boldsymbol{x},\boldsymbol{w}} - \pi/2)^2. \tag{5.2}$$

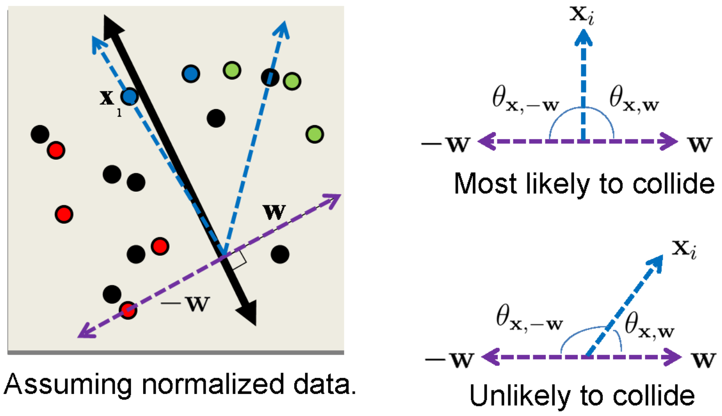


Figure 5.1: In order to retrieve those points for which $|\boldsymbol{w}^T\boldsymbol{x}|$ is small, we want probable collision for perpendicular vectors.

Consider the following two-bit function that maps two input vectors $\boldsymbol{a}, \boldsymbol{b} \in \Re^d$ to $\{0,1\}^2$:

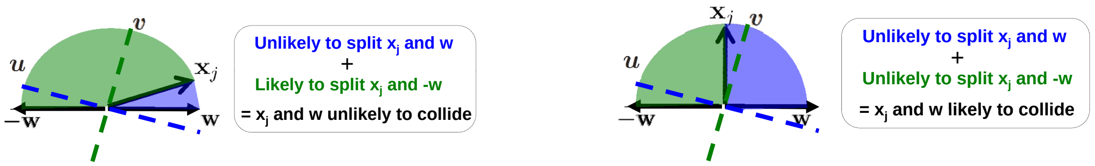
$$h_{\boldsymbol{u},\boldsymbol{v}}(\boldsymbol{a},\boldsymbol{b}) = [h_{\boldsymbol{u}}(\boldsymbol{a}), h_{\boldsymbol{v}}(\boldsymbol{b})] = [\text{sign}(\boldsymbol{u}^T\boldsymbol{a}), \text{sign}(\boldsymbol{v}^T\boldsymbol{b})], \tag{5.3}$$

where $h_{\boldsymbol{u}}(\boldsymbol{a}) = \text{sign}(\boldsymbol{u}^T\boldsymbol{a})$ returns 1 if $\boldsymbol{u}^T\boldsymbol{a} \geq 0$, and 0 otherwise, and $\boldsymbol{u}$ and $\boldsymbol{v}$ are sampled independently from a standard d-dimensional Gaussian, i.e., $\boldsymbol{u}, \boldsymbol{v} \sim \mathcal{N}(0, I)$.

We define our **hyperplane hash** (H-Hash) function family $\mathcal{H}$ as:

$$h_{\mathcal{H}}(\boldsymbol{z}) = \begin{cases} h_{\boldsymbol{u},\boldsymbol{v}}(\boldsymbol{z},\boldsymbol{z}), & \text{if } \boldsymbol{z} \text{ is a database point vector,} \\ h_{\boldsymbol{u},\boldsymbol{v}}(\boldsymbol{z},-\boldsymbol{z}), & \text{if } \boldsymbol{z} \text{ is a query hyperplane vector.} \end{cases}$$

The idea behind our H-Hash solution is that we generate two hash bits using independent random vectors $\boldsymbol{u}$ and $\boldsymbol{v}$: one for comparing the angle between $\boldsymbol{w}$ and $\boldsymbol{x}$ and the other for $-\boldsymbol{w}$ and $\boldsymbol{x}$. If $\boldsymbol{x}$ and $\boldsymbol{w}$ are almost parallel to each other (see Figure 5.2(a)), random vector $\boldsymbol{v}$ has a large probability of assigning different bits to $\boldsymbol{x}$ and $-\boldsymbol{w}$ because of the large angle between $\boldsymbol{x}$ and $-\boldsymbol{w}$. On the other hand, if $\boldsymbol{x}$ and $\boldsymbol{w}$ are perpendicular to each other as shown in Figure 5.2(b)), then neither $\boldsymbol{u}$ nor $\boldsymbol{v}$ have a high chance of assigning different bits.


v
u
x_j
−w
w
Unlikely to split x_j and w
+
Likely to split x_j and -w
= x_j and w unlikely to collide
x_j
v
u
−w
w
Unlikely to split x_j and w
+
Unlikely to split x_j and -w
= x_j and w likely to collide

(a) For parallel vectors

(b) For perpendicular vectors

Figure 5.2: The basic intuition behind our H-Hash solution. We generate two independent random vectors $\boldsymbol{u}$ and $\boldsymbol{v}$: one to capture the angle between $\boldsymbol{x}$ and $\boldsymbol{w}$, and the other for $\boldsymbol{x}$ and $-\boldsymbol{w}$. The probability that the vectors do not split the corresponding angles is highest when $\boldsymbol{x}$ and $\boldsymbol{w}$ are perpendicular, as seen in the figure.

Therefore, when considering bits assigned by the two random vectors together, the probability of collision is highest for vectors that are perpendicular.

5.1.2.1 Proof of Locality-sensitivity for Hyperplane Hashing

Next, we formally prove that this family of hash functions is locality-sensitive (Definition 5.1.1).

Claim 5.1.2. *The family* $\mathcal{H}$ *is* $\left(r, r(1+\epsilon), \frac{1}{4} - \frac{1}{\pi^2}r, \frac{1}{4} - \frac{1}{\pi^2}r(1+\epsilon)\right)$*-sensitive for the distance* $d_\theta(\cdot,\cdot)$*, where* $r, \epsilon > 0$.

Proof. Since the vectors $\boldsymbol{u}$, $\boldsymbol{v}$ used by hash function $h_{\boldsymbol{u},\boldsymbol{v}}$ are sampled independently, then for a query hyperplane vector $\boldsymbol{w}$ and a database point vector $\boldsymbol{x}$,

$$\begin{aligned}\Pr[h_{\mathcal{H}}(\boldsymbol{w}) = h_{\mathcal{H}}(\boldsymbol{x})] &= \Pr[h_{\boldsymbol{u}}(\boldsymbol{w}) = h_{\boldsymbol{u}}(\boldsymbol{x}) \text{ and } h_{\boldsymbol{v}}(-\boldsymbol{w}) = h_{\boldsymbol{v}}(\boldsymbol{x})], \\ &= \Pr[h_{\boldsymbol{u}}(\boldsymbol{w}) = h_{\boldsymbol{u}}(\boldsymbol{x})]\ \Pr[h_{\boldsymbol{v}}(-\boldsymbol{w}) = h_{\boldsymbol{v}}(\boldsymbol{x})]. \end{aligned} \tag{5.4}$$

Next, we use the following fact proven in [39],

$$\Pr[\text{sign}(\boldsymbol{u}^T\boldsymbol{a}) = \text{sign}(\boldsymbol{u}^T\boldsymbol{c})] = 1 - \frac{\theta_{\boldsymbol{a},\boldsymbol{c}}}{\pi}, \tag{5.5}$$

where $\boldsymbol{u}$ is sampled as defined above, and $\theta_{\boldsymbol{a},\boldsymbol{c}}$ denotes the angle between the two vectors $\boldsymbol{a}$ and $\boldsymbol{c}$.

Using 5.4 and 5.5, we get:

$$\Pr[h_{\mathcal{H}}(\boldsymbol{w}) = h_{\mathcal{H}}(\boldsymbol{x})] = \frac{\theta_{\boldsymbol{x},\boldsymbol{w}}}{\pi}\left(1 - \frac{\theta_{\boldsymbol{x},\boldsymbol{w}}}{\pi}\right) = \frac{1}{4} - \frac{1}{\pi^2}\left(\theta_{\boldsymbol{x},\boldsymbol{w}} - \frac{\pi}{2}\right)^2.$$

Hence, when $\left(\theta_{\boldsymbol{x},\boldsymbol{w}} - \frac{\pi}{2}\right)^2 \leq r$, $\Pr[h_{\mathcal{H}}(\boldsymbol{w}) = h_{\mathcal{H}}(\boldsymbol{x})] \geq \frac{1}{4} - \frac{r}{\pi^2} = p_1$. Similarly, for any $\epsilon > 0$ such that $\left(\theta_{\boldsymbol{x},\boldsymbol{w}} - \frac{\pi}{2}\right)^2 \geq r(1+\epsilon)$, $\Pr[h_{\mathcal{H}}(\boldsymbol{w}) = h_{\mathcal{H}}(\boldsymbol{x})] \leq \frac{1}{4} - \frac{r(1+\epsilon)}{\pi^2} = p_2$. ☐

We note that unlike traditional LSH functions, ours are asymmetric. That is, to hash a database point $\boldsymbol{x}$ we use $h_{\boldsymbol{u},\boldsymbol{v}}(\boldsymbol{x},\boldsymbol{x})$, whereas to hash a query hyperplane $\boldsymbol{w}$, we use $h_{\boldsymbol{u},\boldsymbol{v}}(\boldsymbol{w},-\boldsymbol{w})$. The purpose of the two-bit hash is to constrain the angle with respect to both $\boldsymbol{w}$ and $-\boldsymbol{w}$, so that we do not simply retrieve examples for which we know only that $\boldsymbol{x}$ is $\pi/2$ *or less* away from $\boldsymbol{w}$.

With these functions in hand, we can now form hash keys by concatenating k two-bit pairs from k hash functions from $\mathcal{H}$, store the database points in the hash tables, and query with a novel hyperplane to retrieve its closest points (see Section 5.1.1).

5.1.2.2 Approximation Guarantees for Hyperplane Hashing

We next derive the approximation guarantees and prove the correctness of this scheme by adapting the proof of Theorem 1 in [38]. We first recall the data-structure used for LSH. We store l-hash tables and every hash table contains k-bit hash keys. So, the s-th hash table has a corresponding function $g_s : \mathbb{R}^d \rightarrow 0,1^k$ that given a vector, maps the vector to k-

bit hash keys. Each function g_s is obtained by randomly sampling $\mathcal{H}$ with replacement: $g_s = (h_{s_1}, h_{s_2}, \dots, h_{s_k})$.

Here, we show that using locality-sensitive hash functions for the distance $d_\theta(\cdot,\cdot)$ along with hash tables, we can get a $(1+\epsilon)$-approximate solution to our hyperplane-to-point search problem in sub-linear time.

In particular, we prove the following theorem:

Theorem 5.1.3. *Let $\mathcal{H}$ be a family of $(r, r(1+\epsilon), p_1, p_2)$-locality hash functions (see Definition 3.1 (*Main Text*)), with $p_1 > p_2$. Now given a database of N points, we set $k = \log_{1/p_2} N$ and $l = N^\rho$, where $\rho = \frac{\log p_1}{\log p_2}$. Now using $\mathcal{H}$ along with l-hash tables over k-bits, given a hyperplane query $\boldsymbol{w}$, with probability at least $\frac{1}{2} - \frac{1}{e}$, the algorithm solves the (r, ϵ)-neighbor problem, i.e., if there exists a point $\boldsymbol{x}$ s.t. $d_\theta(\boldsymbol{x}, \boldsymbol{w}) \leq (1+\epsilon)r$, then the algorithm will return the point with probability $\geq 1/2 - 1/e$. The retrieval time is bounded by $O(N^\rho)$.*

Proof. Our proof is a simple adaption of the proof of Theorem 1 in Gionis et al. [38]. We present it here for the sake of completeness.

Following [38] we prove two properties:

P1: Let $\boldsymbol{x}^*$ be a point such that $d_\theta(\boldsymbol{x}^*, \boldsymbol{w}) \leq r$, then $g_j(\boldsymbol{x}^*) = g_j(\boldsymbol{w})$ for some $1 \leq j \leq l$ with probability $1/2 - 1/e$.

Proof: Now we know that

$$\Pr[g_j(\boldsymbol{x}^*) = g_j(\boldsymbol{w})] \geq p_1^k = p_1^{\log_{1/p_2} N} = N^{-\rho}.$$

Hence,

$$\Pr[g_j(\boldsymbol{x}^*) \neq g_j(\boldsymbol{w}), \forall j] = \Pi_j \Pr[g_j(\boldsymbol{x}^*) \neq g_j(\boldsymbol{w})] \leq (1 - N^{-\rho})^l = (1 - N^{-\rho})^{N^\rho} \leq 1/e.$$

Thus, P1 holds with probability $> 1 - 1/e$.

P2: Consider the set $S = \{\boldsymbol{y}$ s.t., $d_\theta(\boldsymbol{y}, \boldsymbol{w}) > r(1+\epsilon)$ and $g_j(\boldsymbol{y}) = g_j(\boldsymbol{w})$ for some $j\}$. Then $|S| \leq cl$ with probability at least $1 - 1/c$.
Proof: Now if $d_\theta(\boldsymbol{y}, \boldsymbol{w}) > r(1+\epsilon)$, then $\Pr[h(\boldsymbol{y}) = h(\boldsymbol{w})] \leq p_2$. Hence, for any j,

$$\Pr[g_j(\boldsymbol{y}) = g_j(\boldsymbol{w})] \leq p_2^k = p_2^{\log_{1/p_2} n} = 1/N.$$

Thus the expected number of collisions for a single j is $N \cdot \Pr[g_j(\boldsymbol{y}) = g_j(\boldsymbol{w})] = 1$ and hence $E[|S|] = l$. Therefore, by Markov's inequality:

$$\Pr(|S| > cl) \leq 1/c.$$

Hence, P2 holds with probability $> 1 - 1/c$.

The theorem now immediately follows from P1 and P2, as by P1 we are assured of retrieving the point $\boldsymbol{x}^*$ with probability $> 1/2 - 1/e$, and by P2 we are assured of not looking at more than $cl = O(N^\rho)$ points. $\square$

In summary, we have shown that with high probability our LSH scheme will return a point within a distance $(1+\epsilon)r$, where $r = \min_i d_\theta(\boldsymbol{x}_i, \boldsymbol{w})$, in time $O(N^\rho)$, where $\rho = \frac{\log p_1}{\log p_2}$. As $p_1 > p_2$, we have $\rho < 1$, i.e., the approach takes sub-linear time for all values of r, ϵ. Furthermore, as $p_1 = \frac{1}{4} - \frac{r}{\pi^2}$, and $p_2 = \frac{1}{4} - \frac{r(1+\epsilon)}{\pi^2}$, ρ can also be bounded as $\rho \leq \frac{1 - \log(1 - \frac{4r}{\pi^2})}{1 + \frac{\epsilon}{1 + \frac{\pi^2}{4r} \log 4}}$. Note that this bound for ρ is dependent on r, and is more efficient for larger values of r.

5.1.2.3 Improved Hyperplane Hashing based on Angle Distance (IH-Hash)

We can further improve the approximation guarantees of H-Hash based on the observation that the bits computed using $\boldsymbol{u}$ and $\boldsymbol{v}$ need not be ordered. Recall that in H-Hash bits computed using $\boldsymbol{u}$ always compare $\boldsymbol{x}$ with $\boldsymbol{w}$ and those computed using $\boldsymbol{v}$ compare $\boldsymbol{x}$ with $-\boldsymbol{w}$, which is too restrictive. Therefore, by mapping the 2-bit vectors obtained through the random hyperplanes to a single bit using the logical exclusive or function ($\oplus$), we can double the probability of p_1 for H-Hash.

We define our **improved hyperplane hash** (IH-Hash) function family $\mathcal{I}$ as:

$$h_{\mathcal{I}}(\boldsymbol{z}) = \begin{cases} h_{\boldsymbol{u}}(\boldsymbol{z}) \oplus h_{\boldsymbol{v}}(\boldsymbol{z}), & \text{if } \boldsymbol{z} \text{ is a database point vector,} \\ h_{\boldsymbol{u}}(\boldsymbol{z}) \oplus h_{\boldsymbol{v}}(-\boldsymbol{z}), & \text{if } \boldsymbol{z} \text{ is a query hyperplane vector.} \end{cases}$$

Claim 5.1.4. *The family* $\mathcal{I}$ *is* $\left(r, r(1+\epsilon), \frac{1}{2} - \frac{1}{\pi^2}r, \frac{1}{2} - \frac{1}{\pi^2}r(1+\epsilon)\right)$*-sensitive for the distance* $d_\theta(\cdot,\cdot)$*, where* $r, \epsilon > 0$.

Proof. Since the vectors $\boldsymbol{u}$, $\boldsymbol{v}$ used by hash function $h_{\boldsymbol{u},\boldsymbol{v}}$ are sampled independently, then for a query hyperplane vector $\boldsymbol{w}$ and a database point vector $\boldsymbol{x}$,

$$\begin{aligned}
\Pr[h_{\mathcal{I}}(\boldsymbol{w}) = h_{\mathcal{I}}(\boldsymbol{x})]) &= \Pr[(h_{\boldsymbol{u}}(\boldsymbol{w}) \oplus h_{\boldsymbol{v}}(-\boldsymbol{w})) = (h_{\boldsymbol{u}}(\boldsymbol{x}) \oplus h_{\boldsymbol{v}}(\boldsymbol{x}))], \\
&= \Pr[(h_{\boldsymbol{u}}(\boldsymbol{w}) = h_{\boldsymbol{u}}(\boldsymbol{x}) \text{ and } h_{\boldsymbol{v}}(-\boldsymbol{w}) = h_{\boldsymbol{v}}(\boldsymbol{x})) \\
&\quad\ \text{or } (h_{\boldsymbol{u}}(\boldsymbol{w}) \neq h_{\boldsymbol{u}}(\boldsymbol{x}) \text{ and } h_{\boldsymbol{v}}(-\boldsymbol{w}) \neq h_{\boldsymbol{v}}(\boldsymbol{x}))] \\
&= \Pr[h_{\boldsymbol{u}}(\boldsymbol{w}) = h_{\boldsymbol{u}}(\boldsymbol{x})] \ \Pr[h_{\boldsymbol{v}}(-\boldsymbol{w}) = h_{\boldsymbol{v}}(\boldsymbol{x})] \\
&\quad + \Pr[h_{\boldsymbol{u}}(-\boldsymbol{w}) = h_{\boldsymbol{u}}(\boldsymbol{x})] \ \Pr[h_{\boldsymbol{v}}(\boldsymbol{w}) = h_{\boldsymbol{v}}(\boldsymbol{x})] \qquad (5.6)
\end{aligned}$$

Using 5.6 and 5.5, we get:

$$\begin{aligned}\Pr[h_{\mathcal{I}}(\boldsymbol{w}) = h_{\mathcal{I}}(\boldsymbol{x})] &= \frac{\theta_{\boldsymbol{x},\boldsymbol{w}}}{\pi}\left(1 - \frac{\theta_{\boldsymbol{x},\boldsymbol{w}}}{\pi}\right) + \left(1 - \frac{\theta_{\boldsymbol{x},\boldsymbol{w}}}{\pi}\right)\frac{\theta_{\boldsymbol{x},\boldsymbol{w}}}{\pi} \\ &= \frac{1}{2} - \frac{1}{\pi^2}\left(\theta_{\boldsymbol{x},\boldsymbol{w}} - \frac{\pi}{2}\right)^2. \qquad (5.7)\end{aligned}$$

Hence, when $\left(\theta_{\boldsymbol{x},\boldsymbol{w}} - \frac{\pi}{2}\right)^2 \leq r$, $\Pr[h_{\mathcal{H}}(\boldsymbol{w}) = h_{\mathcal{H}}(\boldsymbol{x})] \geq \frac{1}{2} - \frac{r}{\pi^2} = p_1$. Similarly, for any $\epsilon > 0$ such that $\left(\theta_{\boldsymbol{x},\boldsymbol{w}} - \frac{\pi}{2}\right)^2 \geq r(1+\epsilon)$, $\Pr[h_{\mathcal{H}}(\boldsymbol{w}) = h_{\mathcal{H}}(\boldsymbol{x})] \leq \frac{1}{2} - \frac{r(1+\epsilon)}{\pi^2} = p_2$. ☐

Note that the p_1 obtained above for IH-Hash is exactly twice the p_1 obtained using H-Hash. Hence, the factor $\rho = \frac{\log p_1}{\log p_2}$ improves upon H-Hash, remaining lower for lower values of ϵ leading to better approximation guarantees. Specifically, ρ can be bounded as $\rho \leq \frac{1 - \log(1 - \frac{2r}{\pi^2})}{1 + \frac{\epsilon}{1 + \frac{\pi^2}{2r}\log 2}}$.

5.1.3 Embedded Hyperplane Hashing based on Euclidean Distance (EH-Hash)

Our second approach for the NNQH problem relies on a Euclidean embedding for the hyperplane and points. Figure 5.3 illustrates the basic intuition behind our solution. It offers stronger bounds than H-Hash, but at the expense of more preprocessing.

Given a d-dimensional vector $\boldsymbol{a}$, we compute an embedding inspired by [7] that yields a d^2-dimensional vector by vectorizing the corresponding rank-1 matrix $\boldsymbol{a}\boldsymbol{a}^T$:

$$V(\boldsymbol{a}) = \mathrm{v}ec(\boldsymbol{a}\boldsymbol{a}^T) = \left[a_1^2,\ a_1a_2, \ldots,\ a_1a_d,\ a_2^2,\ a_2a_3, \ldots,\ a_d^2\right], \qquad (5.8)$$

where a_i denotes the i-th element of $\boldsymbol{a}$. Assuming $\boldsymbol{a}$ and $\boldsymbol{b}$ to be unit vectors, the Euclidean distance between the embeddings $V(\boldsymbol{a})$ and $-V(\boldsymbol{b})$ is given by $||V(\boldsymbol{a}) - (-V(\boldsymbol{b}))||^2 =$

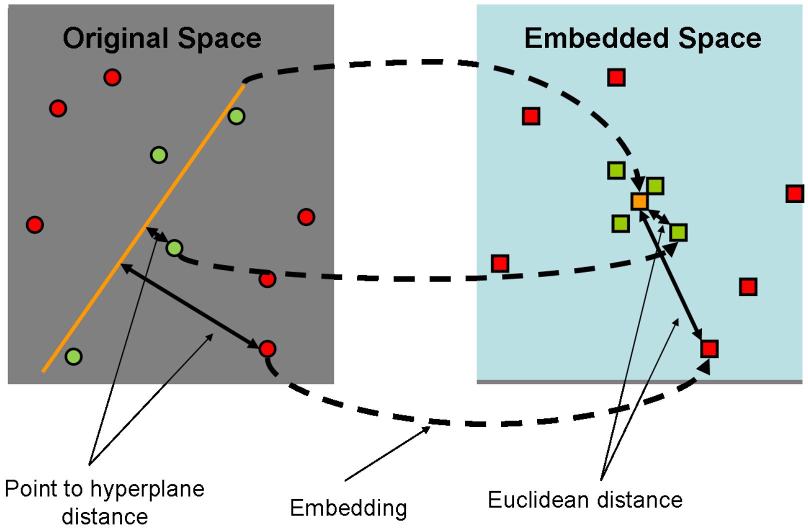


Figure 5.3: The basic intuition behind our EH-Hash solution. We compute an embedding on both points and hyperplanes such that the distance between a point and a hyperplane in the original space is proportional to the Euclidean distance between their embeddings. In the left figure points in green are close to the hyperplane and points in red are far from the hyperplane in terms of the hyperplane to point distance. Correspondingly the points in green on the right are close to the hyperplane in Euclidean distance in the embedded space.

$2 + 2(\boldsymbol{a}^T\boldsymbol{b})^2$. Hence, minimizing the distance between the two embeddings is equivalent to minimizing $|\boldsymbol{a}^T\boldsymbol{b}|$, our intended function.

Given this, we define our **embedding-hyperplane hash** (EH-Hash) function family $\mathcal{E}$ as:

$$h_{\mathcal{E}}(\boldsymbol{z}) = \begin{cases} h_{\boldsymbol{u}}\left(V(\boldsymbol{z})\right), & \text{if } \boldsymbol{z} \text{ is a database point vector,} \\ h_{\boldsymbol{u}}\left(-V(\boldsymbol{z})\right), & \text{if } \boldsymbol{z} \text{ is a query hyperplane vector,} \end{cases}$$

where $h_{\boldsymbol{u}}(\boldsymbol{z}) = \text{sign}(\boldsymbol{u}^T\boldsymbol{z})$ is a one-bit hash function parameterized by $\boldsymbol{u} \sim \mathcal{N}(0, I)$.

Claim 5.1.5. *The family of functions $\mathcal{E}$ defined above is* $\Big(r, r(1+\epsilon),\ \frac{1}{\pi}\cos^{-1}\sin^2(\sqrt{r}),\ \frac{1}{\pi}\cos^{-1}\sin^2(\sqrt{r(1+$ *sensitive for $d_\theta(\cdot,\cdot)$, where $r, \epsilon > 0$.*

Proof. Using the result of [39], for any vector $\boldsymbol{w}, \boldsymbol{x} \in \mathbb{R}^d$,

$$\Pr\left[\text{sign}\left(\boldsymbol{u}^T(-V(\boldsymbol{w}))\right) = \text{sign}\left(\boldsymbol{u}^T V(\boldsymbol{x})\right)\right] = 1 - \frac{1}{\pi}\cos^{-1}\left(\frac{-V(\boldsymbol{w})^T V(\boldsymbol{x})}{\|V(\boldsymbol{w})\|\ \|V(\boldsymbol{x})\|}\right), \quad (5.9)$$

where $\boldsymbol{u} \in \mathbb{R}^{d^2}$ is sampled from a standard d^2-variate Gaussian distribution, $\boldsymbol{u} \sim \mathcal{N}(0, I)$. Note that for any unit vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^{d^2}$, $V(\boldsymbol{a})^T V(\boldsymbol{b}) = \text{Tr}(\boldsymbol{a}\boldsymbol{a}^T\boldsymbol{b}\boldsymbol{b}^T) = (\boldsymbol{a}^T\boldsymbol{b})^2 = \cos^2\theta_{\boldsymbol{a},\boldsymbol{b}}$.

Using 5.9 together with the definition of $h_{\mathcal{E}}$ above, given a hyperplane query $\boldsymbol{w}$ and database point $\boldsymbol{x}$ we have:

$$\Pr[h_{\mathcal{E}}(\boldsymbol{w}) = h_{\mathcal{E}}(\boldsymbol{x})] = 1 - \frac{1}{\pi}\cos^{-1}\left(-\cos^2(\theta_{\boldsymbol{x},\boldsymbol{w}})\right) = \cos^{-1}\left(\cos^2(\theta_{\boldsymbol{x},\boldsymbol{w}})\right)/\pi \quad (5.10)$$

Hence, when $(\theta_{\boldsymbol{x},\boldsymbol{w}} - \frac{\pi}{2})^2 \leq r$,

$$\Pr[h_{\mathcal{E}}(\boldsymbol{w}) = h_{\mathcal{E}}(\boldsymbol{x})] \ \geq\ \frac{1}{\pi}\cos^{-1}\sin^2(\sqrt{r}) = p_1, \quad (5.11)$$

Similarly, for any $\epsilon > 0$ such that $\left(\theta_{\boldsymbol{x},\boldsymbol{w}} - \frac{\pi}{2}\right)^2 \geq r(1+\epsilon)$

$$\Pr[h_{\mathcal{E}}(\boldsymbol{w}) = h_{\mathcal{E}}(\boldsymbol{x})] \ \leq\ \frac{1}{\pi}\cos^{-1}\sin^2(\sqrt{r(1+\epsilon)}) = p_2, \quad (5.12)$$

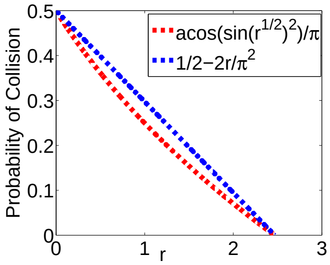


Figure 5.4: Comparison of the probability of collision for our EH-Hash method, $p_1 = \frac{1}{\pi}\cos^{-1}\sin^2(\sqrt{r})$, with that of IH-Hash, $p_1 = \frac{1}{2} - \frac{2r}{\pi^2}$, which is also twice the p_1 for H-Hash. The two functions are very close to each other and therefore the approximation guarantees of EH-Hash are similar to IH-Hash.

$\square$

We observe that this p_1 behaves similarly to $2(\frac{1}{4} - \frac{r}{\pi^2})$. That is, as r varies, EH-Hash's p_1 returns values close to those returned by IH-Hash's p_1 or twice those returned by H-Hash's p_1 (See Figure 5.4)). Hence, the factor $\rho = \frac{\log p_1}{\log p_2}$ improves upon that of the previous section, remaining lower for lower values of ϵ, and leading to better approximation guarantees. Section 5.1.4 provides a more detailed comparison of the two bounds.

On the other hand, EH-Hash's hash functions are significantly more expensive to compute. Specifically, it requires $O(d^2)$ time, whereas H-Hash requires only $O(d)$. To alleviate this problem, we use a form of randomized sampling when computing the hash bits for a query that reduces the time to $O(1/\epsilon'^2)$, for $\epsilon' > 0$. Our method relies on the following lemma, which states that sampling a vector $\boldsymbol{v}$ according to the weights of each element leads to good approximation to $\boldsymbol{v}^T\boldsymbol{y}$ for any vector $\boldsymbol{y}$ (with constant probability). Similar sampling schemes have been used for a variety of matrix approximation problems (see [53]).

Lemma 5.1.6. *Let* $\boldsymbol{v} \in \mathbb{R}^d$ *and define* $p_i = v_i^2/\|\boldsymbol{v}\|^2$. *Construct* $\tilde{\boldsymbol{v}} \in \mathbb{R}^d$ *such that the* i*-th* *element is* v_i *with probability* p_i *and is* 0 *otherwise. Select* t *such elements using sampling with replacement. Then, for any* $\boldsymbol{y} \in \mathbb{R}^d$, $\epsilon > 0$, $c \geq 1$, $t \geq \frac{c}{\epsilon'^2}$,

$$\Pr[|\tilde{\boldsymbol{v}}^T\boldsymbol{y} - \boldsymbol{v}^T\boldsymbol{y}| \leq \epsilon' \|\boldsymbol{v}\|^2 \|\boldsymbol{y}\|^2] > 1 - \frac{1}{c}. \tag{5.13}$$

Proof. Let i_k denote the randomly sampled index (using probability distribution p defined in the lemma) at the k-th round, i.e., i_k is index j with probability p_j. Next, we define a random variable G_k as,

$$G_k = v_{i_k} y_{i_k} / p_{i_k}.$$

Note that,

$$E[G_k] = \sum_j p_j v_j y_j / p_j = \boldsymbol{v}^T\boldsymbol{y}, \tag{5.14}$$

$$Var(G_k) = \sum_j p_j (v_j y_j / p_j)^2 - (\boldsymbol{v}^T\boldsymbol{y})^2 \leq \frac{v_j^2 t_j^2}{t_j^2 / \|\boldsymbol{y}\|^2} = \|\boldsymbol{v}\|^2 \|\boldsymbol{y}\|^2 = 1. \tag{5.15}$$

$$\tag{5.16}$$

Now, our final approximation for $\boldsymbol{v}^T\boldsymbol{y}$ is obtained by averaging random variables G_k, i.e.,

$$\tilde{\boldsymbol{v}}^T\boldsymbol{x} = \frac{1}{t} \sum_k G_k.$$

Now, using Bernstein's inequality:

$$\Pr(|\sum_{k=1}^{t} (G_k - \boldsymbol{v}^T\boldsymbol{y})| \geq t\epsilon) < \exp(-t\epsilon^2).$$

Hence, if we select $t = \frac{c}{\epsilon^2}$, then with probability at least $1 - \log(1/c)$,

$$|\tilde{\boldsymbol{v}}^T\boldsymbol{y} - \boldsymbol{v}^T\boldsymbol{y}| \leq \epsilon.$$

□

The lemma implies that at query time our hash function $h_{\mathcal{E}}(\boldsymbol{w})$ can be computed while incurring a small additive error in time $O(\frac{1}{\epsilon'^2})$, by sampling its embedding $V(\boldsymbol{w})$ accordingly, and then cycling through only the non-zero indices of $V(\boldsymbol{w})$ to compute $\boldsymbol{u}^T(-V(\boldsymbol{w}))$. Note that we can substantially reduce the error in the hash function computation by sampling $O(\frac{1}{\epsilon'^2})$ elements of the vector $\boldsymbol{w}$ and then using v$ec(\boldsymbol{w}\tilde{\boldsymbol{w}}^T)$ as the embedding for $\boldsymbol{w}$. However, in this case, the computational requirements increase to $O(\frac{d}{\epsilon'^2})$.

While one could alternatively use the Johnson-Lindenstrauss (JL) lemma to reduce the dimensionality of the embedding with random projections, doing so has two major difficulties: first, the $d-1$ dimensionality of a subspace represented by a hyperplane implies the random projection dimensionality must still be large for the JL-lemma to hold, and second, the projection dimension is dependent on the sum of the number of database points *and* query hyperplanes. The latter is problematic when fielding an arbitrary number of queries over time or storing a growing database of points—both properties that are intrinsic to our target active learning application. In contrast, our sampling method is instance-dependent and incurs very little overhead for computing the hash function.

Comparison to [7]. Basri et al. define embeddings for finding nearest subspaces [7]. In particular, they define Euclidean embeddings for affine subspace queries and database points, which could be used for NNQH, although they do not specifically apply it to hyperplane-to-point search in their work. Also, their embedding is not tied to LSH bounds in terms of the distance function in Equation 5.2, as we have shown above. Finally, our proposed instance-specific sampling strategy offers a more compact representation with the advantages discussed above.

5.1.4 Comparison of Approximation Guarantees

In this section we compare the bounds on retrieval for both of our hashing methods. To recall, our H-Hash method guarantees the $(1+\epsilon)$-approximate solution in time N^ρ, where $\rho \leq \frac{1-\log(1-\frac{4r}{\pi^2})}{1+\frac{\epsilon}{1+\frac{\pi^2}{4r}\log 4}}$ and $\rho \leq \frac{1-\log(1-\frac{2r}{\pi^2})}{1+\frac{\epsilon}{1+\frac{\pi^2}{2r}\log 2}}$ using the improved hash-bit computation in IH-Hash. Similarly, our EH-Hash method guarantees the $(1+\epsilon)$-approximate solution in time N^ρ, where $\rho \leq \frac{\log\cos^{-1}\sin^2(\sqrt{r})-\log\pi}{\log\cos^{-1}\sin^2(\sqrt{r}(1+\epsilon/2))-\log\pi}$. Note that the function $\cos^{-1}\sin^2(\sqrt{r})$ behaves similarly to $\frac{1}{2}-\frac{2r}{\pi^2}$, which is equal to the probability of collision for IH-Hash and twice the probability of collision for H-Hash method when the points are within distance r (Figure 5.4). This indicates that the bounds for EH-Hash and IH-Hash methods should be similar and significantly stronger than the corresponding bounds for H-Hash.

Figure 5.5 compares the values of ρ obtained for EH-Hash and H-Hash for different values of ϵ. We can clearly see that for our EH-Hash method the value of ρ is always smaller than the corresponding value for H-Hash method. Now, we give a concrete example. Let $\epsilon = 3.5$. Then it can be easily computed that if the closest point to the hyperplane is at angle of around 5^o, then H-Hash will return a point within 9^o in time $N^{0.97}$ while the corresponding bound for EH-Hash method will be $N^{0.89}$, a significant gain.

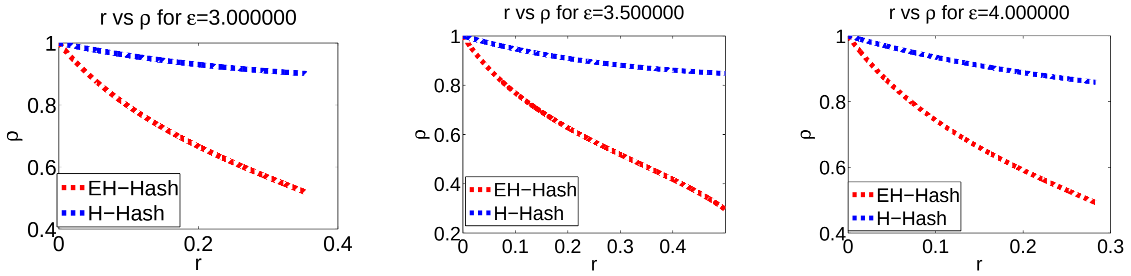


Figure 5.5: Comparison of the values of ρ, which is the parameter that determines the number of examples to search (N^{ρ}), for our H-Hash and EH-Hash methods with different values of $\epsilon = \{3.0, 3.5, 4.0\}$

5.1.5 Recap of the Hashing Approaches

To summarize, I presented two locality-sensitive hashing approaches for the NNQH problem. Our first H-Hash approach defines locality-sensitivity in the context of NNQH, and then provides suitable two-bit hash functions together with a bound on retrieval time. Our second EH-Hash approach consists of a d^2-dimensional Euclidean embedding for vectors of dimension d that in turn reduces NNQH to the Euclidean space nearest neighbor problem, for which efficient search structures (including LSH) are available. While EH-Hash has better bounds than H-Hash, its hash functions are more expensive. To mitigate the expense for high-dimensional data, we use a well-justified heuristic where we randomly sample the given query embedding, reducing the query time to linear in d.

Note that both of our approaches attempt to minimize $d_\theta(\boldsymbol{w}, \boldsymbol{x})$ between the retrieved $\boldsymbol{x}$ and the hyperplane $\boldsymbol{w}$. Since that distance is only dependent on the *angle* between $\boldsymbol{x}$ and $\boldsymbol{w}$, any scaling of the vectors do not effect our methods, and we can safely treat the provided vectors to be unit norm.

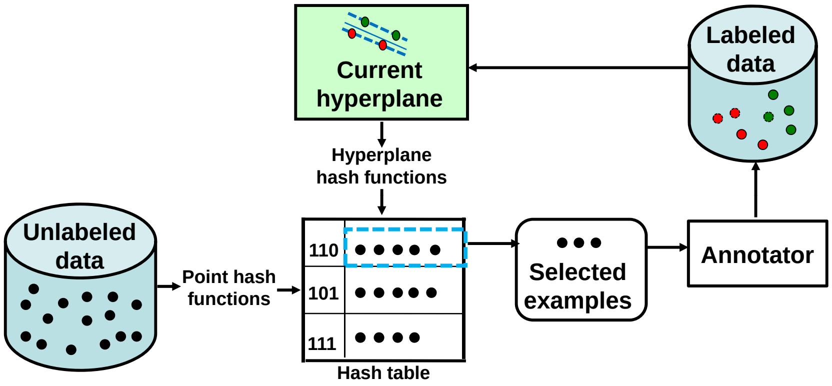


Figure 5.6: Our hashing based solution to run active selection on millions of unlabeled examples efficiently. Offline, we hash unlabeled data into a table using the locality-sensitive hash functions proposed in Section 5.1.5 for the NNQH problem. During the active selection loop we hash the current classifier as the query vector to directly retrieve next examples for labeling. By virtue of our hash function design, these examples are guaranteed to be close to the hyperplane margin, and the retrieval process has a time complexity that is sub-linear in the size of the unlabeled pool.

5.1.6 Application to Large-scale Active Learning

The NNQH problem, which is to obtain database vectors that minimize the distance to a query hyperplane $\boldsymbol{x}^* = \operatorname{argmin}_{\boldsymbol{x}_i \in \mathcal{U}} |\boldsymbol{w}^T \boldsymbol{x}_i|$, directly corresponds to the "simple margin" selection criterion for linear SVM classifiers [98, 85, 15]. Thus, our two NNQH solutions supply exactly the hash functions needed to rapidly identify the next point to label: first we hash the unlabeled database into tables, and then at each active learning loop, we hash the current classifier $\boldsymbol{w}$ as a query. Figure 5.6 provides a flowchart of this procedure for running approximate active selection on large unlabeled pools.

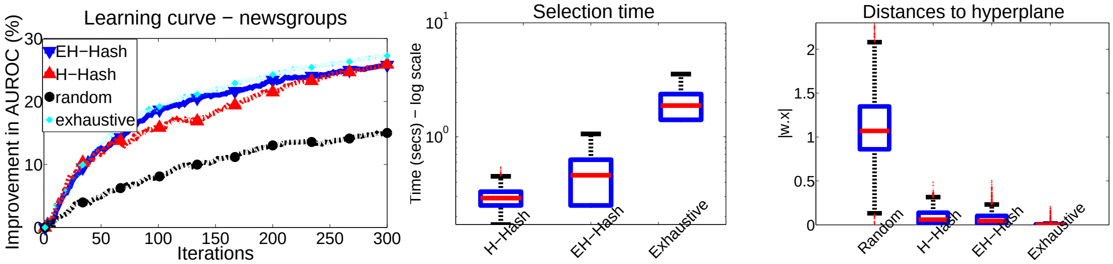
5.2 Results

I demonstrate our approach applied to large-scale active learning tasks. I compare our methods (H-Hash in Section 5.1.2 and EH-Hash in Section 5.1.3) to two baselines: 1) passive learning, where the next label request is randomly selected, and 2) exhaustive active selection, where the margin criterion in Equation 5.1 is computed over all unlabeled examples in order to find the true minimum. The main goal is to show our algorithms can retrieve examples nearly as well as the exhaustive approach, but with substantially greater efficiency.

5.2.1 Datasets and Implementation Details

We use three publicly available datasets.

- The 20 Newsgroups consists of 20,000 documents from 20 newsgroup categories. We use the provided 61,118-dimensional bag-of-words features, and a test set of 7,505.
- The CIFAR-10 [58] dataset consists of 60,000 images from 10 categories. It is a manually labeled subset of the 80 Million Tiny Image dataset [99], which was formed by searching the Web for all English nouns and lacks ground truth labels. We use the provided train and test splits of $50K$ and $10K$ images, respectively.
- The Tiny-1M dataset consists of the first 1,000,000 (unlabeled) images from [99]. For both CIFAR-10 and Tiny-1M, we use the provided 384-dimensional GIST descriptors as features.



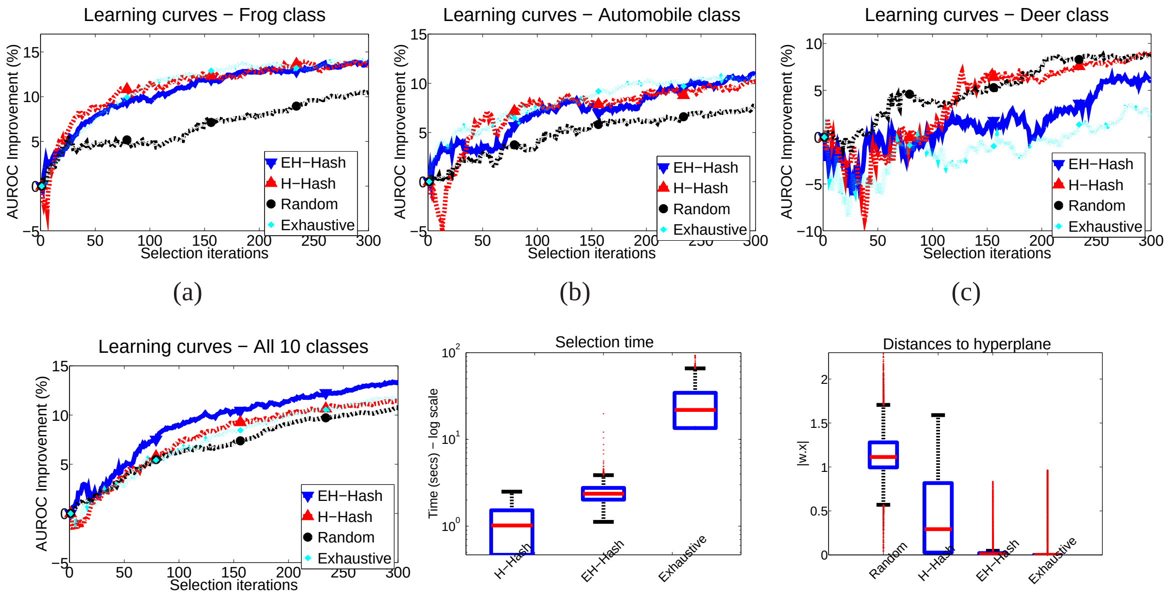
(a) (b) (c)

Figure 5.7: **Newsgroups results.** **(a)** Improvements in prediction accuracy relative to the initial classifier, averaged across all 20 categories and runs. **(b)** Time required to perform selection. **(c)** Value of $|\boldsymbol{w}^T\boldsymbol{x}|$ for the selected examples. Lower is better. Both of our approximate methods (H-Hash and EH-Hash) significantly outperform the passive baseline; they are nearly as accurate as ideal exhaustive active selection, yet require 1-2 orders of magnitude less time to select an example. (Best viewed in color.)

For all datasets, we train a linear SVM in the one-vs-all setting using a randomly selected labeled set (5 examples per class), and then run active selection for 300 iterations. We report results using the area under the ROC curve (AUROC) metric and average results across five such runs. We fix $k = 300$, $N^\rho = 500$, $\epsilon' = 0.01$.

5.2.2 Document Classification: Newsgroups Results

Figure 5.7 shows the results on the 20 Newsgroups, starting with the learning curves for all four approaches (a). The active learners (exact and approximate) have the steepest curves, indicating that they are learning more effectively from the chosen labels compared to the random baseline. Both of our hashing methods perform similarly to the exhaustive selection, yet require scanning an order of magnitude fewer examples (b). Note that Random requires ~ 0 time for selecting a point. Figure 5.7(c) shows the actual values of $|\boldsymbol{w}^T\boldsymbol{x}|$ for the selected examples over all iterations, categories, and runs; in line with our methods' guarantees, they



(a) (b) (c)

(d) (e) (f)

Figure 5.8: **CIFAR-10 results. (a)-(c)** Example learning curves. **(d)-(f)** Plotted as in above figure. Our methods compare very well with the significantly more expensive exhaustive baseline. Our EH-Hash provides more accurate selection than our H-Hash, though requires noticeably more query time.

select points close to those found with exhaustive search. We also observe the expected trade-off: H-Hash is more efficient, while EH-Hash provides better results (only slightly better for this smaller dataset).

5.2.3 Object Recognition: CIFAR-10 Results

Figure 5.8 shows the same set of results on the CIFAR-10. The trends are mostly similar to the above, although the learning task is more difficult on this data, narrowing the margin between active and random. Averaged over all classes, we happen to outperform exhaustive selection (Figure 5.8(d)); this can happen since there is no guarantee that the best active

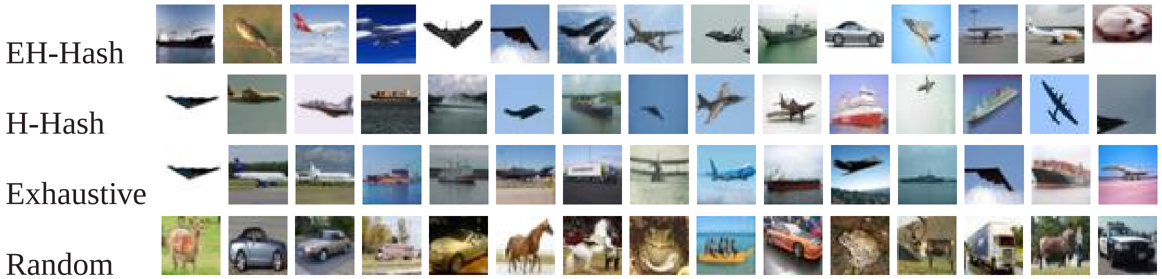


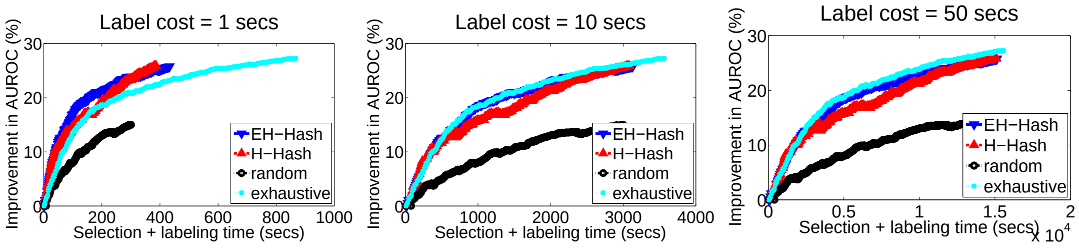
Figure 5.9: First 15 examples selected per method when learning the CIFAR-10 Airplane class.

choice will help test accuracy, and it also reflects the wider variation across per-class results. The boxplots in (f) more directly show the hashing methods are behaving as expected. Both (e) and (f) illustrate their trade-offs: EH-Hash has stronger guarantees than H-Hash (and thus retrieves lower $\boldsymbol{w}^T\boldsymbol{x}$ values), but is more expensive.

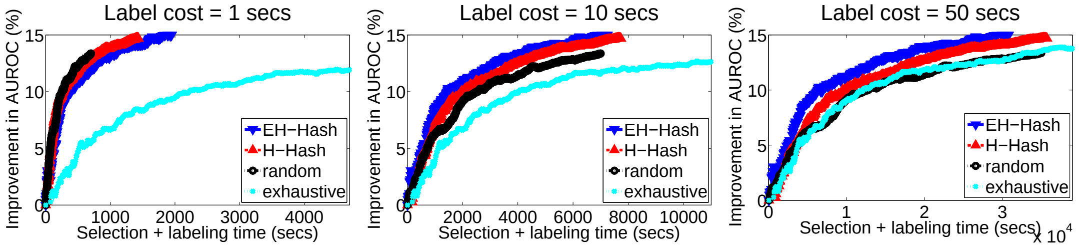
Figure 5.9 shows example image selection results when learning the Airplane class. Both exhaustive search and our hashing methods manage to choose images useful for learning about airplanes/non-airplanes. This shows that we can efficiently obtain relevant training data using our approach for large-scale datasets.

5.2.4 Minimizing both Selection and Labeling Times

Figure 5.10 shows the prediction accuracy plotted against the total time taken per iteration, which includes both *selection* and *labeling* time, for both datasets. This requires knowledge of the time required to label a single example on both datasets (label cost), which is not available for these benchmark datasets. Therefore, we provide results for a range of values for this parameter in order to study the trade-offs involved in minimizing both selection and labeling



(a) Newsgroups dataset



(b) CIFAR-10 dataset

Figure 5.10: Improvements in prediction accuracy as a function of the total time taken, including both selection and labeling time. We provide results for different values of the cost of labeling a single example on both datasets. By minimizing *both* selection and labeling time, our methods provide the best accuracy per unit time.

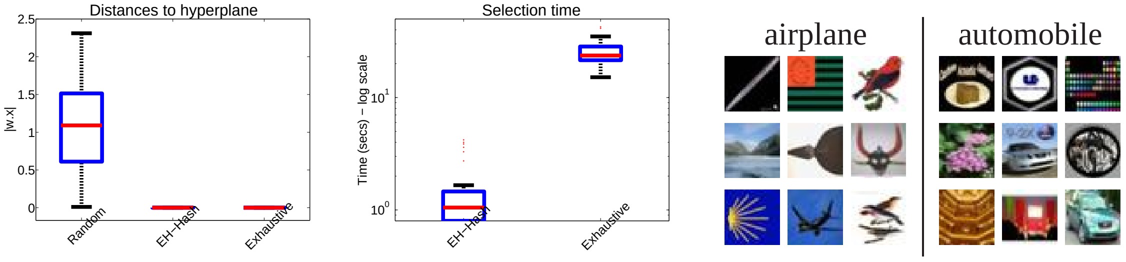
costs. A large value for this parameter indicates that minimizing labeling time is more important and therefore could favor exhaustive active selection. On the other hand, a small value for this parameter indicates that the computational cost of selection is more important and hence could favor fast selection schemes. We show results for representative labeling costs of $\{1, 10, 50\}$ seconds, which are reasonable estimates for annotating documents and images.

As expected, exhaustive selection performs better for larger labeling costs, whereas random selection fares better for lower labeling costs. These results best show the advantage of our approximate methods: accounting for both types of cost inherent to training the classifier, they outperform both exhaustive and random selection in terms of the accuracy gains per unit time for most values on either dataset. While exhaustive active selection suffers because of its large *selection* time, random selection suffers because it wastes expensive *labeling* time on irrelevant examples. Thus, our algorithms provide the best accuracy gains by minimizing both selection and labeling time.

5.2.5 Active Section from 1 Million Images

Finally, to demonstrate the practical capability of our hyperplane hashing approach, we perform active selection on the one million tiny image set. We initialize the classifier with 50 examples from CIFAR-10. The 1M set lacks any labels, making this a "live" test of active learning (we ourselves annotated whatever the methods selected). We use our EH-Hash method, since it offers stronger performance.

Even on this massive collection, our method's selections are very similar in quality to the exhaustive method (see Figure 5.11(a)), yet require orders of magnitude less time (b). The



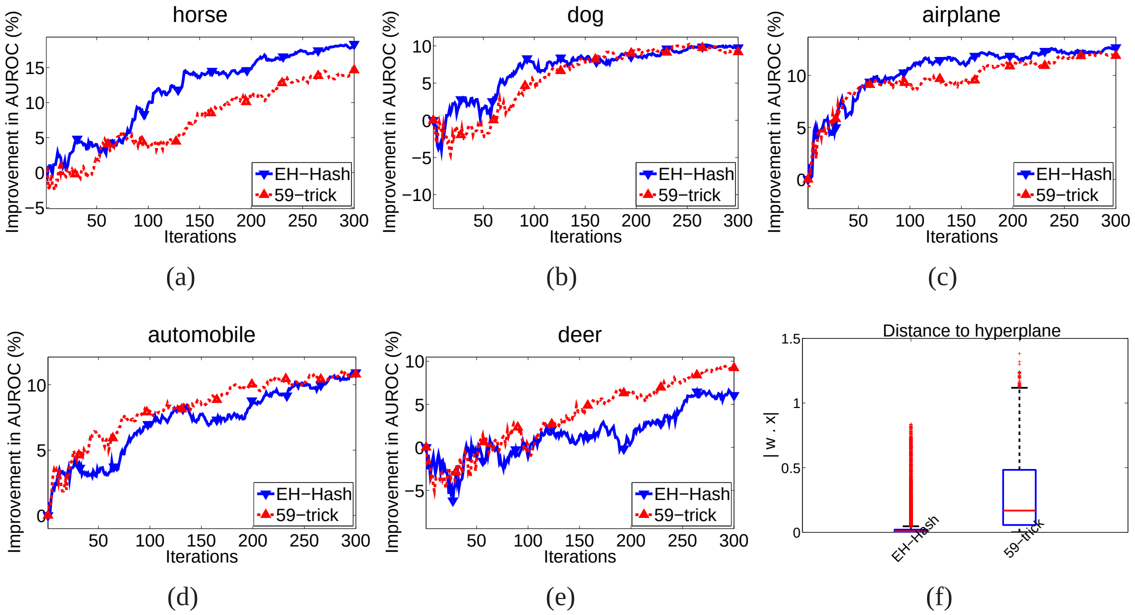
(a) (b) (c)

Figure 5.11: **Tiny-1M results. (a)** Error of examples selected. **(b)** Time required. **(c)** Examples selected by EH-Hash among 1M candidates in the first nine iterations when learning the Airplane and Automobile classes.

images (c) show the selections made from this large pool during the “live” labeling test; among all one million unlabeled examples (nearly all of which likely belong to one of the other 1000s of *classes*) our method retrieves seemingly relevant instances. To our knowledge, this experiment exceeds any previous active selection results in the literature in terms of the scale of the unlabeled pool.

5.2.6 Comparison with the 59-trick Heuristic

So far, we have provided results comparing our approach with exhaustive and random selection, which are the most relevant baselines for our approximate selection schemes. Nonetheless, one could design a simple heuristic to improve upon random selection by considering T randomly selected examples from the unlabeled pool and ranking them based on the active selection criterion. Depending on the size of the subset T, such an approach could be computationally fast and at the same time outperform passive selection since it selects the most informative example from a small subset of the unlabeled pool. In [95], the authors



(a) (b) (c)

(d) (e) (f)

Figure 5.12: Comparison with the '59-trick', an approximate heuristic proposed in [95]. **(a)-(e)** Representative learning curves on CIFAR-10. **(f)** The distances to the hyperplane of the unlabeled examples selected by all approaches. Our approach outperforms the baseline on some categories (a-c) and is worse on others (d-e). However, our approximate method always selects points that are close to the margin (f) which is precisely the guarantee that we provide.

outline such a heuristic and further provide approximation guarantees when the examples are uniformly distributed. In particular, they show that by choosing just $T = 59$ examples, one could obtain an estimate that is with probability 0.95 among the best 5% of the ranking function. We refer to this technique as the '59-trick'.

The weakness of this approximation is that it assumes that the unlabeled pool is uniformly distributed, which is seldom true in realistic applications. Particularly for active learning, one would expect the number of useful examples for a category to be a small fraction of all possible unlabeled data. If not, random selection should perform as well as active learning.

In order to justify this claim, we provide some empirical results comparing the 59-trick.

Figure 5.12 shows representative learning curves obtained on different categories in CIFAR-10 for our approach and 59-trick, for which we randomly choose 59 examples and select the one that is closest to the hyperplane for labeling. For some categories our approach clearly outperforms the heuristic (a-c), whereas it is slightly worse on (d-e). However, our approximate selection scheme always selects points that are close to the margin as seen in Figure 5.12(f). There is no such guarantee for the baseline. This suggests that the efficacy of the baseline approximation would depend on the exact distribution of the set of useful examples, which is not known in general.

5.3 Discussion

In this chapter, I introduced two methods for the NNQH search problem. Both permit efficient large-scale search for points near to a hyperplane, and experiments with three datasets clearly demonstrate the practical value for active learning with massive unlabeled pools. My approach is the first to perform efficient active selection on unlabeled pools in the scale of millions of examples.

In addition, most existing active learning techniques assume that human effort is more scarce and expensive than machine cycles and thus effectively minimize just the number of examples to label or the total annotation time. In contrast, my approach minimizes both *annotation* time, by adopting an active learning strategy, and *selection* time, by using an efficient hashing based approach. Thus when considering the total time expended per iteration my approaches produce the largest improvement in the classifier.

The previous chapters considered active selection functions that computed the expected change in misclassification risk on unlabeled data or in the classifier objective. In contrast, in this chapter, we used an active selection function that depends only on the unlabeled point under consideration. We chose such a function out of necessity because the former functions would require at least quadratic time complexity for performing active selection, which would be impractical for the large-scale datasets that we have considered here. Nevertheless, the selection function used in this chapter has strong theoretical guarantees in terms of the reduction in the size of the version space as shown in [98] and therefore, it can reduce overall manual effort compared to passive learning.

My approach makes two assumptions regarding the data and the classifier which we believe are well justified. First, we assume that the data is normalized, which is the case for most image representations. For example, even for the popular bag of visual words representation it has been shown that normalizing with the L_2-norm results in better generalization than using unnormalized data [103]. Nonetheless, we would like to further explore other hash-functions for our H-hash scheme and data structures that could enable faster selection for the sake of generality. Second, we assume that the classifier is a hyperplane, which is the case for a linear SVM. This is not too limiting as recent research in image classification shows that with a non-linear representation such as sparse coding, even a linear classifier can outperform more expensive kernel based methods [118]. At the same time, it would be interesting to investigate sublinear time methods for kernel based active learning. Recent work on embeddings for the χ^2 RBF kernel [104] and solutions to the kernel LSH problem [60] might provide some insight into extending our hashing schemes for non-linear classifier functions.

In the next chapter, I show that our approximate active learning scheme is suitable for large-scale problems by building a large-scale, autonomous, online learning system for training object detectors.

Chapter 6

A Large-scale System for Autonomous Online Visual Learning

In the previous chapters, I defined and provided solutions to several problems that will enable large-scale transfer of human knowledge for learning visual concepts. In this chapter, I will demonstrate the effectiveness of our solution as a viable protocol for learning visual models, by building the first complete end-to-end system for scalable, automatic online learning of object detectors. We introduce the concept of *live learning* of object detectors where both examples and annotations are autonomously collected using web-based resource, and the learning of the detector proceeds in a live manner.

Object detection is a suitable setting to demonstrate aspects of our solution because (1) it typically requires identifying a single tight-fitting window among thousands of windows within an image, an ideal setting for our large-scale selection approach; (2) state-of-the-art methods for detection typically require large numbers of training examples annotated using bounding boxes, whose expense can be significantly reduced using our active approach; (3) it is an extremely challenging problem where any progress is diligently recorded and encouraged [23, 30, 103, 62, 29].

The system design and large-scale experimental setup I present in this chapter addresses three limiting factors of current methods to train object detectors: first, the vision researcher has already determined the dataset's source and scope, meaning which images will even be considered for labeling is fixed (and possibly biased), for both the training and test sets. Second, active learning methods have only been tested on "sandbox" data for which the true labels are really known, and merely temporarily withheld from the selection algorithm. In fact, nearly all work targets the active *image classification* problem—not detection—and so images in the unlabeled pool are artificially assumed to contain only one prominent object. These common simulations likely inflate the performance of both the active and passive learner, since anything chosen for labeling is relevant. Third, most crowd-sourced collection processes require iterative manual fine-tuning by the algorithm designer (e.g., revising task requirements, pruning responses, barring unreliable Mechanical Turkers) before the data is in usable form. Thus, it is unknown to what extent current approaches could translate to real settings, where the designer of the recognition algorithm is not in the loop.

Rather than fill the data pool with some canned dataset, our system itself gathers possibly relevant images via keyword search (we use Flickr). Keyword-based search is often used for dataset creation [116, 83, 24, 21] followed by manual pruning. In contrast, our system repeatedly surveys the data to identify unlabeled sub-windows that are most uncertain according to the current model, and generates tasks on Mechanical Turk to get the corresponding bounding box annotations without further human involvement. After an annotation budget is spent, we evaluate the resulting detectors both on benchmark data, as well as a novel test set from Flickr. Notably, throughout the procedure *we do not intervene with what goes into the system's data pool, nor the annotation quality from the hundreds of online annotators.*

To make the above a reality requires handling some important technical issues. Active selection for window-based detection is particularly challenging since the object extents (bounding boxes) are unknown in the unlabeled examples; naively one would need to evaluate all possible windows within the image in order to choose the most uncertain. This very quickly leads to a prohibitively large unlabeled pool to evaluate exhaustively. Thus, we introduce a novel part-based detector amenable to linear classifiers, and show how to identify its most uncertain instances in sub-linear time with the hashing-based solution I proposed in Chapter 5.

We show that our detector strikes a good balance between speed and accuracy, with results competitive with and even exceeding the state-of-the-art on the PASCAL VOC, a widely accepted challenging benchmark for object detection. Most importantly, we show successful live learning in an uncontrolled setting. The system learns accurate detectors with much less human effort than strong baselines that rely on human-verified keyword search results.[1]

6.1 Live Learning of Object Detectors

Our goal is to enable online active crowd-sourced object detector training. Given the name of a class of interest, our system will produce a detector to localize novel instances using automatically obtained images and annotations. In order to make this feasible, we first propose a part-based linear SVM detector, and then show how to identify its uncertain examples efficiently using a hashing scheme.

---

[1] The contents of this chapter were published in [111].

6.1.1 Object Representation and Linear Classifier

We first introduce our part-based object representation. Our goal is to design the representation such that a simple linear classifier will be adequate for robust detection. A linear model has many complexity advantages important to our setting: i) SVM training requires time linear in the number of training examples, rather than cubic [51], ii) classification of novel instances requires constant time rather than growing linearly with the number of training examples, iii) exact incremental classifier updates are possible, which makes an iterative active learning loop practical, and iv) my hashing based algorithm (Chapter 5) enables sub-linear time search to map a query *hyperplane* to its nearest points according to a linear kernel.

Inspired by recent findings in sparse coding for image classification [126, 118, 10], we explore a detection model based on sparse coding of local features combined with a max pooling scheme. Previous representations pool coded SIFT features in a single global window or in a fixed class-independent hierarchy of grid cells (i.e., a spatial pyramid structure). While sufficient for whole-image classification, we instead aim to represent an *object* separately from its context, and to exploit its part-based structure with *class-dependent* subwindow pooling.

To this end, we propose an object model consisting of a root window r, multiple part windows $\{p_1, \ldots, p_P\}$ that overlap the root, and context windows $\{c_1, \ldots, c_C\}$ surrounding it. See Figure 6.1. Let $O = [r, p_1, \ldots, p_P, c_1, \ldots, c_C]$ denote a candidate object configuration within an image, and let $\varphi(W)$ denote the sparse feature encoding for local image descriptors extracted from a given subwindow W (to be defined below). The detector scores a candidate

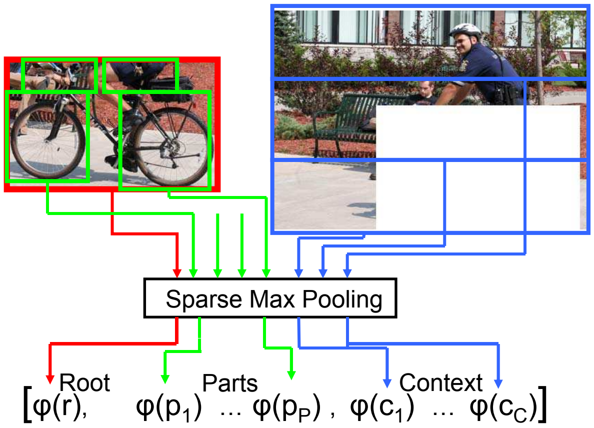


Figure 6.1: Proposed part-based object representation.

configuration as a simple linear sum:

$$
\begin{aligned}
f(O) &= \boldsymbol{w}^T \varphi(O) \qquad (6.1) \\
&= \boldsymbol{w}_r \varphi(r) + \sum_{i=1}^{P} \boldsymbol{w}_{p_i} \varphi(p_i) + \sum_{i=1}^{C} \boldsymbol{w}_{c_i} \varphi(c_i),
\end{aligned}
$$

where $\boldsymbol{w}$ denotes the learned classifier weights, which we obtain with SVM training. We next flesh out the window descriptions; Section 6.1.2 explains how we obtain candidate root placements.

6.1.1.1 Window Descriptions

Given a novel test image, we first extract local image descriptors; we use a dense multi-scale sampling of SIFT in our implementation. Each window type (r, p_i, or c_j) uses these features to create its encoding $\varphi(\cdot)$. The *root window* provides a global summary of the object appearance, and is invariant to spatial translations of features within it.

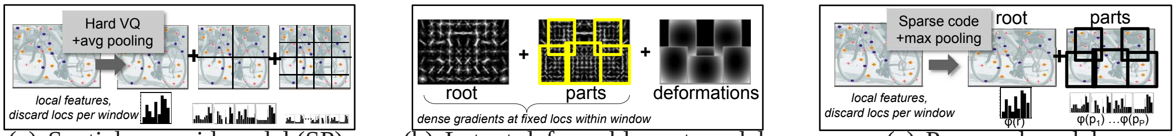
Similarly, each *part window* summarizes the local features within it, discarding their positions; however, the location of each part is defined relative to the current root, and depends on the object class under consideration (i.e., bicycles and cars each have a different configuration of the p_i windows). Thus, they capture the locations and spatial configurations of the most important parts of the object. We train with the part locations and bounds output by the detector in [30]; alternatively, they could be requested once directly from annotators.

The *context windows* incorporate contextual cues surrounding the object, such as the presence of "sky", "ground", "road", etc., and also help discard poorer candidate windows that cover only parts of objects (in which case object features spill into the context window). We create the context windows using a 3×1 partition of r's complement, as shown in the top right of Figure 6.1. We find that providing this context strengthens certain categories, which agrees with recent findings [102].

6.1.1.2 Feature Encoding

Each window is represented using a nonlinear feature encoding based on sparse coding and max-pooling, which we refer to as Sparse Max Pooling (SMP). The SMP representation is related to the well-known bag-of-features (BoF); however, unlike BoF, each component local descriptor is first encoded as a combination of *multiple* visual words, and the weights are then pooled within a region of interest using the max function.

Offline, we cluster a large corpus of randomly selected features to obtain a dictionary of $|V|$ visual words: $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{|V|}]$, where each column $\boldsymbol{v}_i \in \Re^{128}$ is a cluster center in SIFT space. For any window W (whether root/part/context), let $F = \{\boldsymbol{f}_i\}_{i=1}^{|F|}$ be the set of local features falling within it, where each $\boldsymbol{f}_i \in \Re^{128}$ is a SIFT descriptor. We represent this



(a) Spatial pyramid model (SP)
(b) Latent deformable part model (LSVM)
(c) Proposed model

Figure 6.2: Sketch to illustrate contrasts with related existing models. See text for details.

window with a sparse $|V|$-dimensional vector, as follows.

First, each feature $\boldsymbol{f}_i$ is quantized into a $|V|$-dimensional sparse vector $\boldsymbol{s}_i$ that approximates $\boldsymbol{f}_i$ using some existing sparse coding algorithm and the dictionary $\boldsymbol{V}$, that is, $\boldsymbol{f}_i \approx \boldsymbol{s}_i \boldsymbol{V}$. Taking this encoding for every $\boldsymbol{f}_i$ as input, the SMP representation of W is given by:

$$\begin{aligned} \varphi(W) &= [\,\varphi^1, \ldots, \varphi^{|V|}\,], \text{where} \\ \varphi^j &= \max\left(\boldsymbol{s}_i(j)\right), i = 1, \ldots, |F|, \end{aligned} \tag{6.2}$$

for $j = 1, \ldots, |V|$, and $\boldsymbol{s}_i(j)$ is the j-th dimension of the sparse vector encoding the i-th original feature, $\boldsymbol{f}_i$. Finally, we normalize $\varphi(W)$ by its L_2 norm.[2]

The rationale behind the SMP window encoding is twofold: the sparse coding gives a fuller representation of the original features by reflecting nearness to multiple dictionary elements (as opposed to BoF's usual hard vector quantization), while the max pooling gives better discriminability amidst high-variance clutter [10]. See [10, 126] for useful comparisons between various sparse coding approaches, which shows their clear advantage when combined with a linear kernel as compared to the popular BoF.

---

[2] We use Locality-constrained Linear Coding (LLC) [118] to obtain the sparse coding, though other algorithms could also be used for this step.

6.1.1.3 Relationship to Existing Detection Models

Our model intentionally strikes a balance between two recent state-of-the-art detection models: i) a nonlinear SVM with a spatial pyramid (SP) in which each grid cell is a histogram of unordered visual words [103], and ii) a latent SVM (LSVM) with a root+deformable part model in which each part is a rigid histogram of ordered oriented gradients [30]. See Figure 6.2.

On the one hand, the SP model is robust to spatial translations of local features within each grid cell. On the other hand, its nonlinear kernels (required for good performance [103]) makes the classifier quite expensive to train and test, and rigid class-independent bins may fail to capture the structure of the best parts on an object (see Figure 6.2(a)). In contrast, the LSVM model can robustly capture the parts, since it learns multiple part filters that deform relative to the root. However, its dynamic programming step to compute parts' alignment makes it expensive to train. Furthermore, its use of the spatially dense gradient histograms for both the root and parts make them less tolerant to *internal* shifts and rotations (see Figure 6.2(b)).

Our model attempts to incorporate positive aspects of the above two models, while maintaining a much lower computational cost. In particular, we have class-specific part configurations, like [30], but they are fixed relative to the root, like [103]. Our SMP-based encoding is robust to shifts within the part and object windows, thereby tolerating some deformation to the exact part placement without needing the additional DP alignment step during detection. In short, by utilizing a part-based representation and a linear classifier, our approach provides a very good trade-off in terms of model complexity and accuracy.

6.1.2 Generating Candidate Root Windows

So far we have defined a representation and scoring function for any candidate window. Now we discuss how to generate the candidates, whether in novel test images or unlabeled images the system is considering for annotation. A thorough but prohibitively expensive method would be the standard *sliding window* approach; instead, we use a grid-based variant of the *jumping window* method of [20, 113].

The jumping window approach generates candidate windows in a Hough-like projection using visual word matches, and prioritizes these candidates according to a measure of how discriminative a given word and coarse location is for the object class (see Figure 6.3). First, each root window in the training images is divided into an $N \times M$ grid. Let $W_{loc}(r)$ denote a root window's position and scale. Given a training window r and a visual word $\boldsymbol{v}$ occurring at grid position $g \in \{1, \ldots, NM\}$, we record the triplet $(\boldsymbol{v}, g, W_{loc}(r))$. We build a lookup table indexed by the $\boldsymbol{v}$ entries for all training examples. Then, given a test image, for each occurrence of a visual word $\boldsymbol{v}$, we use the lookup table to retrieve all possible $W_{loc}(r)$'s, and project a bounding box in the test image relative to that $\boldsymbol{v}$'s position. Note, candidates can vary in aspect ratio and scale.

The grid cell component g in each triple is used to assign a priority score to each candidate, since we may not want to examine all possible candidates mapped from the lookup table. Specifically, we score a given pair $(\boldsymbol{v}, g)$ based on how predictive it is for the true object bounding box across the training set: $P(\boldsymbol{v}, g)$ is the fraction of the occurrences of word $\boldsymbol{v}$ that appear at grid location g. This function gives a higher score to bounding boxes where the visual word occurs consistently across positive training examples at a particular position

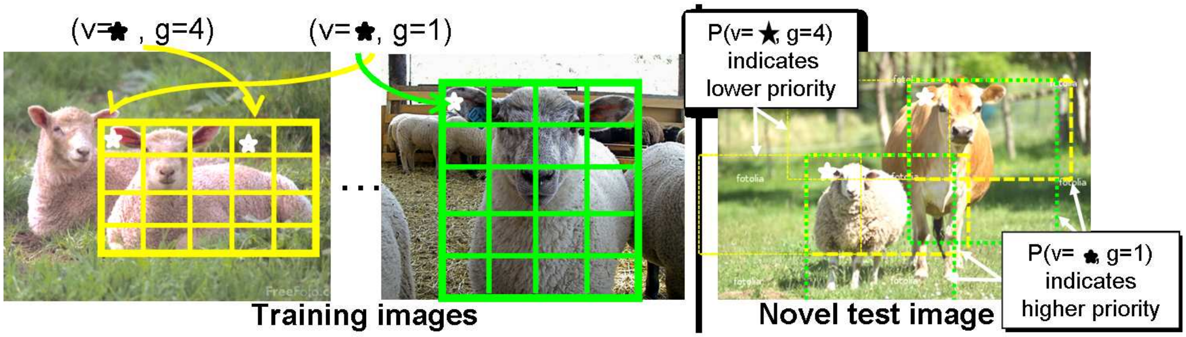

(v=★ , g=4)
(v=★, g=1)
P(v=★, g=4) indicates lower priority
P(v=★, g=1) indicates higher priority
Training images
Novel test image

Figure 6.3: Illustration of jumping window root candidates. Grid cells serve to refine the priority given to each box (but do not affect its placement).

(see Figure 6.3).

Given a test image, we take the top K candidate jumping windows based on their priority scores. The detector is run only on these boxes. In experiments, we obtain 95% recall on most categories when taking just $K = 3,000$ candidates per test image. The same level of recall would require up to 10^5 bounding boxes if using sliding windows (see [103]).

6.1.3 Active Selection of Object Windows

We initialize the online active learning system with a linear SVM trained with a small number of labeled examples for the object. Then, it crawls for a pool of potentially relevant unlabeled data using keyword search with the object name (i.e., it downloads a set of images tagged ‘dog’ when learning to detect dogs). We want to efficiently determine which images among those retrieved should be labeled next by the human annotators. As an active learning criterion, we use the “simple margin” selection method for SVMs [98], a widely used criterion that seeks points that most reduce the version space. This is the same criterion I dealt with in Chapter 5. Given an SVM with hyperplane normal $\boldsymbol{w}$ and an unlabeled pool of data $\mathcal{U}_O = \{\varphi(O_1), \ldots, \varphi(O_n)\}$, the point that minimizes the distance to the current decision boundary is selected for labeling: $O^* = \operatorname{argmin}_{O_i \in \mathcal{U}_O} |\boldsymbol{w}^T \varphi(O_i)|$.

As discussed in Chapter 5, a naive application of this criterion entails computing the classifier response on all unlabeled data, ranking them by $|\boldsymbol{w}^T \varphi(O_i)|$. However, even with a linear classifier, exhaustively evaluating all unlabeled examples at each iteration is prohibitively expensive. Whereas previous active learning work is generally unconcerned about the amount of time it actually takes to compute the next labeling request, it becomes a real issue when working out of the sandbox, since we have live annotators awaiting the next labeling jobs and massive unlabeled data pools. In particular, since we need to apply the active selection function at the level of the *object*, not the entire *image*, we have an enormous number of instances—all bounding boxes within the unlabeled image data. Even using jumping windows, thousands of images yield millions of candidates. Thus, a simple linear scan of all unlabeled data is infeasible.

Therefore, we use our *hyperplane-hashing* algorithm defined in Chapter 5 to identify the most promising candidate windows in sub-linear time. Recall that the algorithm maps inputs to binary keys using a randomized hash function that is locality-sensitive for the angle between the hyperplane normal and a database point. Given a “query hyperplane”, one can hash directly to those points that are nearest to the hyperplane, with high probability.

Formally, let $\mathcal{U}_I$ denote the set of unlabeled images, and $\mathcal{U}_O$ denote the pool of candidate object windows obtained using the jumping window predictor on $\mathcal{U}_I$. Note that $|\mathcal{U}_O| = K \times |\mathcal{U}_I|$. The locality-sensitive hash family $\mathcal{H}$ generates randomized functions with two-bit outputs:

$$h_{\mathcal{H}}(\boldsymbol{z}) = \begin{cases} h_{\boldsymbol{u},\boldsymbol{v}}(\varphi(O_i), \varphi(O_i)), & \text{if } \boldsymbol{z} \text{ is a database vector,} \\ h_{\boldsymbol{u},\boldsymbol{v}}(\boldsymbol{w}, -\boldsymbol{w}), & \text{if } \boldsymbol{z} \text{ is a query hyperplane}, \end{cases}$$

where the component function is defined as

$$h_{\boldsymbol{u},\boldsymbol{v}}(\boldsymbol{a}, \boldsymbol{b}) = [\text{sign}(\boldsymbol{u}^T\boldsymbol{a}), \text{sign}(\boldsymbol{v}^T\boldsymbol{b})], \tag{6.3}$$

sign($\boldsymbol{u}^T\boldsymbol{a}$) returns 1 if $\boldsymbol{u}^T\boldsymbol{a} \geq 0$, and 0 otherwise, and $\boldsymbol{u}$ and $\boldsymbol{v}$ are sampled from a standard multivariate Gaussian, $\boldsymbol{u}$, $\boldsymbol{v} \sim \mathcal{N}(0, I)$. These functions guarantee high probability of collision for a query hyperplane and the points nearest to its boundary. The two-bit hash limits the retrieved points' deviation from the perpendicular by constraining the angle with respect to both $\boldsymbol{w}$ and $-\boldsymbol{w}$.

We use these functions to hash the crawled data into the table.[3] Then, at each iteration of the active learning loop, we hash the current classifier as a query, and directly retrieve examples closest to its decision boundary. We search only those examples, i.e., we compute $|\boldsymbol{w}^T\varphi(O_i)| = |f(O_i)|$ for each one, and rank them in order of increasing value. Finally, the system issues a label request for the top T images under this ranking. Since we only need to evaluate the classifier for examples that fall into a particular hash bucket—typically less than 0.1% of the total number of unlabeled examples—this strategy combined with our new detector makes online selection from large datasets feasible.

---

[3] Hyperplane hashes can be used with existing approximate near-neighbor search algorithms; we use Charikar's formulation, which guarantees the probability with which the nearest neighbor will be returned.

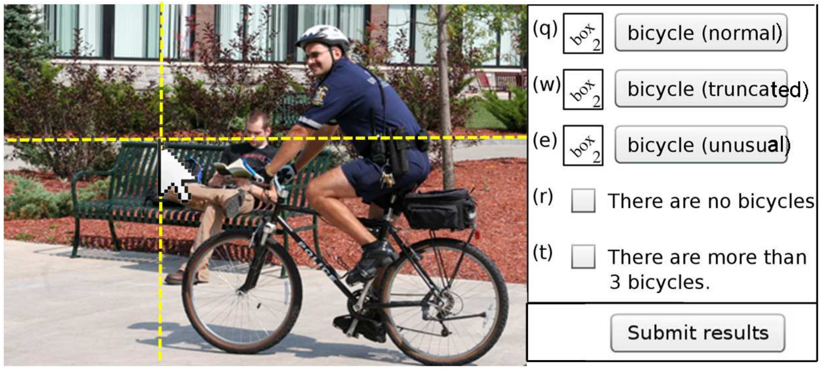

(q) box 2 bicycle (normal)
(w) box 2 bicycle (truncated)
(e) box 2 bicycle (unusual)
(r) ☐ There are no bicycles
(t) ☐ There are more than 3 bicycles.
Submit results

Figure 6.4: MTurk interface to obtain bboxes on actively selected examples.

6.1.4 Online Annotation Requests

To automatically obtain annotations on the actively selected examples, our system posts jobs on Mechanical Turk, where it can pay anonymous workers to provide labels. The system gathers the images containing the most uncertain bounding boxes, and the annotators are instructed to use a rectangle-drawing tool to outline the object of interest with a bounding box (or else to report that none is present). We ask annotators to further subdivide instances into “normal”, “truncated”, or “unusual”, consistent with PASCAL annotations, and to flag images containing more than 3 instances. Figure 6.4 shows the annotation interface.

While MTurk provides easy access to a large number of annotators, the quality of their labels varies. Thus, we design a simple but effective approach to account for the variability. We issue each request to 10 unique annotators, and then cluster their bounding boxes using mean shift to obtain a consensus. We keep only those clusters with boxes from more than half of the annotators. Finally, we obtain a single representative box from each cluster by selecting the one with the largest mean overlap with the rest.

6.1.5 Training the Detector

Training our detector entails learning the weights in Equation 6.1; we use a linear SVM trained to discriminate between windows that do and do not contain the object of interest. All regions that do not overlap the target object are potential negatives; to limit the number used to train, we mine for “hard” negatives, a common approach that converges to the optimal SVM classifier [30]. At each active iteration, we use the updated classifier to extract candidate windows from only the newly obtained training images. We then add the 10 top-scoring windows as negatives if they overlap the target class by less than 20%.

We can now actively train an object detector automatically using minimal crowd-sourced human effort. To recap, the main loop consists of using the current classifier to generate candidate jumping windows, storing all candidates in a hash table, querying the hash table using the hyperplane classifier, giving the actively selected examples to online annotators, taking their responses as new ground truth labeled data, and updating the classifier. See Figure 6.5 for a summary of the complete system.

6.2 Results

The goal of our experiments is three-fold. First, we compare the proposed detector to the most closely related state-of-the-art techniques. Second, we validate our large-scale active selection approach with benchmark data. Third, we deploy our complete live learning system with crawled images, and compare to strong baselines that request labels for the keyword search images in a random sequence.

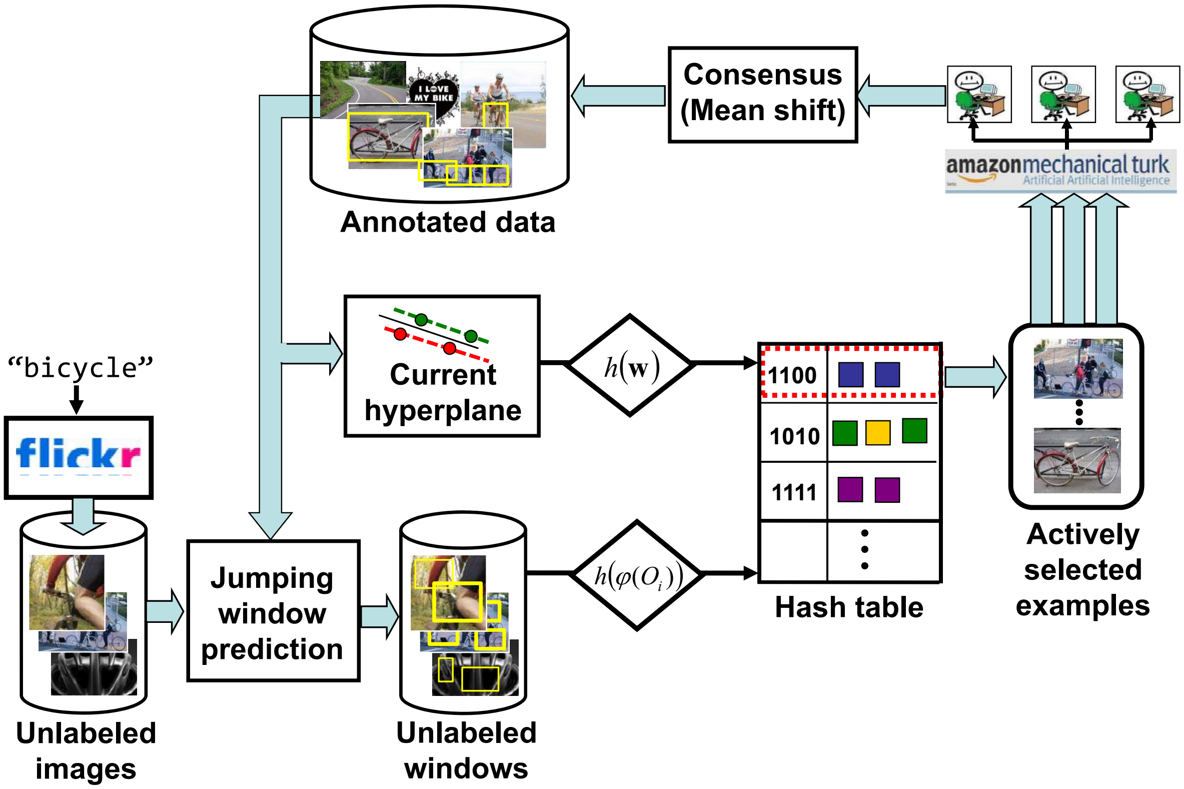


Figure 6.5: Summary of our system for live learning of object detectors. The system autonomously gathers unlabeled examples by querying web-based image collections. We then generate candidate object windows using our jumping window scheme and store all unlabeled windows in a large hash-table using our hyperplane hashing approach. During active selection we use the current classifier as a query to directly retrieve the most uncertain windows and autonomously post them for labeling on Mechanical Turk.

	bird	boat	chair	dog	pottedplant	sheep
Flickr-crawled	2936	3138	2764	1831	1566	1570
Flickr-test	655	628	419	780	364	820

Table 6.1: Number of images in the crawled data and the new Flickr test set.

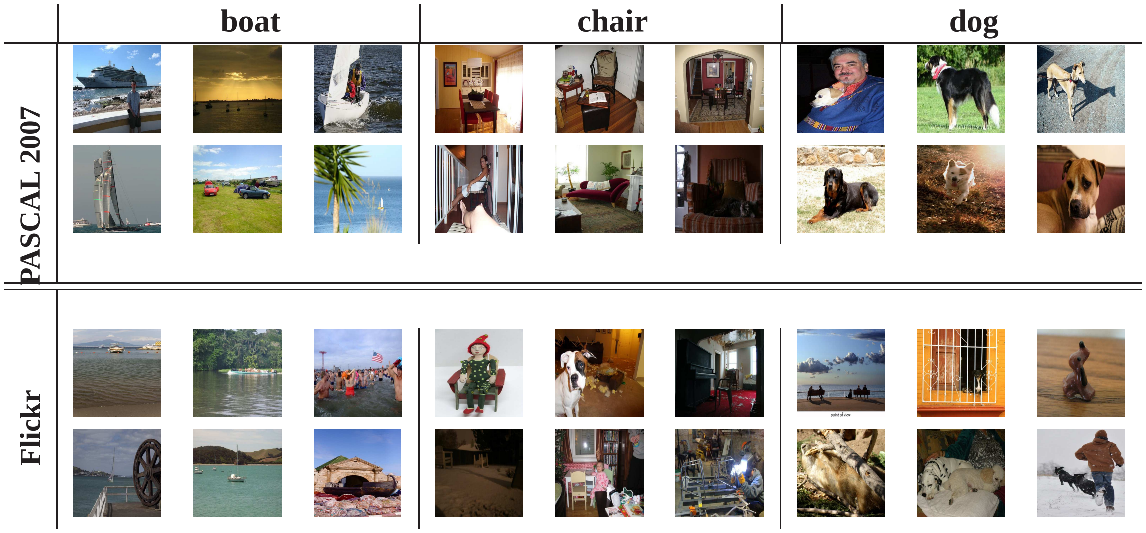


Figure 6.6: Some randomly chosen example images from PASCAL 2007 testset (top) and Flickr testset (bottom) for the categories *boat, chair, dog*.

6.2.1 Datasets and Implementation details

We use two datasets: the PASCAL VOC 2007, and a new Flickr dataset (details below).

- The PASCAL VOC 2007 dataset contains about 5000 training and an equal number of test images from 20 classes selected from the broad categories: animals, vehicles and indoor objects. Since the dataset was downloaded from Flickr and manually pruned the images contain a large number of objects in varying scales, locations, viewpoints and significant background clutter. Figure 6.6(top) shows some random images from the dataset.
- The new Flickr dataset contains about 3000 images per class for six of the most difficult categories in the PASCAL dataset. To form the Flickr test set, we download images tagged with the class names dated in 2010 and for training images, our sys-

tem is restricted to images dated in 2009. The Flickr test set was not manually pruned and therefore it should contain a more general sample of images available on the web than the PASCAL test set. Annotations on the Flickr test set were obtained using the same interface used for the live learning process (see Section 6.1.4). Table 6.1 provides some data statistics. Figure 6.6(bottom) shows some randomly chosen examples from the Flickr test set.

We use dense SIFT at three scales (16, 24, 32 pixels) with grid spacing of 4 pixels, for 30,000 features per image. We obtain $|V| = 56,894$ visual words with two levels of hierarchical k-means on a sample of training images. We use the fast linear SVM code `svm_perf` [51], with $C = 100$. For the LLC coding, we use code by [118], setting k, the number of non-zero values in the sparse vector $\boldsymbol{s}_i$ to 5, following [118]. We use $P = 6$ parts per object from each of a 2-mixture detector from [30], take $T = 100$ instances per active cycle, and set $N, M = 4$. We fix $N^{\rho} = 500$ and $\epsilon' = 0.01$ for the hash table as reported in the previous chapter. During detection we perform non-max suppression on top ranked boxes and select 10 per image.

6.2.2 Comparison to State-of-the-Art Detectors

First we compare our detector to the algorithms with the current best performance on VOC 2007 benchmark of 20 objects, as well as our own implementation of two other relevant baselines. All methods are trained and tested with the same PASCAL-defined splits. We report accuracies using the PASCAL benchmark metric, which computes the average precision of detected instances where an instance is said to be detected if its overlap score (intersection/union) with the ground truth is ≥ 0.5. Average precision is the average of precision

	classif	parts	feats	cands	aero.	bicyc.	bird	boat	bottl	bus	car	cat	chair	cow	dinin.	dog	horse	motor.	person	potte.	sheep	sofa	train	tvmon.	Mean
Ours	linear	yes	single	jump	**48.4**	48.3	14.1	13.6	15.3	43.9	49.0	**30.7**	11.6	30.3	13.3	**21.8**	43.6	45.0	18.2	11.1	**28.8**	**33.0**	**47.7**	43.0	30.5
BoF SP	linear	no	single	jump	30.4	43.1	6.9	3.5	10.8	35.8	45.0	17.7	11.5	24.6	3.5	18.0	43.5	44.0	15.3	1.5	19.1	14.7	35.7	34.9	23.0
LLC SP	linear	no	single	jump	35.9	46.7	6.4	6.3	16.5	45.6	49.8	26.7	12.5	27.3	6.8	18.2	44.9	45.0	18.2	4.6	23.2	22.6	41.3	42.0	27.0
LSVM+HOG [30]	nonlinear	yes	single	slide	32.8	**56.8**	2.5	**16.8**	**28.5**	39.7	**51.6**	21.3	**17.9**	18.5	**25.9**	8.8	49.2	41.2	**36.8**	**14.6**	16.2	24.4	39.2	39.1	29.1
SP+MKL [103]	nonlinear	no	multiple	jump	37.6	47.8	**15.3**	15.3	21.9	**50.7**	50.6	30.0	17.3	**33.0**	22.5	21.5	**51.2**	**45.5**	23.3	12.4	23.9	28.5	45.3	**48.5**	**32.1**

Table 6.2: Average precision compared to a spatial pyramid BoF baseline (BoF SP), a sparse coding max pooling spatial pyramid baseline modeled after [118] (LLC SP), and two state-of-the-art approaches [30, 103] on the PASCAL VOC, where all methods are trained and tested on the standard benchmark splits.

values computed at a set of uniformly sampled recall levels. It emphasizes ranking relevant detections higher and is therefore appropriate for object detection.

Table 6.2 shows the results. The first three rows all use the same original SIFT features, a linear SVM classifier, and the same jumping windows in the test images. They differ, however, in the feature coding and pooling. The **BoF SP** baseline maps the local features to a standard 3-level spatial pyramid bag-of-words descriptor with L_2-normalization. The **LLC SP** baseline applies sparse coding and max pooling within the spatial pyramid grid cells. LLC SP is the method of [118]; note, however, we are applying it for detection, whereas the authors propose their approach for image classification.

The linear classifier with standard BoF coding is the weakest. The LLC SP baseline performs quite well in comparison, but its restriction to a global spatial pyramid structure does appear to hinder accuracy. In contrast, our detector improves over LLC SP noticeably for most objects (compare rows 1 and 3), likely due to its part windows.

Our detector is competitive with both of the state-of-the-art approaches discussed in Section 6.1.1: **SP+MKL** [103], which uses a cascade of classifiers that culminates with a learned combination of nonlinear SVM kernels over multiple feature types, and **LSVM+HOG** [30],

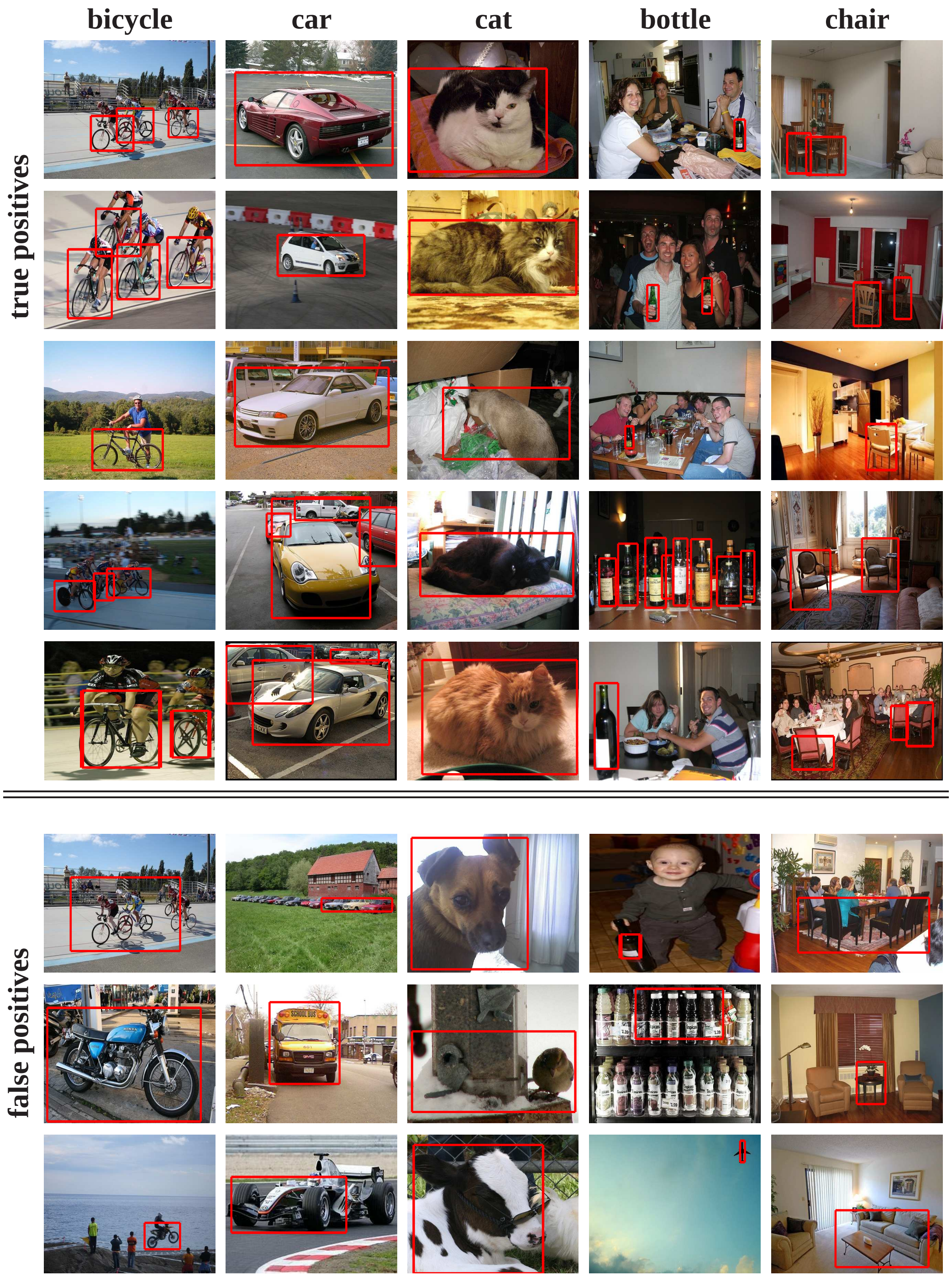


Figure 6.7: Example detections on the PASCAL dataset obtained by our detector for five representative categories (bicycle, car, cat, bottle, chair). Our detector provides accurate localization despite large variations in appearance, pose and number of objects. Top scoring false positives are mainly from similar categories (e.g. cat vs. dog, bicycle vs. motorbike) or due to the presence of a large number of similar objects (row 6 column 1, row 7 column 4) or inaccurate localization.

which uses the latent SVM and deformation models for parts. In fact, our detector outperforms **all existing results** for 6 of the 20 objects, improving the state-of-the-art. At the same time, it is significantly faster to train (about 50 to 600 times faster; see Table 6.5).

The classes where we see most improvements seem to make sense, too: our approach outperforms the rigid spatial pyramid representation used in [103] for cases with more class-specific part structure (aeroplane, bicycle, train), while it outperforms the dense gradient parts used in [30] for the more deformable objects (dog, cat, cow).

This is a very promising result, given our algorithm's major complexity advantages during both training and testing, as well as its reliance on only a single feature type. For comparison, the nonlinear stage of [103] takes 50 seconds on 100 candidates, whereas our linear detector requires only 5 seconds, giving us a speed-up of roughly an order of magnitude at detection. The advantage is greater during training; the LSVM model requires about 4 hours to train [30], while ours requires only $\sim$5-10 minutes.

Figure 6.7 shows some example detections (high-scoring true and false positives) by our detector for five representative categories.

6.2.3 Active Detector Training on PASCAL

We next compare our active selection scheme to a passive learning baseline that randomly selects images for bounding box annotation. We select six representative categories from PASCAL: we take two each from those that are "easier" ($>$40 AP), "medium" (25-40 AP) and "hard" (0-25 AP) according to the state-of-the-art result (max of rows 4 and 5 in Ta-

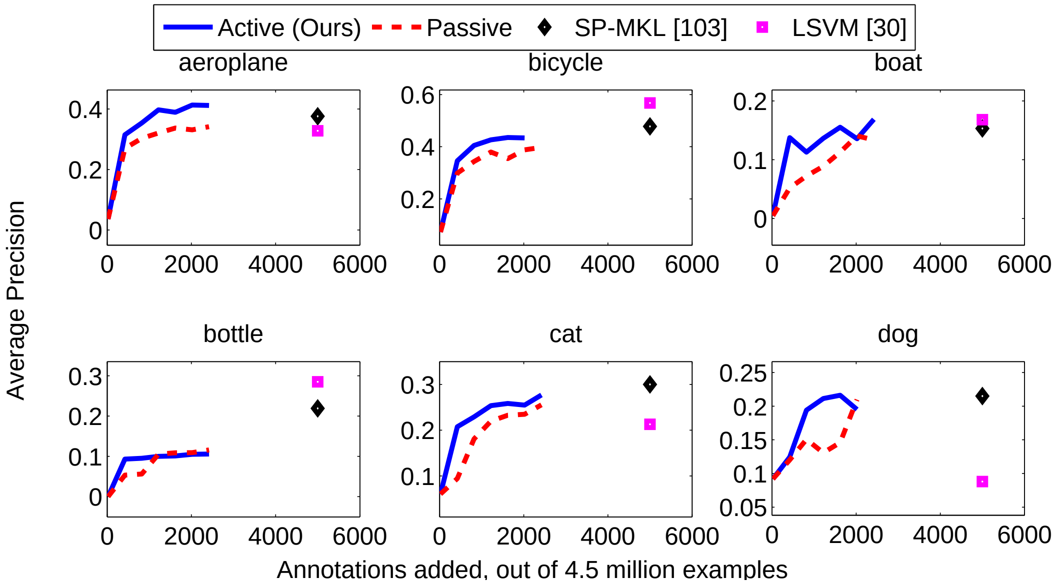


Figure 6.8: Active detector training on PASCAL. Our large-scale active selection yields steeper learning curves than passive selection, and reaches peak state-of-the-art performance using only ∼30% of the data.

ble 6.2).[4] We initialize each object's classifier with 20 examples, and then let the remainder of the training data serve as the unlabeled pool, a total of 4.5 million examples. At each iteration, both methods select 100 examples, add their true bounding boxes (if any) to the labeled data, and retrain. This qualifies as learning in the "sandbox", but is useful to test our jumping window and hashing-based approach. Furthermore, the natural cluttered images are significantly more challenging than data considered by prior active object learning approaches, and our unlabeled pool is orders of magnitude larger.

Figure 6.8 shows the results. We see our method's clear advantage; the steeper learning

---

[4] In spite of our method's efficiency, *evaluating* this experiment on a single category is quite costly: each point on the learning curve requires running the detector on 5,000 test images and re-doing hard negatives.

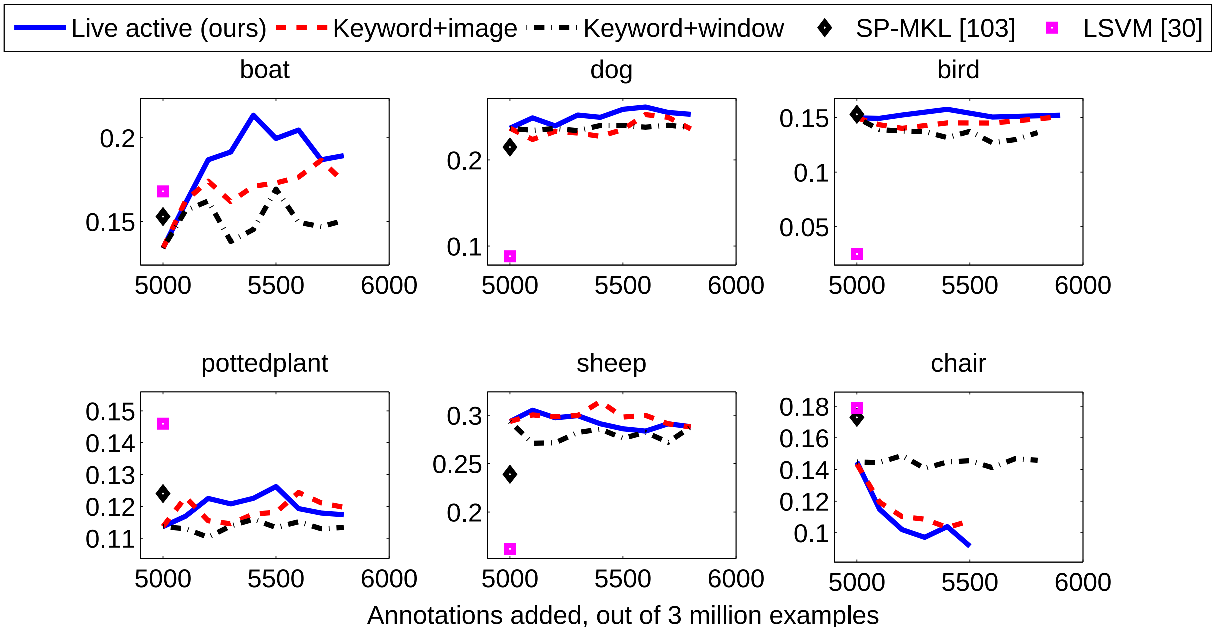


Figure 6.9: Live learning results on PASCAL test set.

curves indicate it improves accuracy on the test set using fewer labels. In fact, in most cases our approach reaches state-of-the-art performance (see markers above 5000 labels) using only one-third of the available training data.

6.2.4 Online Live Learning on Flickr

Finally, we deploy our complete live learning system, where new training data is crawled on Flickr, and apply it to both PASCAL and a new Flickr test set. We consider all object classes for which state-of-the-art AP is less than 25.0 (boat, dog, bird, pottedplant, sheep, chair) in order to provide the most challenging case study, and to seek improvement through live learning where other methods have struggled most. To form the Flickr test set, we download images tagged with the class names dated in 2010; when running live training, our system is restricted to images dated in 2009. See Table 6.1 for the data statistics. Figure 6.6 shows the diversity and difficulty of randomly chosen examples from the Flickr test set in comparison with the PASCAL 2007 testset.

We compare to (1) a **Keyword+image baseline** that uses the same crawled image pool, but randomly selects images to get annotated on MTurk, and (2) a **Keyword+window baseline** that randomly picks jumping windows to get labeled. These are strong baselines since most of the images will contain the relevant object. In fact, *Keyword+image exactly represents the status quo approach*, where one creates a dataset by manually pruning keyword search results. We initialize all methods with the PASCAL-trained models, and run for 10 iterations.

6.2.4.1 Live Learning Applied to PASCAL Test Set

Figure 6.9 shows the results for the PASCAL test set. Note that the learning curves start at $x = 5000$ because the training set consists of the 5000 PASCAL training examples in addition to annotations requested on the Flickr training set. For four of the six categories, our system improves test accuracy, and outperforms the keyword approaches. The final AP also exceeds the current state-of-the-art for three categories (see Table 6.3). This is an important and exciting result, given the size of the unlabeled pools (∼3 million examples), and the fact that the system learned its refined models completely automatically.

However, for two classes (chair, sheep), live learning decreases accuracy. Of course, more data cannot always guarantee improved performance on a fixed test set. We suspect the decline is due to stark differences in the distribution of PASCAL and Flickr images, since the PASCAL dataset creators do some manual preparation and pruning of all PASCAL data. Our

	aeroplane	bird	boat	cat	dog	sheep	sofa	train
Ours	**48.4**	**15.8***	**18.9***	**30.7**	**25.3***	**28.8**	**33.0**	**47.7**
Previous best	37.6	15.3	16.8	30.0	21.5	23.9	28.5	45.3

Table 6.3: Categories for which our method yields the best AP on PASCAL VOC 2007, compared to any result we found in the literature. (*means extra Flickr data automatically obtained by our system was used to train.)

next result seems to confirm this.

In addition, on further examination of the training images obtained for "chair" we noticed that the examples were particularly challenging with heavy occlusion (due to tables, people, etc.) and had several ambiguous annotations on similar categories such as sofa and couch. The unreliability of the automatically collected annotations due to the ambiguous category definition could have affected the accuracy to a fair extent.

6.2.4.2 Live Learning Applied to Flickr Test Set

Figure 6.10 shows the results on the new Flickr test set, where we apply the same live-learned models from above. Again, the x-axis starts at 5000 since we initialize the classifier with PASCAL training examples. The accuracy of the models trained on PASCAL examples (at $x = 5000$) is poor for most categories possibly because this test set is more challenging and diverse than PASCAL(see Figure 6.6). However, the improvements made by our approach with additional Flickr data are dramatic—both in terms of its absolute climb, as well as its margin over the baselines. In all, the results indicate that our large-scale live learning approach can autonomously build models appropriate for detection tasks with realistic and unbiased data.

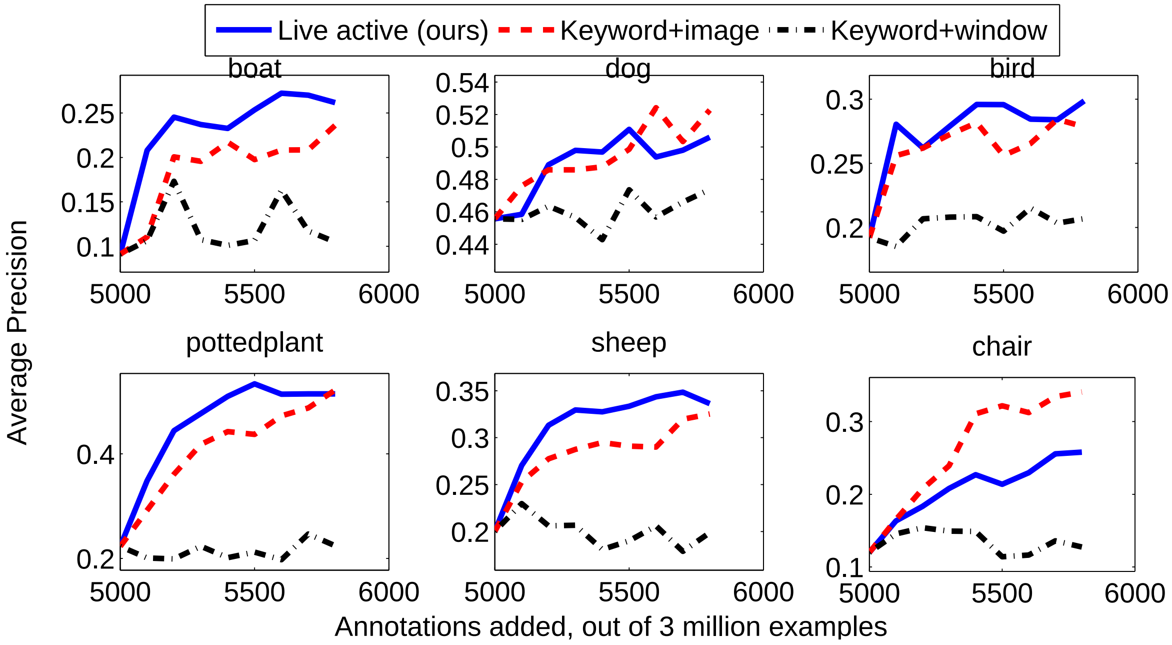


Figure 6.10: Live learning results on Flickr test set.

Figure 6.11 shows selections made by either method when learning “boat”, illustrating how ours focuses human attention among the crawled tagged images. Most of the regions selected by our approach correspond roughly to the category being learned. In addition, the selected images are diverse since there is large variability in the pose, scale and illumination of the object of interest. On the other hand, examples that are randomly selected either do not contain the category of interest or contain instances that are not particularly useful (note the extremely small scale of the boats in images selected by the baseline in Figure 6.11). This shows the importance of obtaining a few useful annotations over a large number of irrelevant examples.

Table 6.4 provides a quantitative measure of the relevance of the selected windows by computing the mean overlap scores of the windows selected for querying by either approach with

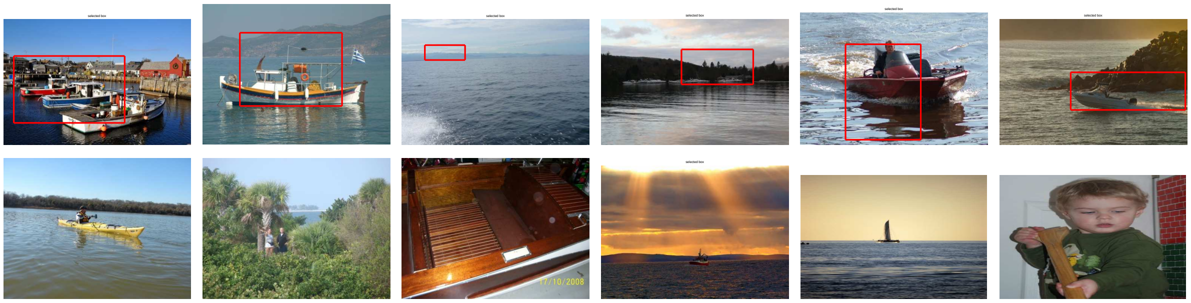


Figure 6.11: Selections by our live approach (top), Keyword+image (bottom).

	bird	boat	chair	dog	pottedplant	sheep
Live active (ours)	38.4	29.5	23.4	43.2	33.2	34.3
Keyword+window	21.8	17.5	12.6	26.7	14.8	18.8

Table 6.4: Mean overlap scores of the windows selected by our approach and the keyword+window baseline with the ground truth bounding box provided by Mechanical Turk annotators. The higher scores for live active (by up to 200% for some categories) indicates that our approach correctly picks the most relevant object region for querying.

the ground truth bounding box provided by the Mechanical Turk annotators. A higher overlap score with the ground truth indicates that the selected windows capture the main region of interest better. Our approach has up to 200% higher values of this score and therefore it ends up picking windows that are highly overlapping with the most relevant object in the image and is thus able to better focus annotator attention on the more relevant images.

6.2.5 Computation Time

Table 6.5 shows the time complexity of various stages of our approach and illustrates our major advantages for active selection and classifier retraining in comparison to the state-of-the-art methods of [30, 103]. Our reported times are based on a dual-core 2.8 GHz CPU, which is comparable to the systems used by [30, 103]. Our jumping window+hashing scheme

	Active selection	Training	Detection per image
Ours + active	10 mins	5 mins	150 secs
Ours + passive	0 mins	5 mins	150 secs
LSVM [30]	3 hours	4 hours	2 secs
SP+MKL[103]	93 hours	$>$ 2 days	67 secs

Table 6.5: Run-time comparisons of different stages of our detector against the passive baseline and other state-of-the-art detectors. Our detection time is mostly spent pooling the sparse codes. Active times are estimated for [30, 103] models based on linear scan. Our approach's efficiency in selecting useful images and retraining the classifier makes live learning practical.

requires on average 2-3 seconds to retrieve 2,000 examples nearest the current hyperplane, and an additional 250 seconds to rank and select 100 images to query. In contrast, a linear scan over the entire unlabeled pool would require about 60 hours.

The entire online learning process requires 45-75 minutes per iteration: 5-10 min. to retrain, 5 min. for selection, and $\sim$1 hour to wait for the MTurk annotations to come back (typically 50 unique MTurkers gave labels per task). Thus, waiting on MTurk responses takes the majority of the time, and could likely be reduced almost arbitrarily with better payment/incentives. In comparison, direct exhaustive active selection with the detector of [30, 103] would require about 8 hours to 1 week, respectively, per iteration.

6.3 Discussion

In summary, our contributions in this chapter are i) a novel efficient part-based linear detector that provides excellent performance, ii) a jumping window and hashing scheme suitable for the proposed detector that retrieves relevant instances among millions of candidates, and iii) the first active learning results for which both image data and annotations are automatically obtained, with minimal involvement from vision experts. Tying all these parts together, I demonstrated an effective end-to-end system for learning object detectors that provides state-of-the-art results on two challenging datasets.

My result is significant for several reasons. First, nearly all active learning work targets the *image classification* problem and so images in the unlabeled pool are artificially assumed to contain only one prominent object. This is partly because of the significant challenge of active selection for window-based detection since the object extents (bounding boxes) are unknown in the unlabeled examples. Therefore, naively one would need to evaluate all possible windows within the image in order to choose the most uncertain. I dealt with this large-scale selection issue by introducing our novel part-based detector amenable to linear classifiers for which the most uncertain instances can be efficiently obtained in sub-linear time using our hashing-based solution proposed in Chapter 5.

Second, unlike existing object recognition results where the vision researcher has already determined which images will even be considered for labeling, our system autonomously obtains both image examples and the most relevant labels through web-based resources without any involvement from an expert. In addition, while most active learning approaches unrealistically simulate the active learning process using datasets that have already been collected and labeled, we are the first to test our active approach in the "live" setting where multiple human annotators directly provide requested annotations. We neither intervened with what went into the data pool nor the annotations returned by the anonymous annotators and yet obtained state-of-the-art results on the challenging PASCAL VOC 2007 dataset. This is a significant step towards our goal of transferring human knowledge with as little human effort as possible.

Chapter 7

Future Work

There are several interesting directions of future work for this thesis in both choosing the right examples and questions to ask and the process of collecting supervision information from humans. I discussed future work for the different components of the thesis at the end of their corresponding chapters. In this chapter I outline further research along the broader theme of this thesis.

A known issue with many active learning approaches is that by always selecting uncertain/informative examples according to the *current hypothesis* they might not "explore" other regions of instance space as effectively. For example, if the concept being learned consists of multiple modes (e.g. corresponding to different viewpoints of an object), an active learning system could end up exploring just one of the viewpoints depending on the initially sampled training set. This could be an important issue when learning hundreds and thousands of object categories found in web images.

It would be interesting to study if active learning schemes in general can provide convergence guarantees with respect to the optimal classifier trained using all labeled data. One could also consider balancing exploiting the current hypothesis with exploring other regions of feature space in order to mitigate this issue. A fairly straightforward approach would be to alternate between rounds of exploration, where examples are randomly drawn, and exploitation, where an active selection criterion is used. Thus, it would be interesting to investigate how to correctly balance exploration versus exploitation so that near optimal classifiers can be trained with minimal effort. Our framework provides the best groundwork for such studies because we can collect large-scale datasets and automatically obtain annotations.

In terms of annotation collection, while a consensus from multiple annotators is an effective approach for obtaining the true answer amidst unreliable annotators, it is rather wasteful in terms of overall effort. Therefore, utilizing multiple annotators cost-effectively to obtain the correct answer with high confidence is an important problem to consider next.

As a possible solution, one could dynamically evaluate the agreement of an annotator with the consensus and target specific annotators based on their performance and speed on the task in future iterations. The number of annotators to use for a particular example could also depend on the difficulty of the example. Our approach for predicting required effort on novel test images provides one way of measuring a task's difficulty. Confusion in the answers of multiple annotators could also mean an example is difficult. One could use such measures in order to automatically choose the number of annotators required for novel examples. This could minimize the number of annotators to use in addition to minimizing the number of training examples.

Thus, further research in this direction could lead to faster and more effective transfer of human knowledge in the future.

Chapter 8

Conclusion

In this thesis I discussed research that enables large-scale transfer of human knowledge while learning visual categories by solving several important problems in active learning.

I first generalized traditional active learning to *cost-sensitive multi-level* active learning where the learner can pose multiple annotation queries and each annotation question costs a variable amount of manual effort. My approach provides a cost-effective solution for how to actively choose not only which instance to label, but also what type of image annotation to acquire. I have shown that compared to traditional active learning which restricts supervision to yes/no questions, a richer means of providing supervision and a method to effectively select supervision based on both information gain and cost to the supervisor is better-suited for building classifiers with minimal human intervention.

My method is general enough to accept other types of annotations or classifiers, as long as the cost and risk functions can be appropriately defined. While we have concentrated mostly in the domain of object recognition, the problem of comparing different types annotations in a unified framework is potentially applicable to several other domains both in vision and machine learning such as video annotation, tracking, or document classification.

My work on this problem opens up several interesting directions. Annotators have variable capabilities and speeds depending on the specific task and image content. Therefore, one could extend the approach to target specific annotators and build user-specific cost functions. This would require designing the VOI criterion to choose not only what annotation type and image looks most promising, but also which user ought to be responsible for annotating it. Allowing further levels of supervision, such as scene layout, contextual cues, or part labels, would further enable us to improve the way in which human supervisors can interact with computer vision systems. It would be interesting to pursue probabilistic models that can integrate such diverse annotation cues.

The above approach provides a solution for choosing a *single* annotation from a large pool of multiple types of annotation queries which is well-suited for the case where a single human annotator is available to interact with the system. However, it might be preferable to farm out a *batch* of good queries at once when one has access to multiple distributed annotators simultaneously.

Towards this end, I considered the problem of *budgeted batch active learning* where at each iteration the active learner must select a batch of examples meeting a given budget of supervision that can be parallelly annotated using multiple simultaneous annotators. The budget is determined by the funds (or time) available to spend on annotation. Solutions to the problem can directly utilize the multiple annotators that are available through crowd-sourcing services such as Mechanical Turk.

I formulated the budgeted selection task as a continuous optimization problem where we determine which subset of possible queries should maximize the improvement to the classifier's objective, without overspending the budget. I provided an efficient alternating minimization procedure in order to find the local optimum of the objective function. Our results indicate that budgeted batch selection is crucial for efficient active learning in practical scenarios, clearly outperforming conventional myopic selection and batch techniques. This is because unlike the baseline techniques, our approach considers how much the classifier objective changes if one were to obtain the most optimistic labels on the selected examples and is therefore able to utilize large budgets most effectively.

However, like most existing active selection methods our batch selection approach has a time complexity that is at least quadratic in the size of the unlabeled pool. This could make it impractical for really large unprepared unlabeled data available on the web.

In order to handle such large-scale selection problems, we considered the problem of *sub-linear time active learning*, where one needs to retrieve the database points that are most informative to a classifier in time that is sub-linear in the number of unlabeled examples, i.e., without having to exhaustively scan the entire unlabeled pool. Towards this end, we introduced two solutions for the nearest neighbor to a query hyperplane (NNQH) search problem. The solutions permit efficient large-scale active learning using the widely used simple margin criterion for linear SVM classifiers on millions of examples. Our experiments with three datasets clearly demonstrated the practical value for active learning with massive unlabeled pools.

Our work opens up several interesting directions including exploring more accurate hash-functions for our H-hash scheme, other data structures that could enable faster selection. It would interesting to see if further improvements can be made on our IH-Hash scheme or investigate if there are hard upper bounds on the amount of time one can save using such approximate hashing schemes. It would also be interesting to consider similar approaches for performing sublinear time selection when using non-linear kernel based active learning. Recent work on solutions to the kernel LSH problem [60] might provide some insight in this regard.

Finally, tying all these together I proposed the first approach for *live-learning* of object detectors using web-based image collections and crowd-sourcing services. Instead of manually collecting, pruning and annotating training datasets, my system itself gathers possibly relevant images via keyword search and repeatedly identifies unlabeled sub-windows that are most uncertain according to the current model, and generates tasks on Mechanical Turk to get the corresponding bounding box annotations without any involvement from the algorithm designer. Using the system we were able to learn object detectors for several classes and improved on the state-of-the-art on the challenging PASCAL VOC dataset.

The significance of our improvements are further enhanced by the fact that we neither intervened with what was added to the training set nor fine-tuned the annotations returned by annotators on Mechanical Turk. This is an important step towards our goal of effectively transferring human knowledge with as little human effort as possible.

In summary, my thesis work aids in developing vision systems that continuously improve their knowledge of the world by learning to ask the right kind of questions to a human supervisor in the most cost-effective way. My work has fundamentally expanded the way in which visual and other learning systems can obtain information from humans and has opened up several interesting problems in this sub-field.

Bibliography

[1] A. Andoni and P. Indyk. Near-Optimal Hashing Algorithms for Near Neighbor Problem in High Dimensions. In *Proceedings Symposium on Foundations of Comp. Sci.*, 2006.

[2] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support Vector Machines for Multiple-Instance Learning. In *Advances in Neural Information Processing Systems*, 2002.

[3] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Fast kernel learning using sequential minimal optimization. Technical Report UCB/CSD-04-1307, Feb 2004.

[4] J. Baldridge and M. Osborne. Active Learning and Logarithmic Opinion Pools for Hpsg Parse Selection. *Nat. Lang. Eng.*, 14(2):191–222, 2008.

[5] K. Barnard, P. Duygulu, N. de Freitas, D. Forsyth, D. Blei, and M. Jordan. Matching Words and Pictures. *Journal of Machine Learning Research*, 3:1107–1135, 2003.

[6] E. Bart and S. Ullman. Cross-Generalization: Learning Novel Classes from a Single Example by Feature Replacement. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[7] R. Basri, T. Hassner, and L. Zelnik-Manor. Approximate Nearest Subspace Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[8] T. Berg, A. Berg, J. Edwards, and D. Forsyth. Who's in the picture? In *Advances in Neural Information Processing Systems*, 2004.

[9] T. Berg and D. Forsyth. Animals on the Web. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[10] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning Mid-level Features for Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[11] Y. Y. Boykov and M. P. Jolly. Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in n-d Images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, 2001.

[12] K. Brinker. Incorporating Diversity in Active Learning with Support Vector Machines. In *Proceedings of International Conference on Machine Learning*, 2003.

[13] R. Bunescu and R. Mooney. Multiple Instance Learning for Sparse Positive Bags. In *Proceedings of International Conference on Machine Learning*, 2007.

[14] Caltech 101 Image Database. *http://www.vision.caltech.edu/ImageDatasets/Caltech101/*, 2004.

[15] C. Campbell, N. Cristianini, and A. Smola. Query Learning with Large Margin Classifiers. In *Proceedings of International Conference on Machine Learning*, 2000.

[16] P. Carbonetto, G. Dorko, C. Schmid, H. Kuck, and N. de Freitas. *Toward Category-Level Object Recognition*, chapter A Semi-supervised Learning Approach to Object

Recognition with Spatial Integration of Local Features and Segmentation Cues. Springer Berlin, 2006.

[17] G. Cauwenberghs and T. Poggio. Incremental and Decremental Support Vector Machine Learning. In *Advances in Neural Information Processing Systems*, 2000.

[18] E. Chang, S. Tong, K. Goh, and C.-W. Chang. Support Vector Machine Concept-Dependent Active Learning for Image Retrieval. *IEEE Transactions on Multimedia*, 2005.

[19] M. Charikar. Similarity Estimation Techniques from Rounding Algorithms. In *ACM Symp. on Theory of Computing*, 2002.

[20] O. Chum and A. Zisserman. An Exemplar Model for Learning Object Classes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[21] B. Collins, J. Deng, K. Li, and L. Fei-Fei. Towards scalable dataset construction: An active learning approach. In *Proceedings of European Conference on Computer Vision*, 2008.

[22] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models-their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, 1995.

[23] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 886–893, 2005.

[24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[25] T. Dietterich, R. Lathrop, and T. Lozano-Perez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.

[26] D. Dueck and B. Frey. Non-metric Affinity Propagation for Unsupervised Image Categorization. In *Proceedings of the International Conference On Computer Vision (ICCV)*, 2007.

[27] P. Duygulu, K. Barnard, N. de Freitas, and D. Forsyth. Object Recognition as Machine Translation: Learning a Lexicon for a Fixed Image Vocabulary. In *Proceedings of European Conference on Computer Vision*, 2002.

[28] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[29] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes Challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[30] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object Detection with Discriminatively Trained Part Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(1), 2009.

[31] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning Object Categories from Google's Image Search. In *Proceedings of the International Conference On Computer Vision (ICCV)*, 2005.

[32] R. Fergus, P. Perona, and A. Zisserman. Object Class Recognition by Unsupervised Scale-Invariant Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003.

[33] R. Fergus, P. Perona, and A. Zisserman. A Visual Category Filter for Google Images. In *Proceedings of European Conference on Computer Vision*, 2004.

[34] J. Freidman, J. Bentley, and A. Finkel. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Transactions on Mathematical Software*, 3(3):209–226, September 1977.

[35] Y. Freund, H. Seung, E. Shamir, and Tishby. Selective Sampling Using the Query by Committee Algorithm. *Machine Learning*, 28, 1997.

[36] C. Galleguillos, B. Babenko, A. Rabinovich, and S. Belongie. Weakly Supervised Object Localization with Stable Segmentations. In *ECCV '08: Proceedings of the 10th European Conference on Computer Vision*, pages 193–207, Berlin, Heidelberg, 2008. Springer-Verlag.

[37] T. Gartner, P. Flach, A. Kowalczyk, and A. Smola. Multi-Instance Kernels. In *Proceedings of International Conference on Machine Learning*, 2002.

[38] A. Gionis, P. Indyk, and R. Motwani. Similarity Search in High Dimensions via Hashing. In *Proceedings of the 25th Intl Conf. on Very Large Data Bases*, 1999.

[39] M. Goemans and D. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *Journal of the Association for Computing Machinery*, 42(6):1115–1145, 1995.

[40] K. Grauman and T. Darrell. Unsupervised Learning of Categories from Sets of Partially Matching Image Features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[41] R. Greiner, A. J. Grove, and D. Roth. Learning Cost-Sensitive Active Classifiers. *Artif. Intell.*, 139(2):137–174, 2002.

[42] Y. Guo and D. Schuurmans. Discriminative Batch Mode Active Learning. In *Advances in Neural Information Processing Systems*, 2007.

[43] S. Gupta, J. Kim, K. Grauman, and R. Mooney. Watch, listen & learn: textscCo-Training on captioned images and videos. In *ECML/PKDD '08: Proceedings of the 2008 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 457–472, 2008.

[44] S. Hacker and L. von Ahn. Matchin: Eliciting User Preferences with an Online Game. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 55–64. ACM Press, 2009.

[45] S. Hoi, R. Jin, J. Zhu, and M. Lyu. Semi-supervised SVM Batch Mode Active Learning with Applications to Image Retrieval. *ACM Transactions on Information Systems*, 1(1), 2009.

[46] R. Hwa. Sample Selection for Statistical Parsing. *Computational Linguistics*, 2004.

[47] P. Indyk and N. Thaper. Fast Image Retrieval via Embeddings. In *Intl Wkshp on Stat. and Comp. Theories of Vision*, 2003.

[48] P. Jain and A. Kapoor. Active Learning for Large Multi-class Problems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[49] P. Jain, S. Vijayanarasimhan, and K. Grauman. Hashing Hyperplane Queries to Near Points with Applications to Large-Scale Active Learning. In *Advances in Neural Information Processing Systems*, 2010.

[50] T. Joachims. Transductive Inference for Text Classification using Support Vector Machines. In *Proceedings of International Conference on Machine Learning*, 1999.

[51] T. Joachims. Training Linear SVMs in Linear Time. In *Proceedings of International Conference on Knowledge Discovery and Data Mining*, 2006.

[52] A. Joshi, F. Porikli, and N. Papanikolopoulos. Multi-Class Active Learning for Image Classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[53] R. Kannan and S. Vempala. Spectral Algorithms. *Foundations and Trends in Theoretical Computer Science*, 4(3-4):157–288, 2009.

[54] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Active Learning with Gaussian Processes for Object Categorization. In *Proceedings of the International Conference On Computer Vision (ICCV)*, 2007.

[55] A. Kapoor, E. Horvitz, and S. Basu. Selective Supervision: Guiding Supervised Learning with Decision-Theoretic Active Learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[56] G. Kim, C. Faloutsos, and M. Hebert. Unsupervised Modeling of Object Categories Using Link Analysis Techniques. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[57] A. Krause and C. Guestrin. Nonmyopic Active Learning of Gaussian Processes: an Exploration-Exploitation Approach. In *Proceedings of International Conference on Machine Learning*, 2007.

[58] A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, 2009.

[59] B. Kulis and K. Grauman. Kernelized Locality-Sensitive Hashing for Scalable Image Search. In *Proceedings of the International Conference On Computer Vision (ICCV)*, 2009.

[60] B. Kulis and K. Grauman. Kernelized Locality-sensitive Hashing for Scalable Image Search. In *Proceedings of the International Conference On Computer Vision (ICCV)*, 2009.

[61] J. T. Kwok and P. Cheung. Marginalized multi-instance kernels. In *Proceedings of International Joint Conferences on Artificial Intelligence*, 2007.

[62] C. Lampert, M. Blaschko, and T. Hofmann. Beyond Sliding Windows: Object Localization by Efficient Subwindow Search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[63] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the Kernel Matrix with Semidefinite Programming. *J. Mach. Learn. Res.*, 5:27–72, 2004.

[64] I. Laptev and T. Lindeberg. Space-time Interest Points. In *Proceedings of the International Conference On Computer Vision (ICCV)*, 2003.

[65] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning Realistic Human Actions from Movies. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[66] Y. Lee and K. Grauman. Foreground Focus: Finding Meaningful Features in Unlabeled Images. In *Proceedings of the British Machine Vision Conference*, 2008.

[67] B. Leibe, A. Leonardis, and B. Schiele. Combined Object Categorization and Segmentation with an Implicit Shape Model. In *Workshop on Statistical Learning in Computer Vision*, 2004.

[68] A. Levin, P. Viola, and Y. Freund. Unsupervised Improvement of Visual Detectors using Cotraining. In *Proceedings of the International Conference On Computer Vision (ICCV)*, 2003.

[69] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994.

[70] L. Li, G. Wang, and L. Fei-Fei. Optimol: automatic online picture collection via incremental model learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[71] M. Lindenbaum, S. Markovitch, and D. Rusakov. Selective Sampling for Nearest Neighbor Classifiers. *Machine Learning*, 54(2), 2004.

[72] A. Magen. Dimensionality Reductions that Preserve Volumes and Distance to Affine Spaces, and their Algorithmic Applications. In *Randomization and Approximation Techniques in Computer Science*, 2002.

[73] O. Maron and A. Ratan. Multiple-Instance Learning for Natural Scene Classification. In *Proceedings of International Conference on Machine Learning*, 1998.

[74] E. N. Mortensen and W. A. Barrett. Intelligent Scissors for Image Composition. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 1995.

[75] N. Panda, K. Goh, and E. Chang. Active Learning in Very Large Image Databases. *Journal of Multimedia Tools and Applications: Special Issue on Computer Vision Meets Databases*, 31(3), December 2006.

[76] J. Platt. *Advances in Large Margin Classifiers*, chapter Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. MIT Press, 1999.

[77] G. Qi, X. Hua, Y. Rui, J. Tang, and H. Zhang. Two-Dimensional Active Learning for Image Classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[78] P. Quelhas, F. Monay, J.-M. Odobez, D. Gatica-Perez, T. Tuytelaars, and L. VanGool. Modeling Scenes with Local Descriptors and Latent Aspects. In *Proceedings of the International Conference On Computer Vision (ICCV)*, Beijing, China, October 2005.

[79] S. Ray and M. Craven. Supervised versus Multiple Instance Learning: An Empirical Comparison. In *Proceedings of International Conference on Machine Learning*, 2005.

[80] K. S. J. C. P. M. Robbie Haertel, Eric Ringger. Assessing the Costs of Sampling Methods in Active Learning for Annotation. In *Proceedings of Workshop on Parsing German*, 2008.

[81] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive Foreground Extraction using Iterated Graph Cuts. In *ACM Transactions on Graphics*, 2004.

[82] B. Russell, A. Efros, J. Sivic, W. Freeman, and A. Zisserman. Using Multiple Segmentations to Discover Objects and their Extent in Image Collections. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[83] B. Russell, A. Torralba, K. Murphy, and W. Freeman. Labelme: a Database and Web-Based Tool for Image Annotation. Technical report, MIT, 2005.

[84] R. Salakhutdinov and G. Hinton. Semantic Hashing. In *Proceedings of the SIGIR Workshop on Information Retrieval and Applications of Graphical Models*, 2007.

[85] G. Schohn and D. Cohn. Less is More: Active Learning with Support Vector Machines. In *Proceedings of International Conference on Machine Learning*, 2000.

[86] F. Schroff, A. Criminisi, and A. Zisserman. Harvesting Image Databases from the Web. In *Proceedings of the International Conference On Computer Vision (ICCV)*, 2007.

[87] B. Settles. Active learning literature survey. Technical Report 1648, University of Wisconsin–Madison, 2009.

[88] B. Settles, M. Craven, and L. Friedland. Active Learning with Real Annotation Costs. In *NIPS Wkshp on Cost-Sensitive Learning*, 2008.

[89] B. Settles, M. Craven, and S. Ray. Multiple-Instance Active Learning. In *Advances in Neural Information Processing Systems*, 2008.

[90] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, 1992.

[91] G. Shakhnarovich, P. Viola, and T. Darrell. Fast Pose Estimation with Parameter-Sensitive Hashing. In *Proceedings of the International Conference On Computer Vision (ICCV)*, 2003.

[92] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In

Proceedings of the International Conference On Computer Vision (ICCV), pages 1–15, 2006.

[93] V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the Point Cloud: from Transductive to Semi-supervised Learning. In *Proceedings of International Conference on Machine Learning*, 2005.

[94] J. Sivic, B. Russell, A. Efros, A. Zisserman, and W. Freeman. Discovering Object Categories in Image Collections. In *Proceedings of the International Conference On Computer Vision (ICCV)*, Beijing, China, October 2005.

[95] A. J. Smola and B. Schlkopf. Sparse Greedy Matrix Approximation for Machine Learning. In *Proceedings of the International Conference On Computer Vision (ICCV)*, pages 911–918, 2000.

[96] A. Sorokin and D. Forsyth. Utility data annotation with amazon mechanical turk. In *CVPR Workshops*, 2008.

[97] M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, pages 945–952. MIT Press, 2002.

[98] S. Tong and D. Koller. Support Vector Machine Active Learning with Applications to Text Classification. In *Proceedings of International Conference on Machine Learning*, 2000.

[99] A. Torralba, R. Fergus, and W. T. Freeman. 80 million Tiny Images: a Large Dataset for Non-Parametric Object and Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, 2008.

[100] M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–590, Hawai, June 1992.

[101] J. Uhlmann. Satisfying General Proximity / Similarity Queries with Metric Trees. *Information Processing Letters*, 40:175–179, 1991.

[102] J. Uijlings and A. Smeulders. What is the Spatial Extent of an Object? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[103] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple Kernels for Object Detection. In *Proceedings of the International Conference On Computer Vision (ICCV)*, 2009.

[104] A. Vedaldi and A. Zisserman. Efficient Additive Kernels via Explicit Feature Maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[105] J. Verbeek and B. Triggs. Scene Segmentation with CRFs Learned from Partially Labeled Images. In *Advances in Neural Information Processing Systems*, 2007.

[106] S. Vijayanarasimhan and K. Grauman. Keywords to Visual Categories: Multiple-Instance Learning for Weakly Supervised Object Categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[107] S. Vijayanarasimhan and K. Grauman. Multi-Level Active Prediction of Useful Image Annotations for Recognition. In *Advances in Neural Information Processing Systems*, 2008.

[108] S. Vijayanarasimhan and K. Grauman. What's It Going to Cost You?: Predicting Effort vs. Informativeness for Multi-Label Image Annotations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[109] S. Vijayanarasimhan and K. Grauman. Top-down Pairwise Potentials for Piecing Together Multi-class Segmentation Puzzles. In *IEEE workshop on Perceptual Organization in Computer Vision 2010*, 2010.

[110] S. Vijayanarasimhan and K. Grauman. Cost-sensitive Active Visual Category Learning. *International Journal of Computer Vision*, 91(1):24–45, 2011.

[111] S. Vijayanarasimhan and K. Grauman. Large-Scale Live Active Learning: Training Object Detectors with Crawled Data and Crowds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[112] S. Vijayanarasimhan, P. Jain, and K. Grauman. Far-Sighted Active Learning on a Budget for Image and Video Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[113] S. Vijayanarasimhan and A. Kapoor. Visual Recognition and Detection Under Bounded Computational Resources. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[114] P. Viola and M. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[115] P. Viola, J. Platt, and C. Zhang. Multiple Instance Boosting for Object Detection. In *Advances in Neural Information Processing Systems*, 2005.

[116] L. von Ahn and L. Dabbish. Labeling Images with a Computer Game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2004.

[117] L. von Ahn, R. Liu, and M. Blum. Peekaboom: A game for locating objects in images. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2006.

[118] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-Constrained Linear Coding for Image Classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[119] M. Weber, M. Welling, and P. Perona. Unsupervised Learning of Models for Recognition. In *Proceedings of European Conference on Computer Vision*, 2000.

[120] Y. Weiss, A. Torralba, and R. Fergus. Spectral Hashing. In *Advances in Neural Information Processing Systems*, 2008.

[121] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *Proceedings of the International Conference On Computer Vision (ICCV)*, 2005.

[122] T.-F. Wu, C.-J. Lin, and R. C. Weng. Probability Estimates for Multi-Class Classification by Pairwise Coupling. *Journal of Machine Learning Research*, 2004.

[123] R. Yan, J. Yang, and A. Hauptmann. Automatically Labeling Video Data using Multi-Class Active Learning. In *Proceedings of the International Conference On Computer Vision (ICCV)*, 2003.

[124] R. Yan, J. Yang, and A. Hauptmann. Automatically Labeling Video Data using Multi-Class Active Learning. In *Proceedings of the International Conference On Computer Vision (ICCV)*, 2003.

[125] C. Yang and T. Lozano-Perez. Image database retrieval with multiple-instance learning techniques. In *Proceedings of IEEE International Conference on Data Engineering*, 2000.

[126] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear Spatial Pyramid Matching Sparse Coding for Image Classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[127] T. Yeh, K. Tollmar, and T. Darrell. Searching the Web with Mobile Images for Location Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[128] A. Yuille, D. Cohen, and P. Hallinan. Feature Extraction from Faces using Deformable Templates. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1989.

[129] Z.-J. Zha, X.-S. Hua, T. Mei, J. Wang, G.-J. Qi, and Z. Wang. Joint Multi-label Multi-instance Learning for Image Classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[130] C. Zhang, X. Chen, M. Chen, S. Chen, and M. Shyu. A multiple instance learning approach for content based image retrieval using one-class support vector machine. In *ICME*, 2005.

[131] M. L. Zhang and Z. H. Zhou. Multi-label Learning by Instance Differentiation. In *Proceedings of Association for the Advancement of Artificial Intelligence*, 2007.

[132] Q. Zhang and S. Goldman. EM-DD: An Improved Multiple-Instance Learning Technique. In *Advances in Neural Information Processing Systems*, 2002.

[133] Z. H. Zhou and M. L. Zhang. Multi-instance Multi-label Learning with Application to Scene Classification. In *Advances in Neural Information Processing Systems*, 2006.

[134] X. Zhu. Semi-supervised learning literature survey. Technical report, Computer Sciences, University of Wisconsin-Madison, 2005.

Vita

Sudheendra Vijayanarasimhan was born in Chennai, Tamil nadu, India, the son of Mr. N. Vijayanarasimhan and Prema Kumari. He graduated from P.S. Senior Secondary School, Mylapore, Chennai in 2001. He received his B.Tech in Computer Science from the Indian Institute of Technology Madras (IITM) in 2006, and joined the UT-Austin Computer Vision Lab headed by Prof. Kristen Grauman in 2007 as a graduate research assistant. He has since worked on many interesting problems in computer vision and machine learning, with an emphasis on active learning, object recognition, and segmentation. He has authored a number of peer-reviewed scientific papers in the top conferences in vision and machine learning, served on their program committees, and holds a patent for his work on active classification with bounded resources done while an intern at Microsoft Research Redmond. He received the Dean's Award at IITM in 2002, the Dean's Excellence Award from the College of Natural Sciences at UT-Austin, the CS department's selective Micro-Computer Development Fellowship in 2006 and the Bert Kay Outstanding thesis award from the UT CS department in 2011. After post-graduation, Sudheendra plans to join Google, Mountain View to work on large-scale video classification problems.

Permanent address: A-14, Sivagami Flats, Sivagamipuram, I cross street,
Chennai - 600041, Tamilnadu, India

This dissertation was typeset with LaTeX[†] by the author.

[†] LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.