Dissertation

Hierarchical Cluster Guided Labeling: Efficient Label Collection for Visual Classification

Submitted by

Maggie Wigness

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2015

Doctoral Committee:

Advisor: Bruce Draper

Ross Beveridge
Adele Howe
Chris Peterson

Abstract

Hierarchical Cluster Guided Labeling: Efficient Label Collection for Visual Classification

Visual classification is a core component in many visually intelligent systems. For example, recognition of objects and terrains provides perception during path planning and navigation tasks performed by autonomous agents. Supervised visual classifiers are typically trained with large sets of images to yield high classification performance. Although the collection of raw training data is easy, the required human effort to assign labels to this data is time consuming. This is particularly problematic in real-world applications with limited labeling time and resources.

Techniques have emerged that are designed to help alleviate the labeling workload but suffer from several shortcomings. First, they do not generalize well to domains with limited a priori knowledge. Second, efficiency is achieved at the cost of collecting significant label noise which inhibits classifier learning or requires additional effort to remove. Finally, they introduce high latency between labeling queries, restricting real-world feasibility.

This thesis addresses these shortcomings with unsupervised learning that exploits the hierarchical nature of feature patterns and semantic labels in visual data. Our *hierarchical cluster guided labeling* (HCGL) framework introduces a novel evaluation of hierarchical groupings to identify the most interesting changes in feature patterns. These changes help localize group selection in the hierarchy to discover and label a spectrum of visual semantics found in the data. We show that employing majority group-based labeling after selection allows HCGL to balance efficiency and label accuracy, yielding higher performing classifiers than other techniques with respect to labeling effort. Finally, we demonstrate the real-world

feasibility of our labeling framework by quickly training high performing visual classifiers that aid in successful mobile robot path planning and navigation.

Acknowledgements

The successful completion of my dissertation was only possible because of the feedback and support I received throughout my graduate studies. Thank you to my committee members for thoroughly critiquing my research, writing and presentation skills throughout the various steps of my graduate program. You have forced me to grow and learn quickly so I can keep up with the best in the field. Thanks to Dr. Bruce Draper, Dr. Ross Beveridge and all of my fellow graduate students in the computer vision research lab at Colorado State. I am lucky to have had the opportunity to work and learn from you all. A special thanks to Dr. Steve O'Hara for his support and guidance during my first years as a graduate student while we worked on the Mind's Eye project together. Your confidence in me helped me gain confidence in myself.

Thank you to the Asset Control and Behavior Branch at U.S. Army Research Laboratory for the fruitful collaboration during the last two years of my dissertation studies. You all have been extremely welcoming. A special thanks to Dr. John Rogers for helping out with all the robotics interfacing for field tests, and providing me with opportunities to publish and collaborate with others in the robotics domain.

Thank you Matt Malensek for engaging in research discussions, life discussions, taking coffee breaks and for overall just being a stand up guy. You proof read more of my documents and listened to more practice talks than anyone else. I am truly thankful to have had your support, and am appreciative of our friendship. Thanks to Caleb Champlin and Brad Over for always providing me with an evening to escape research when I needed a break. Thank you to Connie Winter-Eulburg for following my progress, never doubting my abilities and

proving me with endless kind words and spiritual support throughout this journey. There are so many other friends who have supported me and I thank you all.

Finally, I need to thank all of my family and relatives. You all are amazing people and I am not sure I would have made it without your love and support. Thank you Mom, Dad, Paden, Bailey and Whitney. I love you and I hope I have made you proud.

Dedication

To my parents.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Efficient Visual Concept Labeling

1.1. Classification Tasks and Training Data Collection

Visually intelligent systems that require little to no human supervision are becoming more commonplace. Examples include autonomous vehicles and automated video surveillance. One core component in these technologies, and the motivating task of this thesis, is visual concept classification. Depending on the application, ***visual concept classification*** can include differentiating objects, scenes, terrains or activities from visual data.

In general, humans categorize concepts relatively easily even when visual properties such as color, scale or shape vary or when concepts are viewed from different perspectives, under different illuminations or under occluding circumstances. Visual classifiers, on the other hand, are still progressing to handle these variations with the same ease as humans. To learn well, most visual property variations must exist in the visual classifier's training set. Ultimately, this means that large sets of labeled training data are needed to produce high performing visual classifiers.

Although collecting large sets of visual data is a trivial task, the raw data itself contains no label information for supervised classifiers. The process of mapping visual data to a set of semantically meaningful labels requires human intervention. Figure 1.1 illustrates the general idea of labeling visual data. Assigning a label to a single image can be an easy task, but datasets of thousands to millions of visual data samples requires significant labeling time and resources.

The time consuming labeling process is a significant hindrance to supervised learning algorithms. Thus, there is a strong need for more efficient label collection techniques that

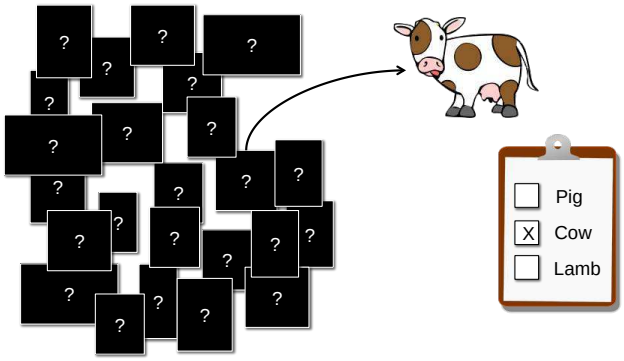


FIGURE 1.1. Illustration of label assignment to a single image.

reduce human labeling effort without significantly compromising classifier learning. In other words, a set of labeled data needs be collected quickly and include all underlying information available in the training set. The research in this thesis presents a framework that addresses this need.

1.2. Problem Definition

At its most basic level the ***label collection problem*** is the process of assigning a set of labels to a set of visual data. However, we identify several objectives that influence the design of our labeling framework. These include learning a label set, reducing workload, labeling a sufficient amount of data and maintaining label accuracy. Each of these objectives plays a critical role in determining the success of classifier learning. These objectives serve as a basis for discussion and analysis when comparing other label collection techniques with the research in this thesis.

1.2.1. Learning a Label Set

Since the training data is initially unlabeled we assume that the set of visual concepts in the data is also initially unknown. For example, training data may be collected by letting a camera capture data from an environment for a specified amount of time. The training data

is assumed to include the typical visual concepts a classifier may eventually want to learn, but the underlying concepts are not known in advance. This process of learning a label set is commonly referred to as ***concept discovery***. Discovery is an important task because classifiers will not learn to recognize concepts when no labeled training examples exist. The semantics of learned label sets will be discussed in greater detail in later chapters.

1.2.2. Reducing Labeling Workload

Given the motivation of this work, an obvious objective is to reduce labeling effort. The degree to which labeling effort is reduced is referred to as ***labeling efficiency***. This is relative to the overall effort required to label each training sample individually, and is discussed in greater detail in Chapter 3. Efficiency is generally a desirable property, but this research is specifically motivated by real-world applications that have limited time and resources to allocate to label collection. In this sense, efficiency can also be thought of as reducing the total amount of time it takes to collect labeled training data. This will be the main focus during real-world evaluations in Chapter 8.

1.2.3. Labeling a Sufficient Amount of Data

Trivially, reducing labeling effort could be achieved by simply labeling fewer training samples. However, a small label set may not be sufficient to train high performing classifiers. We may refer to this as ***exploitation*** of the training data. This objective is related to discovery but focuses on the quantity of labels, not classes, for the classifier to learn from.

1.2.4. Maintaining Label Accuracy

There is always possibility for error when humans assign natural language concepts to visual data. For example, a human may mistake a *cheetah* for a *leopard* or mistype *cat* as *car*.

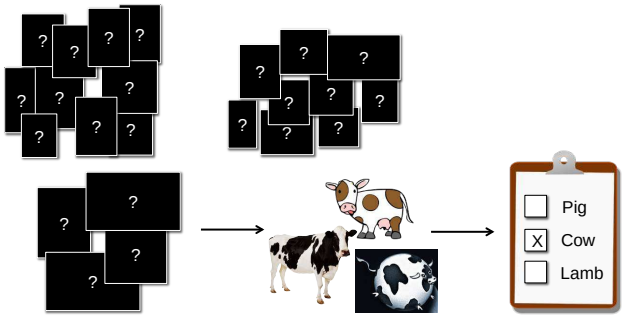


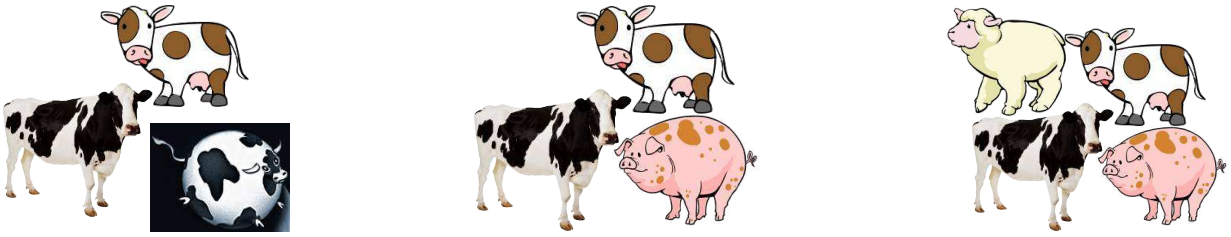
FIGURE 1.2. Illustration of a group-based labeling technique.

The fraction of non-erroneous labels is defined as ***label accuracy***. Conversely, the fraction of label errors is referred to as ***label noise***. High label accuracy is important because it provides the best opportunity for classifiers to learn, whereas label noise creates confusion during learning.

Beyond human error, most of the label noise discussed in this research is introduced by systems that employ group-based labeling to improve labeling efficiency. In this technique, visual data believed to represent the same visual concept are grouped together, and the entire group is assigned a single label that best describes all its data (illustrated in Figure 1.2). Details of this process are left for discussion in later chapters, but Figure 1.3 illustrates how group-based labeling introduces label noise when groups are formed from data representing multiple concepts. In this case, all groups are given the label *cow* because it is the dominating concept among images, but each group has a varying degree of label noise (images that do not represent *cow*).

1.2.5. Balancing Objectives to Produce High Performing Classifiers

Each of the objectives contributes to the overall goal of efficient label collection for visual classifiers. As hinted, there are trade-offs when attempting to optimize any one of these objectives, so the challenge of this problem is designing a labeling system that best balances



(a) Label Accuracy: 1.00 (b) Label Accuracy: 0.75 (c) Label Accuracy: 0.50

FIGURE 1.3. Illustration of label noise that may be introduced during group-based labeling. All three groups would be labeled with the majority concept *cow*, but have varying amounts of label noise.

all of them. Throughout this thesis, labeling techniques will primarily be evaluated by comparing classification performance as a function of labeling effort. However, objective criteria will be evaluated independently to show the trade-offs made by different labeling approaches and how these objective design decisions impact the final performance of visual classifiers.

1.3. Current Techniques and Shortcomings

Many techniques have emerged to improve the efficiency of label collection for large unlabeled sets of visual data. However, these techniques exhibit several shortcomings. First, techniques often assume a priori knowledge of the data is available, which is not the case for all applications. Second, techniques tend to favor either efficiency or label accuracy. High efficiency is often achieved at the cost of introducing significant label noise, but high label accuracy requires more effort. Third, techniques introduce labeling latency via iterative re-clustering or classifier re-training, which results in idle time for annotators during labeling. The rest of this section highlights important existing techniques and their specific shortcomings.

Active learning frameworks [1, 2, 3, 4, 5] label a subset of the most informative samples in the training data. Label collection and classifier learning are tightly coupled because classifiers are iteratively re-trained to make query selections. Incorporating a supervised classifier creates iterative labeling latency, and often requires knowing the visual concepts in advance because a labeled seed set is used to initialize classifiers. These frameworks are only efficient if a subset of the unlabeled data can sufficiently train classifiers since individual instances are labeled at each query.

Group-based labeling techniques are potentially more efficient than instance-based labeling because a single label is attached to multiple images simultaneously. Clustering [6, 7, 8] and topic modeling [9, 10, 11] have been used to partition unlabeled data into groups without supervision to discover the underlying concepts bottom-up. However, these techniques often assume the number of concepts is known to learn a one-to-one partition between concepts and groups.

Group-based labeling is a noisy strategy when groups contain data from multiple visual concepts (as in Figures 1.3(b) and (c)). Thus, some techniques introduce additional effort and labeling latency to improve label accuracy. Active clustering iteratively collects constraints and re-clusters data to improve clustered output [12, 13, 14]. Incremental clustering [15] has been used to iteratively remove small subsets of labeled data to make the next iteration of clustering easier. Some frameworks ask annotators to assign a label and then remove instances that do not match that label [16]. These approaches introduce label latency similar to active learning, and decrease efficiency as the focus is to assign noise-free labels.

While these techniques make an effort to reduce labeling and train high performing classifiers, their shortcomings need to be addressed. The biggest commonality among all

techniques is the desire to force discovery and labeling of visual concepts to occur at a specific granularity - the granularity believed to be of importance to the end classifier. In other words, these techniques ignore the hierarchical patterns and relationships of the data. This thesis research is designed with the hypothesis that exploiting hierarchical information in visual data will address the shortcomings of existing techniques, and produce high performing visual classifiers with minimal human effort.

1.4. Summary of Research: Hierarchical Cluster Guided Labeling

The process of mapping a natural language label to visual data is more complex than the problem statement suggests. Suggesting there is only a single correct label per data point is very restricting, yet existing techniques assume this when designing efficient labeling frameworks. The patterns seen in the underlying features of visual data can represent many visual properties. These similarities may map to visual concept labels that are either coarser or finer-grained than classifiers will eventually learn.

Group-based labeling is a productive technique to label a finite set of training data with less effort than labeling each data sample individually. However, trying to force groups to represent a finite set of labels usually results in significant label noise collection. We show that removing the constraints that force grouping to occur at a particular concept granularity provides a unique framework for an efficient labeling system. We refer to this framework as Hierarchical Cluster Guided Labeling. Specifically, we maintain a hierarchical clustering of visual data to create a space of groupings that can be mapped to a spectrum of visual concepts. In other words, our approach does not create a one-to-one mapping based on a finite set of labels. Instead, we seek to discover all underlying similarities in the groupings

and provide an appropriate label for them even if the granularity of the label differs from that of the end classifier.

The hierarchical clustering structure is large and redundant since each node is a subset of its ancestors. To maintain high efficiency our framework searches the hierarchy for locations of visual concept transitions. This is done by analyzing local structure of groups and changes in structure with respect to a local neighborhood. The idea is that significant changes in local structure are good indicators of visual concept transition. These transitions make up a small subset of groups that can be labeled with minimal human effort.

Our hierarchical grouping and label set provides a way to maintain better label accuracy without introducing latency or additional effort. While this approach may require more labeling queries because we label a larger set of visual concepts, the additional workload makes up for itself with improved label accuracy. This balance between efficiency and accuracy produces higher performing classifiers with respect to labeling effort. Finally, this technique does not introduce any labeling latency since the hierarchical structure is constructed only a single time, off-line, before labeling begins. This allows the labeling framework to be used in real-world scenarios where a large amount of label collection needs to be done quickly.

1.4.1. Summary of Contributions

This thesis presents a new group-based label collection strategy to quickly and efficiently collect labels to train visual classifiers. The contributions of this thesis include the following:

- An efficient labeling framework that can reduce labeling effort by a factor of ten while still training high performing classifiers.
- A technique that generalizes to any unlabeled data as it requires no a priori knowledge and discovers visual concepts bottom-up.

- A novel structural change measure used to evaluate local neighborhoods in a hierarchical clustering of data to identify the most interesting splits made during grouping.
- Results that indicate trading a small amount of label accuracy for higher efficiency does not significantly degrade classification performance.
- A real-world feasible and fast labeling framework.
- Presentation of the first navigation task-based evaluation of visual perception with respect to labeling interaction time.
- A multi-concept image inference technique that exploits the entire hierarchical label set to provide more information to classifiers.

1.4.2. Scope

Many factors can impact performance of a labeling framework. Unfortunately, the evaluation of all factors is beyond the scope of this thesis. We choose not to focus on the evaluation or optimization of any of the following:

- Feature representation: feature vector used to represent each image
- Hierarchical grouping construction: algorithm used to build a top-down or bottom-up hierarchical representation of the visual data
- Segmentation of multi-label images: algorithm used to break an image into multiple smaller samples that each represent a single concept
- Supervised classifier: the final classifier that learns from the collected labeled training data and serves as the primary form of evaluation

Instead, we use existing resources to provide what is believed to be quality, generic representations that provide the building blocks for the main research of this thesis.

1.4.3. Document Outline

Chapter 2 defines terminology, introduces datasets and provides a greater discussion of related research. Deeper motivation for labeling groups of data and using hierarchical clustering to form these groups is provided in Chapters 3 and 4. A detailed presentation and evaluation of our Hierarchical Cluster Guided Labeling framework is provided in Chapter 5. Variants on labeling order and hierarchical analysis are discussed and evaluated in Chapters 6 and 7, respectively. The collaborative work to integrate the current labeling framework into real-world applications is discussed in Chapter 8, and the use of hierarchical labels is discussed in Chapter 9. Finally, we summarize the work of this thesis and discuss future directions in Chapter 10.

CHAPTER 2

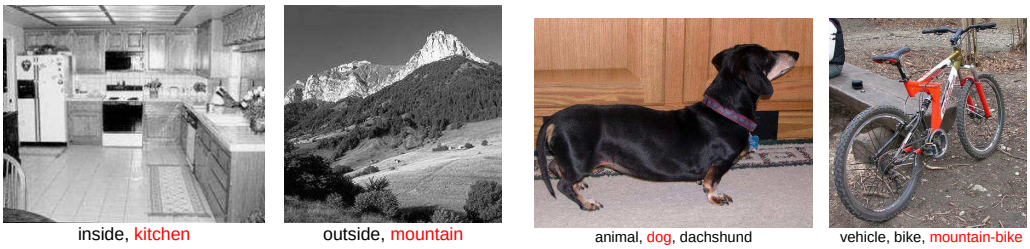
Background

This chapter includes a discussion of the meaning of visual concepts within different classification tasks and the associated data labeled for these tasks. This is followed by an overview of existing techniques that address label collection and other related problems. This overview is intended to show that there are two general approaches to label collection: instance-based and group-based labeling. The finer distinctions can be seen when analyzing the trade-off that each technique makes between labeling efficiency and label accuracy, which is discussed in greater detail in Chapter 3.

2.1. Visual Concept Labels

A ***visual concept*** in the context of object, scene, terrain or activity classification is typically a noun or verb. Adjectives can also represent visual concepts, but the scope of this work limits adjectives to a general concept description and not specific visual properties such as color or orientation. The sets of visual concepts learned by supervised classifiers are task dependent. Figure 2.1 includes examples of visual concept labels for benchmark data from two different classification tasks: scene and object. The label in red indicates the concept granularity the end classifier is interested in learning (based on the ground truth of the benchmark datasets). However, note that all labels associated with an image are valid visual concept descriptions.

We use these examples to illustrate the types of visual concept granularity we seek to label, versus the fixed granularity (indicated in red) that other techniques restrict themselves to. To reiterate, the labels for these examples are not an exhaustive list of descriptors of the



(a) Scene hierarchical labeling (b) Object hierarchical labeling

FIGURE 2.1. Examples of hierarchical visual concept label sets. Labels in red indicate the concept of interest to the classifier.

TABLE 2.1. Dataset overview

Dataset	Num Classes (K)	Num Images	Classification Task
UCI-Pendigits	10	10,992	digits
UCI-Letter	26	20,000	letters
13-Scenes	13	3,859	scene
MSRC-v2	23	591	object

images. For example, the dog could also be labeled as the color black or small, but these are not listed because of the defined scope of the problem.

Throughout this thesis, all visual concept labels will be denoted in italics, e.g., *dog*. The label set of interest to a classifier is denoted as $\mathcal{Y}$ and $|\mathcal{Y}| = K$. In the object labeling example, $dog \in \mathcal{Y}$. A hierarchical label set is denoted as $\hat{\mathcal{Y}}$ such that $\mathcal{Y} \subset \hat{\mathcal{Y}}$. For the same example, $animal,\ dog,\ dachshund \in \hat{\mathcal{Y}}$.

2.2. Image Datasets

Single concept and multi-concept image datasets are used throughout this thesis. Single concept datasets match exactly one ground truth label to each image, so each sample in the training set, $x_i \in \mathcal{T}$, is an entire image. Multi-concept datasets have a pixel-wise ground truth label map associated with each image. Thus, concepts need to be localized and segmented from the entire image. Each segmented region is treated as a training sample. Table 2.1 provides an overview of the benchmark datasets.

2.2.1. UCI-Pendigits and UCI-Letter

The Pendigits and Letter single concept datasets come from the UCI Machine Learning Repository[1]. The Pendigits (often referred to in this thesis just as Digits) dataset is a set of handwritten digits, *0* to *9*, collected from 44 writers. There are 250 samples per writer giving a total of 10,992 data instances. The Letter dataset consists of 20,000 character images, *A* to *Z*, from 20 different fonts with random distortions. The features for each of these datasets can be found at the referenced link.

2.2.2. 13-Scenes

The 13-Scenes dataset [17] is comprised of 3,859 single concept grayscale images that represent 13 natural scene classes. Examples of classes can be seen in Figure 2.2. A subset of this dataset is also used in this thesis, referred to as 5-Scenes, which includes the classes *coast, highway, living room, suburb* and *tall building*. We use GIST descriptors [18] as features for each image which try to capture dominant spatial structures of scenes, e.g., naturalness, openness and ruggedness.

The label ground truth for this dataset does not include hierarchical concepts, but we define three sets of coarse-grained labels using the spatial structures the GIST descriptors try to capture. These are: *inside* and *outside*, *artificial* and *natural*, and *open* and *closed*. Each scene class is placed into one of the two coarse-grained categories for the three sets (seen in Table 2.2).

[1] `https://archive.ics.uci.edu/ml/datasets.html`

FIGURE 2.2. Example images from the 13-Scenes dataset.

TABLE 2.2. Hierarchical label set relationships for the 13-Scenes dataset.

Class	*inside*	*outside*	*artificial*	*natural*	*open*	*closed*
bedroom	X		X			X
coast		X		X	X	
forest		X		X		X
highway		X	X		X	
inside city		X	X			X
kitchen	X		X			X
living room	X		X			X
mountain		X		X		X
office	X		X			X
open country		X		X	X	
street		X	X			X
suburb		X	X			X
tall building		X	X			X

2.2.3. MSRC

MSRC-v2 [19] dataset includes 591 multi-concept images from a total of 23 object categories: *airplane, bicycle, bird, boat, body, book, building, car, cat, chair, cow, dog, face, flower, grass, horse, mountain, road, sheep, sign, sky, tree* and *water*. Only 21 classes are

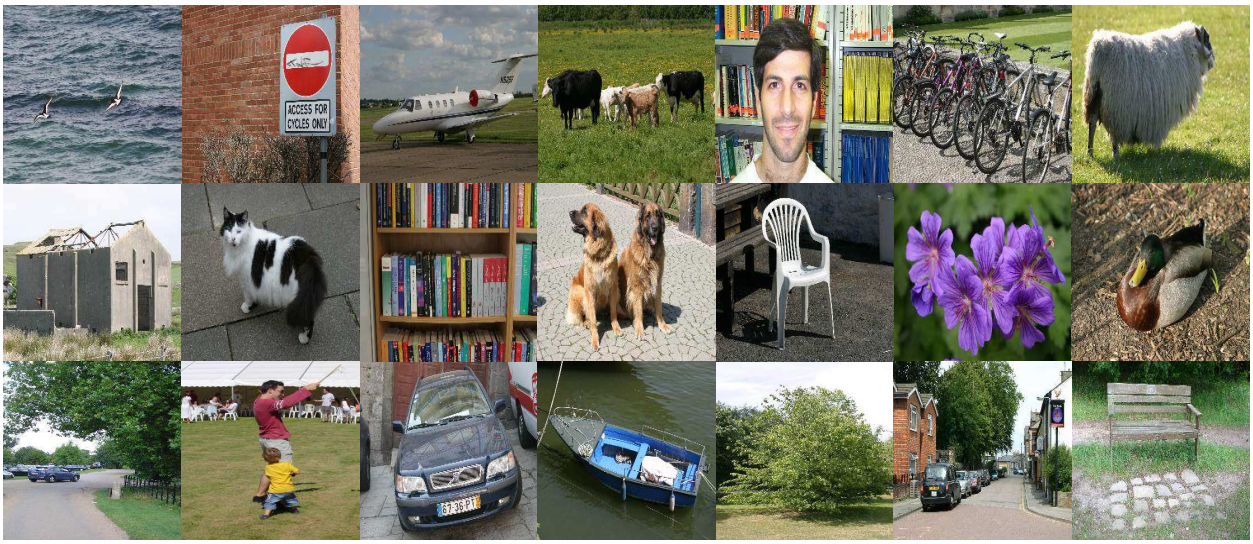


FIGURE 2.3. Example images from the MSRC-v2 dataset.

commonly used because *horse* and *mountain* have a small number of instances. Example images from the MSRC-v2 can be seen in Figure 2.3.

Object classification for this data is not performed on the entire image but instead on regions from within the image. That is, potential objects are first localized and segmented. Many segmentation [20, 21, 22] or region proposal [23, 24, 25] techniques can be used for localization. We use publicly available code [7] based on using multiple segmentations of an image to localize objects [26].

2.3. Instance-Based Labeling Techniques

2.3.1. Active Learning

Active learning frameworks reduce labeling by selecting a subset of training samples to label. The idea is that many training data are highly similar and provide redundant information to the classifier. Such redundant samples do not improve classifier performance but still requires labeling effort. Thus, active learning is an iterative labeling process that uses a supervised classifier to identify the most informative unlabeled samples, i.e., diverse

relative to the currently labeled training set. A general overview of active learning is provided by Settles [27].

The main distinctions between active learning frameworks are classifiers and selection criteria used during the labeling process. In the vision domain, most frameworks use uncertainty sampling to make selections during the labeling process. Selection techniques include probabilistic uncertainty via margin sampling [5], entropy [4, 28] and least certainty [1, 29]. Any probabilistic classifier, e.g., support vector machine, can be used with the uncertainty sampling strategies. New classifiers have also emerged from the active learning domain including a k-nearest neighbor probabilistic variant [1], and a Gaussian process regression method [3] that models uncertainty using the posterior mean and variance.

Li and Guo [2] combine uncertainty sampling with information density to improve selection. They show that selecting uncertain samples in dense areas of feature space is more likely to fit the expected distribution of testing data than using uncertainty alone. Typically, an annotator is asked to provide a class label to each selection, but Joshi et al. use active learning to ask an annotator if a labeled sample and an unlabeled sample are from the same class [30]. Binary queries are used to improve the overall speed of the system while still collecting labels after each "yes" response.

Most active learning techniques assume the label set is known a priori. Specifically, it is assumed that at least one sample of each concept has been labeled to form a seed set that initially trains the supervised classifier. This assumption does not address the discovery task that may be important in domains that have little to no knowledge of the local environment. Joshi et al. demonstrated that their active learning framework functions with an incomplete

seed set [5], but their framework discovered visual concepts at a rate slightly worse than random selection.

Active learning also introduces latency with the use of supervised classifiers to make selections. At each labeling iteration, the new labeled sample(s) are added to the labeled set and the classifier is re-trained. The human annotator must wait idle during this re-training stage. This latency is addressed in many frameworks by selecting multiple samples at each labeling iteration so the classifier is re-trained less often. However, if a set of unlabeled samples make up a dense region of feature space and the classifier identifies them as being the most uncertain, then many similar samples are selected for labeling which defies the underlying idea behind active learning.

2.3.2. Crowdsourcing

Crowdsourcing has emerged as a way to split a problem into smaller units of labor and distribute that task to a set of human resources. This is becoming a popular way to collect labeled image data [31, 32]. Marketplaces such as Amazon Mechanical Turk[2] are used to pay humans to perform specific annotation tasks. Other sites seek volunteers for labeling related tasks. Tomnod[3] has projects where specific objects and places need to be labeled in satellite imagery, and LabelMe[4] [33] has a collection of images that need all visual concepts to be outlined and labeled.

Crowdsourcing is related to the labeling workload problem, but overall does not provide a more efficient label collection solution since all samples are labeled individually. Crowdsourcing has been combined with active learning [34], but Vijayanarasimhan et al. note that

[2]https://www.mturk.com/mturk/welcome
[3]http://www.tomnod.com/
[4]http://labelme.csail.mit.edu/Release3.0/

crowdsourcing is not very effective with active learning since the classifier must be re-trained after each labeling iteration [35]. The benefits of collecting data in parallel via crowdsourcing is eliminated. Thus, they introduce an active learning framework that identifies a set of annotations (labels and bounding boxes) within a budget to collect more label information in parallel before the classifier is re-trained.

Crowdsourcing introduces new challenges when attaching labels to data. Chilton et al. use crowd sourcing to generate taxonomies of visual data, but the study showed that individual annotators are inconsistent in the vocabulary and relationships they identify [36]. Thus, the distributed results from each annotator need to be reconciled to form a common vocabulary for the classifier to be trained. Further, although the total labeling time can be reduced by distributing data and collecting labels in parallel, crowdsourcing may not be a viable solution for all domains. For example, money may not be available to pay people to label data. Even when money is available, the data may not be able to be released to the public for security reasons. This is often the case for military applications.

2.4. Group-Based Labeling Techniques

The main idea behind group-based labeling is that multiple data samples can be labeled simultaneously. Thus, the goal is to form groups that share a single common visual concept. The labeling effort is reduced to m queries where m is the number of groups formed, and in most cases $m \ll n$, where n is the total number of training samples. This thesis also leverages group-based labeling. This is motivated more formally in Chapters 3 and 4.

2.4.1. Partitional Clustering

In addition to efficient labeling, the unsupervised nature of clustering works well with unlabeled data and the task of discovery. Clustering algorithms group data by finding re-occurring feature patterns. Most partitional clustering techniques attempt to significantly reduce the labeling effort by finding a one-to-one mapping between groups and visual concepts [6, 8, 7]. A perfect one-to-one mapping results in each group containing data from the same concept, and no two groups representing the same concept. Ideally, these concepts perfectly represent the concepts of interest to the end classifier.

Feature representation plays a crucial role in learning good partitions, and the primary concern of partitional clustering approaches has been on finding the appropriate representation. Tuytelaars et al. provide an overview of common partitional clustering algorithms and invariant feature representations, e.g., SIFT descriptors [37], used for unsupervised object discovery [8]. They show that different normalization, interest point detectors and dimensionality reduction techniques affect the output of clusters. Dai et al. use an ensemble of weak training sets to generate an improved proximity matrix used for spectral clustering [6]. Lee and Grauman introduce context descriptors using superpixel neighborhoods [7] in addition to appearance features.

Most partitional clustering algorithms require that m be predefined and since most approaches look for a one-to-one mapping they assume the number of visual concepts is known in advance so $m = K$. Even with this assumption, perfect partitions of visual data have proven to be difficult. Overall cluster quality is typically low, i.e., a group of images usually represents multiple concepts, and some concepts are never discovered. This has led to many spin-offs from traditional partitional clustering that try to improve cluster quality.

Biswas and Jacobs present a subclustering technique that uses a subset of the total training images to learn K groups of data [38]. Each group is relatively small, on the order of 10 samples, but by focusing only on small dense regions of feature space the authors show cluster quality remains high. The total labeled data remains small for this approach however, and the authors state it should be enhanced with more data before training classifiers. Other spin-offs take a more active approach to learn groups that can be labeled.

2.4.2. Active and Incremental Clustering

Active clustering is an adaptation of constrained clustering that assumes the training data is completely unlabeled. Constrained clustering uses a set of labeled data to add side constraints that define desired grouping relationships between the data to improve clustered output [39, 40]. Active clustering techniques collect similar constraints, but in an iterative fashion to refine the clustered output. The iterative process has strong parallels to active learning as the constraints are not random, but meaningful based on the current state of the clustered output.

Pairwise must-link and cannot-link constraints are a common active clustering binary query [12, 13]. At each iteration an annotator determines if two data are from the same class or not. This information is used to augment the feature representation so data with cannot-link constraints are pushed further away and must-link constraints are pulled together. Gilbert and Bowden also seek constraint information but ask the annotator to identify samples that are true or false positives relative to the majority concept of their group [14]. Using this information the technique mines for the discriminative and relevant features found in the true-positive samples that do not exist in the false-positives.

Like partitional clustering, active clustering usually suffers from the assumption that the number of visual concepts are known in advance to find a one-to-one mapping. Incremental clustering approaches do not make this assumption so the one-to-one mapping is relaxed. Instead, a set of unlabeled data is clustered at each iteration and a single group is selected to be labeled. This continues until all data have been labeled.

Incremental clustering techniques use the incrementally collected labeled data to improve clustering for the next iterations. Lee and Grauman cluster the "easiest" subset of unlabeled data (based on an "objectness" measure and contextual information) at each iteration [15]. The labeled groups are used to enhance context descriptors for the next round of clustering. Galleguillos et al. cluster all unlabeled data at each iteration and label the most compact group in the output [16]. These labels are then used to adjust similarity measures of the data.

Active and incremental clustering both learn from previous labeling iterations much like active learning. However, just like active learning these techniques introduce latency into the labeling process since unlabeled data needs to be re-clustered at each iteration.

2.4.3. Topic Models

Topic models are statistical models originally designed for the text domain that discover underlying topics in a set of documents. Topics are discovered by looking at the frequency of words in a document with the assumption that the mixture of certain words hint at the underlying topic. These frequencies are used to establish a distribution across possible latent topics in a document. Topic modeling has seen a progression from latent semantic analysis [41] to probabilistic latent semantic analysis (pLSA) [42] and latent Dirichlet allocation

(LDA) [43]. Blei provides a good overview of topic modeling and current variants of LDA for the text domain [44].

In the vision domain, images take on the role of documents and visual features take on the role of words. The "topic" or visual concept distribution found using topic modeling provides a way to group data. The use of a bag of features representation with pLSA has been used for unsupervised concept discovery and classification of objects [10] and scenes [9]. Supervised topic models have been used for scene classification by using existing annotations of objects as visual features of scene images [11].

The primary emphasis of topic modeling is concept discovery, and has been used in other vision related problems that do not focus on classification. Russell et al. group images using LDA and use these groupings to determine the validity of image segmentation [26]. Topic modeling has also been used to improve object localization [45]. Like clustering, topic models usually learn K group topics by assuming the number of concepts is known in advance. Thus, a one-to-one partitioning of the data is the goal.

2.5. Use of Hierarchical Labels and Features

The concept of hierarchical visual semantics has appeared in many related problems. Deng et al. use pre-defined label sets to define a set of binary hierarchical queries that determine if certain concepts exist in an image [46]. They use multi-label images and collect binary labels that represent the existence of 200 object classes. Querying for a coarse-grained concept can provide binary information about multiple classes at once. Discovering taxonomies of visual data have been shown to be naturally hierarchical. As previously mentioned, this has been shown via crowdsourcing [36]. Bart et al. organize images into a tree structure to perform unsupervised taxonomy discovery [47]. They predefine the number

of levels in the tree and focus on hierarchies created using color features to improve image search and compression. Related, Li et al. construct a "semantivisual" hierarchy using visual features and image tags [48]. This technique makes use of visual data available on the web that has some label information associated with it.

Hierarchical feature representation also appears in the literature. Mahmood et al. use hierarchical clustering to identify meaningful features for classification of image sets [49]. Labeled and unlabeled data are clustered together and clusters higher in the tree are used to identify global similarities whereas clusters near the bottom of the tree are used to capture subtle class differences. Sivic et al. use hierarchical features with topic modeling [50]. Different feature representations are used to construct a predefined number of levels in an image hierarchy. Coarser-grained features, i.e., fewer dictionary words in a bag of features representation, are used higher in the tree, and finer-grained features, i.e., more dictionary words, are used in the lower levels. However, performance is still evaluated at the level of being able to extract a perfect one-to-one mapping from the hierarchy.

While it is recognized that natural language has hierarchical semantics, current literature does not seem to embrace discovery or labeling at varying granularities. Hierarchical feature representations are still only used to provide labels at a granularity defined by $\mathcal{Y}$. In the next chapters we outline the potential benefits of exploiting the hierarchical nature of visual labels during the discovery and labeling process. We show how this can be used to reduce label noise without the addition of labeling latency or assumptions introduced by existing techniques.

CHAPTER 3

Labeling Techniques

At the most basic level, the goal of labeling is to create a set of (x_i, y_i) pairs, where y_i is the label of the concept that x_i represents. For instance-based labeling techniques, the human interaction required to achieve this is straightforward (illustrated in Figure 1.1). However, human interactions required from group-based techniques vary greatly in the literature. This can be attributed mostly to the threat of introducing label noise when applying a single label to multiple data at the same time.

This chapter discusses various labeling interactions that have been used during the label collection process. These interactions are analyzed and compared with respect to cognitive load, inclusion of label noise and its impact on classification performance, and the reduction of labeling effort relative to a fully supervised labeling technique. We define a ***fully supervised labeling technique*** as the process of iteratively pairing a training sample with a label until the entire available training set has been labeled. Thus, a total of n labels must be provided by the human annotator. All experiments include the results achieved when a fully supervised labeling technique is used. These are labeled as *supervised* in figure legends.

3.1. Labeling Interactions

Human labeling interactions vary among different labeling frameworks, but each interaction is designed to provide necessary information to the system that ultimately leads to the collection of labels. The following are types of interactions currently used in techniques that address labeling workload:

(1) Providing a class label for an image [1, 2, 3, 5]

(2) Providing a class label for a group of images [6, 9, 10, 15, 16, 7]

(3) Providing a label that indicates a group lacks a common label [51]

(4) Removing an image from a group [16]

(5) Providing binary feedback about whether or not two images represent the same label [12, 13, 30]

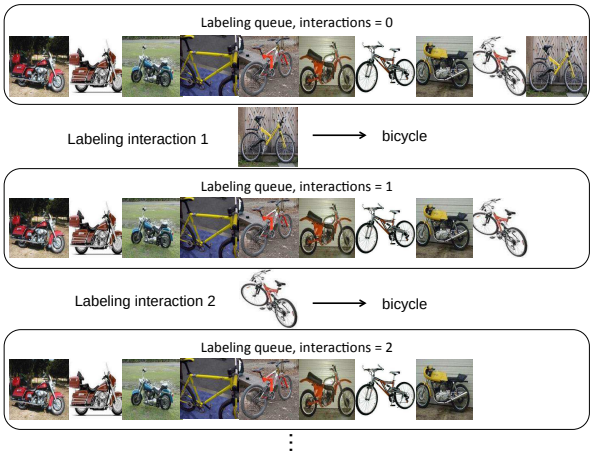
(6) Selecting examples of true positives and false positives from a group of images [14]

We note that the cognitive load for these interactions is not equivalent. Providing binary feedback about two images is the simplest task. Providing a concept label requires more cognitive load as the human needs to map an image to a natural language concept. Further, providing a label for a group of images requires the processing of multiple data samples simultaneously. While cognitive load is an important factor and will be discussed throughout parts of this thesis, it is not a primary focus of this work. Further, to quantitatively compare different labeling frameworks, we define the effort required to label a dataset with n training samples as:

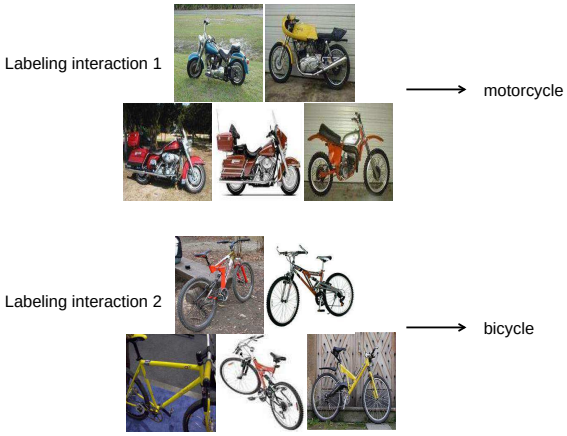
$$\text{Labeling Effort} = \frac{\#\text{ interactions}}{n} \tag{1}$$

Using Equation 1, the labeling effort of a fully supervised system is equal to 1.0 since labeling interaction 1 is performed n times, once for each training image. The values of other systems describe their labeling effort relative to a supervised approach. Any value less than 1.0 is more efficient than a fully supervised labeling approach, which is a primary objective in the problem defined in this thesis. An example of greater efficiency is illustrated using a small set of 10 images seen in Figure 3.1(a). A fully supervised approach iteratively labels all 10 images as depicted in Figure 3.1(b). Assume a grouping technique exists (details of

(a) Example training data



(b) Fully supervised instance-based labeling interactions.



(c) Group-based labeling interactions.

FIGURE 3.1. Labeling example demonstrating the efficiency of group-based labeling over a fully supervised instance-based labeling technique.

grouping are discussed in the next chapter) that forms two groups as seen in Figure 3.1(c), and provides labels using labeling interaction 2. This group labeling example illustrates how labeling effort can be optimally reduced by assuming the total number of classes is known in advance. One-to-one partitions of data require $\frac{K}{n}$ labeling effort. In this example, only two interactions are performed, reducing labeling effort to 0.20.

The objective of reducing labeling workload suggests that the range of labeling effort should be $[0.0, 1.0]$. This is true for instance-based labeling techniques that use labeling interaction 1, e.g., active learning. However, techniques that use other types of labeling interactions do not necessarily conform to this range. As other techniques are discussed throughout this thesis, their range of values for labeling effort will also be described.

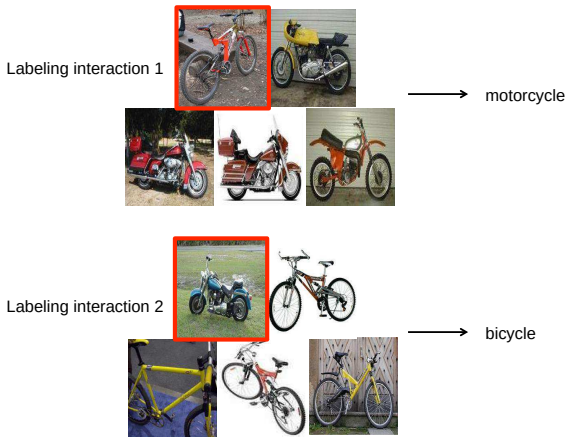


FIGURE 3.2. Group-based labeling where majority labeling results in label noise. Images outlined in red indicate label noise.

3.2. The Trade-Off: Label Efficiency vs Label Noise

Unfortunately, group-based techniques are not always able to partition data as cleanly as in Figure 3.1(c). Groups formed from the 10 images could have looked more like Figure 3.2, and assigning a single class label to these groups of images is less straightforward. Many techniques use ***majority labeling*** in this situation, where the human annotator provides the label that the majority of images represent. The label noise introduced by majority label assignment is illustrated by the images outlined in red.

Limiting label noise is important because it impacts the training and performance of supervised classifiers. This threat has led to the emergence of label interactions 3 through 6 in the previously mentioned list of interactions. These are interactions are used in group-based techniques to reduce or completely eliminate label noise. These interactions (3 through 6) are performed in addition to asking for a group label (interaction 2). Thus, these techniques trade some labeling efficiency for higher label accuracy.

For example, Galleguillos et al. introduce a largest subset labeling technique [16]. Each labeling iteration involves providing the majority label concept to a group of images followed

by the removal of images that do not match this label. In the previous example (Figure 3.2), the largest subset labeling technique requires two interactions to label each group, one to provide the label and one to remove the red outlined image that does not match that label. This is a total of four interactions, and still yields greater efficiency, 0.40 labeling effort, compared to a fully supervised labeling approach. The range of labeling effort for this approach has the ability to exceed 1.0 if the largest subset makes up a plurality instead of a majority of samples. This can happen when a group of data is composed of more than two classes. In this case, labeling interaction 4 is performed more times than the total number of samples that receive a label.

Labeling interaction 5 collects binary constraint information and has been used with active clustering frameworks that try to augment feature representations to find a one-to-one partition. Figure 3.3 shows a set of binary interactions being collected, which are then used to form groups that are labeled using majority labeling. Note, this is a toy example that depicts only a single iteration in the active clustering process. Further, more than three binary constraints are typically collected during each iteration. Although these interactions are cognitively easy, we will show in our experiment comparisons that the number of binary interactions required for the entire active clustering process typically far outweighs the total number of training samples. In a worst case scenario, an active clustering approach could query for all possible binary constraints, n^2 labeling interactions, and then assign a label to each group. Thus, the range of labeling effort for this approach is $[0.0, n^2 + K]$

An intuitive way to avoid collection of label noise is to simply not label groups that represent multiple concepts. Some of our initial work [51] tried to enforce this strict labeling policy so labels are only assigned to groups that represent a single concept, as in Figure 1(c).

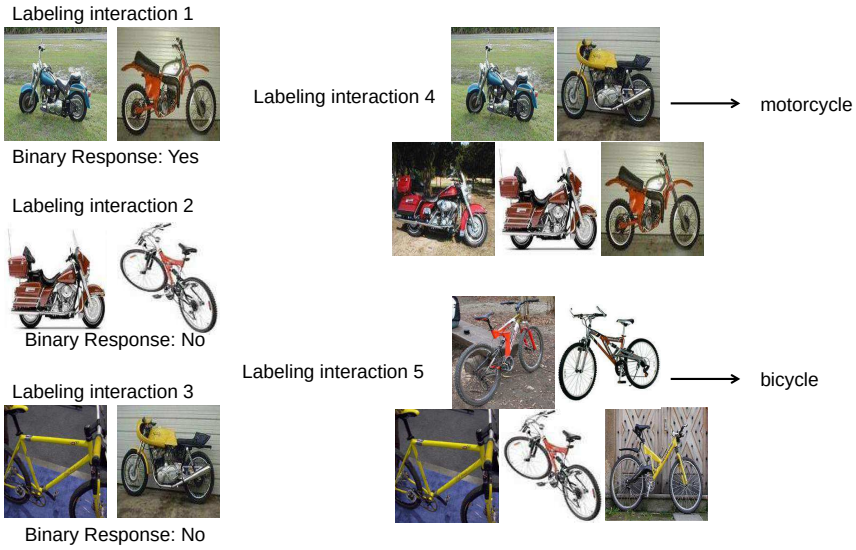


FIGURE 3.3. Binary constraint feedback followed by group-based labeling.

Otherwise, labeling interaction 3 is employed which assigns the label "mixed", indicating label noise is present. A "mixed" label does not collect any information for the classifier and does not collect label noise, but still requires a labeling interaction.

This initial framework, coined Selective Guidance (SG), includes a model that predicts how likely a group is to represent a single visual concept from $\mathcal{Y}$. The details of this model are provided in Appendix B, but the results of SG help motivate the major design in this thesis.

3.3. Experiments to Motivate Majority Group-Based Labeling

Labeling multiple images simultaneously naturally indicates higher efficiency than labeling individual instances. Yet, the issue of label noise that arises from group-based techniques has influenced many other existing systems to sacrifice efficiency for label accuracy to ensure a high performing classifier can be trained. While label noise is a reasonable concern, we present some results to motivate the use of majority group-based labeling without additional labeling interactions to remove label noise.

TABLE 3.1. Evaluation details for group-based versus instance-based labeling techniques.

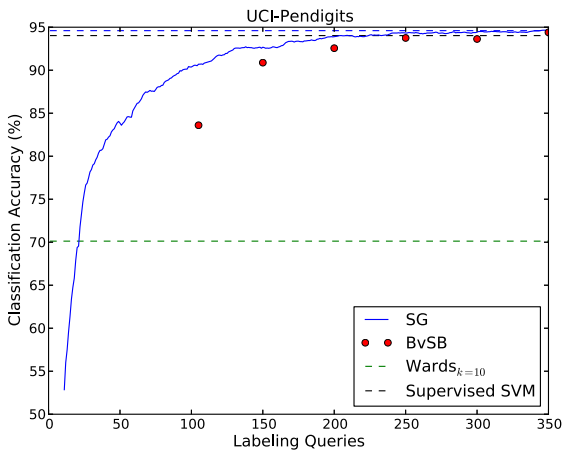
Dataset	# Training	# Testing	Classifier
UCI-Pendigits	5,100	2,000	SVM-linear
UCI-Letter	7,100	5,000	SVM-rbf
13-Scenes	2,500	500	SVM-linear

3.3.1. Group-Based vs Active Learning

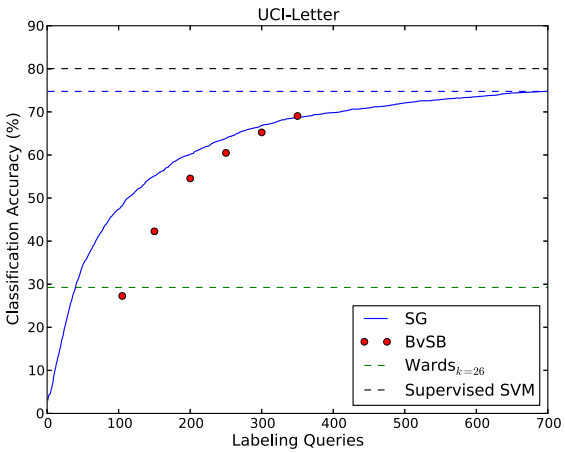
In this experiment we show the benefits group-based labeling provides compared to active learning (instance-based labeling), using the SG framework. Groups labeled by SG are compared to the margin sampling active learning framework BvSB [5]. Since SG does not collect label noise, this experiment demonstrates the benefits of labeling multiple images simultaneously. The technique labeled $Wards$ in these experiment figures is a baseline majority group-based technique and does collect label noise. This method will be discussed in context with other grouping methods in the next chapter.

Recall that SG may result in an interaction that produces no new label information, whereas BvSB will collect a new label at each iteration. We use the UCI-Pendigits, UCI-Letter and 13-Scenes datasets to compare classification accuracy as a function of labeling interactions/queries achieved by these two systems. Experiments are averaged over 20 trials of random training and testing partitions, seen in Table 3.1. The total labeling effort of the BvSB framework is the size of the seed set plus the total number of images labeled during active learning iterations. Classification accuracy for BvSB is only reported once active learning iterations begin.

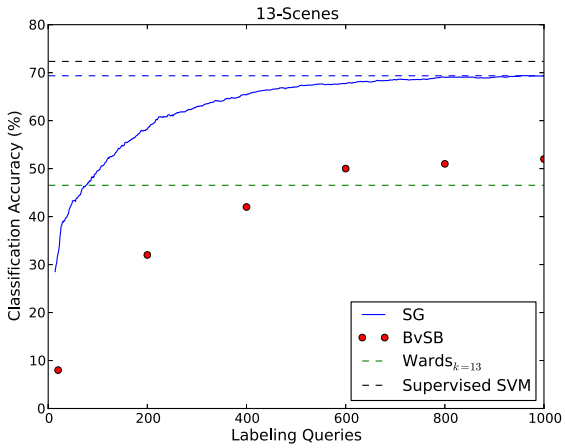
Figure 3.4 shows the classification accuracy per labeling query for three datasets. For all three experiments, SG outperforms BvSB early in the labeling process, and even after many labeling queries are answered, SG never performs any worse than BvSB. Further, SG always approaches the classification accuracy of a completely supervised approach, and does so with



(a) ~ 0.07 labeling effort (350 labeling queries)



(b) ~ 0.10 labeling effort (700 labeling queries)



(c) 0.40 labeling effort (1,000 labeling queries)

FIGURE 3.4. Classification accuracy per labeling query for the (a) Pendigits, (b) Letter and (c) 13-Scenes data sets.

less labeling effort. The 13-Scenes experiment, Figure 4(c), shows that active learning is only efficient if a subset of data is sufficient to train a classifier. BvSB fails to approach the fully supervised classification performance like it did in the other experiments, yet 1,000 queries for this experiment has already reached 0.40 labeling effort. Thus, for more challenging datasets the efficiency of instance-based labeling may not be as significant.

These results indicate that group-based labeling can be more efficient than instance-based labeling, even when some labeling queries, i.e., "mixed", do not provide labeled data for the classifier. This is because the remaining queries result in multiple images receiving labels

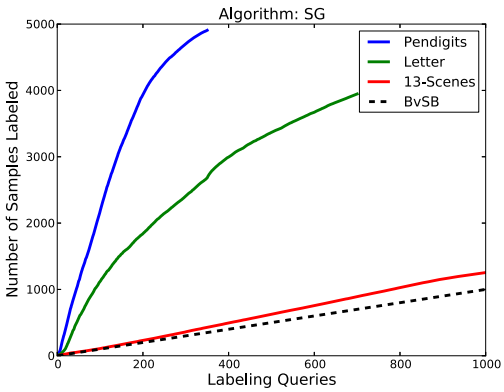


FIGURE 3.5. Labeled samples as a function of labeling queries.

simultaneously. This can be seen in Figure 3.5, which shows the number of training samples that have received labels as a function of labeling queries. Notice that as the datasets get more challenging the gap between instance-based labeling and SG shrinks. This is partially caused by many groups receiving "mixed" labels and the groups that represent a single concept being much smaller than in the other datasets.

When SG collects "mixed" queries, classifier performance cannot improve but labeling effort does increase. On average, the fraction of non-mixed labels assigned during the experiments was 0.64 for Pendigits, 0.71 for Letter and 0.60 for 13-Scenes. Even though SG outperformed instance-based labeling, 30-40% of labeling effort was essentially wasted with respect to the end classifier. This leads us to question whether SG's labeling is too restrictive, and we next analyze the impact of label noise on classification accuracy.

3.3.2. Impact of Label Noise on Classifier Performance

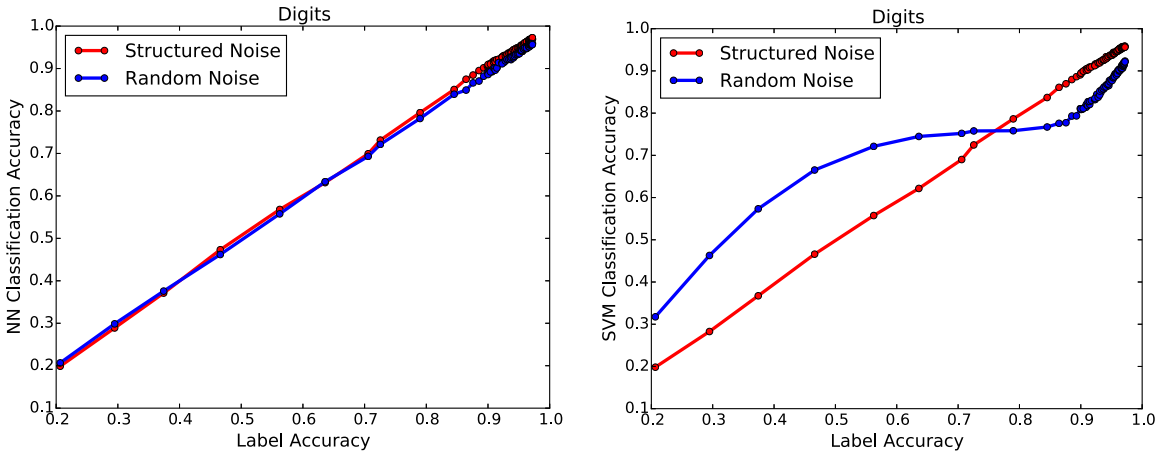
Labeling noise undoubtedly impacts classifier learning [52], but group-based labeling noise is different than other types of noise explored in the current research. Research on the design of classifiers robust to label noise simulate random noise [53, 54], where a percentage of samples are chosen to receive a random incorrect label, or instance-based human error [55,

56] for evaluation. Noise introduced by majority group-based labeling has an underlying structure because the grouping algorithm partitions by a pattern found in feature similarity. Thus, random and structured noise likely impact classifiers differently.

We test this theory using the Digits dataset with a nearest neighbor (NN) classifier and a support vector machine (SVM). To model group-based structured noise we run k-means on the training set for increasing values of K. After each clustering, the K groups are assigned their dominating class label and the data is used to train the classifiers. Using the same percentage of noise introduced by k-means at each iteration, a classifier is also trained with random noise assigned to the training data. Increasing values of K show the performance of classifiers as label noise decreases.

Classification results are shown in Figure 3.6 as a function of label accuracy. The structured k-means output performs worse than random noise when label accuracy is low because the value of K has not yet reached the number of total classes in the dataset. This means labels for all classes cannot be collected which significantly impacts classifier learning. This is discussed further in the next chapter as well. The two classifiers are affected by the noise in different ways, but overall, structured noise appears to negatively impact classifier learning less than random noise after label accuracy improves to around 70%.

Unfortunately, the threat of label noise collection has resulted in many group-based labeling techniques to emphasize label accuracy by introducing more labeling interactions. In the case of SG, those extra labeling interactions are in the form of "mixed" labels when a group of images represents multiple concepts. Interestingly, had SG taken a majority labeling approach, classification with respect to labeling interactions would have improved. This can be seen in Figure 3.7. In this experiment SG-majority ends up with an average label accuracy



(a) NN Classification - Digits (b) SVM Classification - Digits

FIGURE 3.6. Comparison of classifier performance given structured and random label noise.

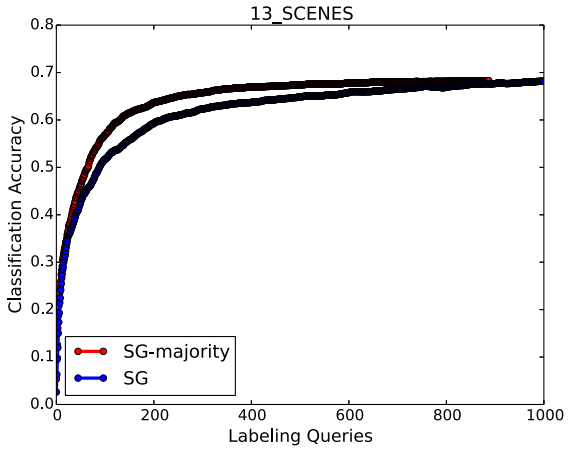


FIGURE 3.7. Classification accuracy for the SG labeling framework when using strict perfect label accuracy labeling and majority labeling.

of ~ 0.67. However, with this noise came an increase in correctly labeled training samples for the classifier to learn from. This balance of efficiency and label accuracy outperforms a strict emphasis on label accuracy.

Other group-based labeling experiments also show that emphasizing only efficiency (using small values of K) does not always produce enough accurate label information for classifiers to learn well. The experiments from Figure 3.4 included the results from a one-to-one

partitional clustering baseline denoted as "Wards". This technique required exactly K interactions, significantly less than SG, but yielded significantly lower classification performance. The average label accuracy across trials for Wards in the three experiments was 0.81 ± 0.20 for Pendigits, 0.41 ± 0.27 for Letter and 0.50 ± 0.18 for 13-Scenes. Although extremely efficient, this noisy data trains classifiers with anywhere from 25-50% performance degradation relative to a supervised classifier. Lee and Grauman [15] test their noisy label collection using the MSRC-v0 object dataset. They favor efficiency by collecting labels with ~ 0.015 labeling effort, resulting in label accuracy of 0.60. The classifier trained with the data yielded 0.49 classification performance compared to 0.64 for a fully supervised approach. With slightly less efficiency the label accuracy may improve and produce classifiers that approach performance of a fully supervised labeling technique.

3.4. Summary

Existing labeling techniques tend to focus on either labeling efficiency or label accuracy. Group-based labeling has traditionally favored the efficiency side of the problem. However, the threat of collecting label noise has shifted many group-based labeling techniques to emphasize label accuracy by introducing more labeling interactions. We have shown that group-based labeling noise and random noise impact classifiers differently, and that when structured noise is limited, classifier learning does not significantly suffer. In the next chapter we discuss why low label accuracy is so common in group-based techniques. We provide a remedy for this with the use of hierarchical clustering and an associated hierarchical label set. This allows us to adopt majority group-based labeling that is efficient but also limits the collection of label noise.

CHAPTER 4

Visual Concept Grouping

We begin by briefly recapping the benefits of group-based labeling. Labeling groups allows multiple images to receive a label simultaneously, thereby increasing efficiency. Unsupervised grouping is more amenable to applications where little to no a priori knowledge is available and bottom-up discovery of concepts must be performed. In the vision domain, clustering is by far the most common grouping technique used to address the label collection problem. Since this thesis also uses clustering to form groups, this chapter focuses only on clustering algorithms to form groups. However, similar trends can be seen for other grouping techniques such as topic modeling. Note that throughout this thesis the terms *cluster* and *group* may be used interchangeably to represent a set of data.

The previous chapter discussed the emphasis that existing group-based labeling frameworks place on label accuracy, thereby reducing efficiency. We discuss why the current group-based techniques are forced to do this. We provide examples and demonstrate the shortcomings of the narrow view that existing techniques take when labeling groups. We use this as motivating evidence to move to hierarchical clustering and hierarchical label sets.

4.1. Visual Data Properties, Feature Similarities and Label Mapping

Unsupervised clustering forms groups based on patterns and similarities found in the underlying feature representation of data. Group-based labeling assumes feature similarities encode visual concepts, so a single label can be applied to the entire group. The problem with existing techniques, however, is they force this encoding to occur at a particular label granularity. This is seen in the literature at two different levels. First, the global parameter

that defines the number of groups to learn (m) is typically set to a value equal to or very near the number of ground truth classes K. Second, all labeling, evaluation and analysis of the groupings is performed at the level of a benchmark ground truth label set.

Given that the long term goal of label collection is to train classifiers, the hope that visual properties can be perfectly partitioned at the level of the classifier label set is understandable. One problem however, is that the relevancy and consistency of visual properties are class dependent, yet these properties are encoded in a global feature representation. For example, color and shape features are commonly used to represent visual data. Color is fairly irrelevant for *cars*, but a distinguishing feature of *ferns*. The shape of certain objects like *balls* are consistent, whereas the shape of non-rigid objects like *dogs* often changes. This impacts intra-class and inter-class similarity, which determines the success of one-to-one mappings between groups and classifier labels.

4.1.1. Performance of One-to-One Grouping and Labeling

Previous chapters have hinted at the performance of grouping techniques whose only labeling interaction is providing labels for groups. Results found in the literature are quickly summarized, and further analysis is provided to motivate the new perspective on grouping and labeling performed in this thesis. A description of the datasets for these experiments is provided in Appendix A. There are two important things to note about the techniques that we summarize. First, these techniques focus on improving feature representation to create a quality partition of the data. Second, they set m using a priori knowledge of K. Specifically, two techniques [6, 8] learn exactly K groups, while the incremental clustering technique [15] learns closer to a two-to-one mapping between groups and labels. Table 4.1

TABLE 4.1. Label accuracy of existing one-to-one group labeling techniques.

Author	Dataset	K	n	# Interactions	Label Accuracy
Sivic et al. [10]	Caltech-4	4	3,190	4	0.9800
	Caltech-4 + background	5	4,090	5	0.7800
Wards-baseline	13-Scenes	13	2,500	13	0.5000
Dai et al. [6]	15-Scenes	15	$\approx 4,500$	15	0.6149
	35-Compound	35	$\approx 8,200$	35	0.5834
	Caltech-101	101	8,677	101	0.4231
Lee & Grauman [15]	MSRC-v0	21	60,000	40	0.6000
Tuytelaars et al. [8]	MSRC-v2	23	591	23	0.8530

shows the average label accuracy achieved by these methods when assigning the majority label to groups.

Varying degrees of label accuracy can be seen in the table. The Caltech-4 dataset is a simple subset of the larger Caltech-101 object dataset [57] and exhibits relatively low inter-class similarity. It is not surprising that Sivic et al. perform well on this data. However, notice that as soon as the *background* class is introduced, performance drops. For more challenging datasets, label accuracy in the table is low, indicating that the underlying similarity of a group is only weakly categorized as a visual concept from the ground truth set. Label accuracy tends to decrease as the number of visual concepts increases. This may indicate that more visual concepts cause inter-class similarity to increase, making the problem more difficult. It may also indicate that there is more intra-class dissimilarity within the problem. Even when Lee and Grauman relax the one-to-one mapping constraint and learn more groups than concepts, label accuracy remains low.

Tuytelaars et al. [8] present a relatively high label accuracy for a 23 class dataset, but this performance value is somewhat misleading. Of the 23 labels provided in this experiment, only 11 of the ground truth concepts were discovered. For example, 5 different groups were assigned the label *grass* because that was the dominating concept within the group. When

the same visual concept dominates multiple groups in what is supposed to be a one-to-one partition, the possibility of collecting labels for all visual concepts is eliminated. So although the labels collected for the classifier are reasonably accurate, classification performance will suffer because fewer than half of the visual concepts are represented in the labeled training data. While discovery performance cannot be confirmed for the other techniques, we hypothesize that similar results follow because visual property variances cause a decrease in intra-class similarity.

Using observations about visual property variations and the existing experimental evidence, we proceed with the assumption that global input parameters do not exist as a viable solution to learn accurate one-to-one partitions at the granularity of the classifier label set. We instead present experiments to show that groups that appear to lack coherency relative to the classifier label set actually may share a more general visual concept label.

4.1.2. Hierarchical Label Set Demonstration

Current techniques have focused on the accuracy or coherency of a group given a specific predefined label set. The following experiments are designed to analyze group similarities in a more general sense. We show that groups with low label accuracy given a label in $\mathcal{Y}$ actually have a coarser-grained concept that they more accurately represent. We use these observations to motivate our use of hierarchical label sets during the label collection process.

For demonstration purposes we create a 5 class subset of the 13-Scenes [17] dataset that we call 5-Scenes. The five scene classes are *coast, highway, living room, suburb* and *tall building*. We perform k-means on 5-Scenes and set $k = 5$ to imitate a one-to-one partition approach. Table 4.2 summarizes the five groups and their label accuracy given majority labeling. The first four groups have high label accuracy, but discovery is limited to three

TABLE 4.2. Results of k-means clustering where $k = 5$ for 5-Scenes.

Group id	Dominating label	Label accuracy
1	living room	1.000
2	living room	0.992
3	suburb	0.906
4	tall building	0.868
5	coast	0.561

classes since two groups are labeled *living room*. Group 5 has a dominating class label of *coast* but low label accuracy. We evaluate the similarities and feature patterns of this group in more detail to discover a new visual label that more accurately represents the group of images.

Figure 4.1 displays 20 randomly sampled images from group 5. Relative to $\mathcal{Y}$, this group of images mostly represents *coast* and *highway* scenes. If we look at the similarities found within these concepts, one reoccurring concept is the openness of the scenes. An *open* scene is one with a visible horizon, i.e., no large objects such as trees or buildings occlude the horizon. It is not surprising this concept is discovered by the clustering algorithm since GIST [18] features are designed to represent this spatial structure. If the label *open* were applied to this group of images, label accuracy would increase significantly. As the size of $\mathcal{Y}$ increases it is likely that more coarser-grained concepts can be extracted as well. Other coarse-grained examples found in the 13-Scenes dataset were discussed in Chapter 2.

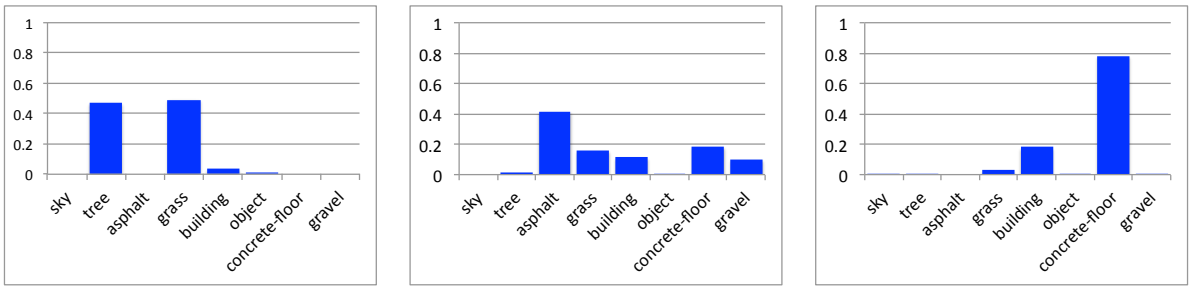
If the end goal is to train classifiers using labels defined in $\mathcal{Y}$, we might simply consider relaxing the one-to-one partition and setting m to be larger than K. In the previous 5-Scenes experiment, it appears that group 5 is composed of two concepts so the number of groups is increased to 6, to allow those two labels to split from one another. Setting $k = 6$ essentially splits group 5 in the original experiment to a group labeled *coast* with label accuracy 0.82

FIGURE 4.1. Randomly sampled images from group 5 of the 5-Scenes dataset k-means clustering.

and a cluster labeled *highway* with label accuracy 0.542, which suggests the label *open* is still more representative of the data.

Again, we see that pre-defining the global parameter m is not a well defined problem. The number of groups needed to perfectly partition the concepts in $\mathcal{Y}$ depends on the intra-class and inter-class similarities. We can also look at a real-world dataset collected for robot navigation (discussed in Chapter 8). The data was collected with the intention of training a classifier to learn eight different terrain and object classes: *asphalt, building, concrete-floor, gravel, object, sky* and *tree*.

We run k-means on this dataset with $k = 20$ and observe that most of groups do not accurately represent one of the eight classifier concepts. Figure 4.2 shows the label distribution of three groups that came from this clustering. Analyzing the distribution of labels in $\mathcal{Y}$ reveals that coarser-grained concepts exist in the data that these groups may more accurately represent. Figure 4.2(a) shows that *grass* and *trees* group well together. The similar color



(a) Visual Concept: *vegetation* (b) Visual Concept: *ground* (c) Visual Concept: *concrete*

FIGURE 4.2. $\mathcal{Y}$ label distribution of groups after k-means where $k = 20$ for real-world robot data.

and texture can be represented as *vegetation*. The concept *ground*, seen in Figure 4.2(b), is the pooling of *asphalt*, *gravel*, *concrete-floor* and *grass*. We also see that terrains and objects made of *concrete* material, *concrete-floor* and *building* group together (Figure 4.2(c)).

These experiments show that even when $m > k$, groups do not accurately represent a single concept in $\mathcal{Y}$. We know that natural language has hierarchical taxonomies. While this research makes no attempt to study the construction of taxonomies in any scope, we use these experiments and examples to demonstrate that patterns found by clustering algorithms are better mapped to a spectrum of visual concept granularities. This includes concepts that are more general or more specific than labels in $\mathcal{Y}$

4.2. Hierarchical Clustering to Facilitate Hierarchical Labeling

Through experiments and analysis we have shown that existing group-based labeling techniques produce significant amounts of label noise because clustering algorithms find feature patterns that do not match the granularity of the classifier label set. Specifically, when a one-to-one partition is used, groups tend to represent a broader concept than the end classifier may want to classify. Increasing the number of groups will naturally break

down the pattern similarities to finer-grained levels, but trying to define a global parameter for data whose similarities are class dependent is an ill-defined problem.

Thus, this thesis moves to a hierarchical clustering approach. Hierarchical clustering produces a tree structure of nested groups where the root contains all data, leaf nodes contain a single data point and internal nodes are non-disjoint subsets of $\mathcal{T}$. We use agglomerative clustering to build the hierarchy bottom-up. Two criteria are needed to build the structure: a distance measure and a linkage criterion. The linkage criterion defines which two groups will be merged at each iteration. We use Ward's linkage [58] which minimizes the resulting intra-cluster variance and Euclidean distance. However, any hierarchical structure formulation could be used in its place.

Maintaining the hierarchical clustering structure introduces some unique properties not seen in most partitional clustering algorithms. First, the number of groups to form does not need to be defined in advance. Second, groups in the hierarchy are non-disjoint since each data sample belongs to a set of clusters along a path from the root to a leaf node. Third, the hierarchical structure maintains multiple levels of similarity between data which likely maps to various granularities of concepts.

The hierarchical structure allows data to break down and group naturally without regard to a specific set of visual labels. By maintaining the hierarchy, we have a spectrum of potential visual concepts represented within the groups. Figure 4.3 illustrates this with a hierarchical clustering of the 5-Scenes dataset used earlier in this chapter. Nodes colored black correspond to groups that contain images from multiple scene classes. The remaining colors indicate groups of images from a single scene class. There is a natural division of the hierarchy into four groups: *tall building* (green), *living room* (blue), *suburb* (yellow) and the

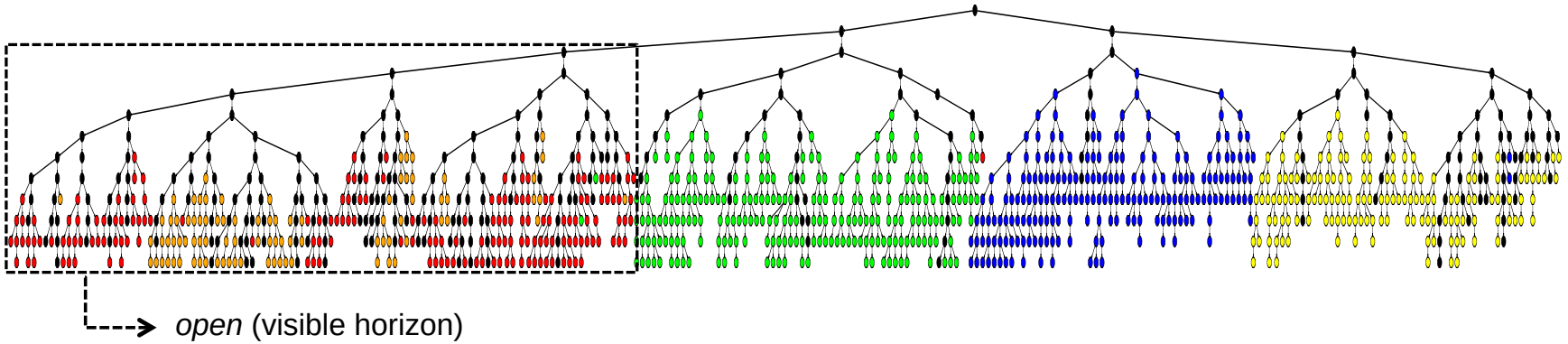


FIGURE 4.3. Hierarchical clustering of five classes from the 13-Scenes dataset.

coarse-grained concept of *open* (dotted outline). The many smaller, inter-weaved partitions of the *coast* (red) and *highway* (orange) classes is evidence of high intra-class similarity.

Although hierarchical clustering does not require the global parameters that many partitional algorithms need, the output is a large structure that contains more groups than data samples. Formally, the hierarchy $\mathcal{H} = \{c_1, c_2, \ldots, c_m\}$ represents a set of $m \approx 2n - 1$ groups. Labeling each group is time consuming and redundant since we know groups are non-disjoint, i.e., children c_l, c_r contain subsets of data from its parent cluster c, $c = c_l \cup c_r$. This redundancy means that not every group in the tree needs to be labeled.

Instead, we select groups from $\mathcal{H}$ that represent possible candidate concepts. In Figure 4.3, this includes groups that represent the five ground truth scene categories, and groups that represent coarser-grained concepts such as *open* or *outside*. Groups are iteratively selected and displayed to a human annotator who assigns the most representative label, coarse or fine-grained, to each group. The label assignment and recognition of the underlying similarity within the group falls on the human annotator since our approach assumes no a priori knowledge or predefined label set. However, we assume that labeling is being performed with a specific application in mind, and that the annotator has a general idea of what labels are necessary for the task. For example, the annotator may know that the data will be used to train classifiers for autonomous navigation. The types of terrain and objects in the training set may not be known in advance, but the annotator knows that labeling specific types of terrain is important because traversable terrain may be platform specific. The next chapter presents the details of our framework, specifically how groups are selected and ordered for iterative labeling by a human annotator.

CHAPTER 5

Hierarchical Cluster Guided Labeling

Our labeling framework is motivated by the findings discussed in previous chapters. First, labeling groups can be more efficient than labeling individual instances. Second, even though group labeling can introduce label noise, the overall impact on classifier performance is small if the amount of noise is minimal. Finally, most group labeling frameworks introduce a significant amount of label noise because they try to force a particular mapping between groups and labels.

This thesis addresses these shortcomings by using hierarchical clustering and assigning labels of the granularity appropriate for the learned groups. We hypothesize that this labeling framework will better balance efficiency and label noise, resulting in fast label collection capable of training high performing visual classifiers.

These design decisions produce a large structure that represents many possible groups to label. The details of our hierarchical cluster guided labeling[5] (HCGL) methodology are discussed in this chapter. We discuss how a meaningful subset of groups can be selected from the hierarchy using the encoded relationships and evaluating structural change.

5.1. Structural Change to Model Concept Transition

Naturally, groups closer to the root of $\mathcal{H}$ represent concepts that are more coarse-grained than their descendants lower in the hierarchy. However, with unlabeled data the level in the hierarchy where groups begin to match the concept granularity found in $\mathcal{Y}$ is unknown. Further, not all concepts follow the same granularity breakdown, as this is dependent on the particular concepts that exist in the dataset and their intra and inter-class similarities.

[5]Material previously published by author in CVPR 2015 [59].

Thus, the goal of HCGL is not to find exactly K groups that represent the set of concepts to be used in the recognition task, but instead to find groups that represent all possible hierarchical visual concepts, $\hat{\mathcal{Y}}$, in the data. This selection technique results in a many-to-one mapping between groups and concepts. Consequently, labeling effort (Equation 1) for HCGL will be greater than one-to-one partitions of data. However, this additional effort is traded for better concept discovery and higher label accuracy, which plays a significant role in classifier learning.

Figure 5.1 is the subtree enclosed in the dotted outline from the hierarchical clustering example in Figure 4.3. Nodes that were previously black now resemble a pie chart that depicts the percentage of images that represent the dominating class label from $\mathcal{Y}$ (red/orange wedge) and all remaining "noisy" images (black wedge). Labeling all three consecutive *open* groups (indicated by the near 50/50 wedge split) down the left most path in the subtree is redundant since descendants of a labeled group naturally inherit the assigned concept. Locating transitions from coarse to fine-grained labels establishes a management subset of groups, $\mathcal{S} \subset \mathcal{H}$, that can be labeled efficiently.

To find $\mathcal{S}$, HCGL searches for interesting locations in $\mathcal{H}$ to label, where ***interestingness*** is defined as the degree of change at a split in $\mathcal{H}$. Specially, we model structural change between feature patterns of two groups to measure interestingness. The idea is that feature similarities encoded in $\mathcal{H}$ map to coarse and fine-grained visual concepts. When the underlying pattern of similarity changes, it is more likely that a visual concept transition has also occurred.

HCGL compares the structural change between a cluster, c, and its parent, p. The internal structure of a cluster c is derived from its data matrix, X_c, where each column is an

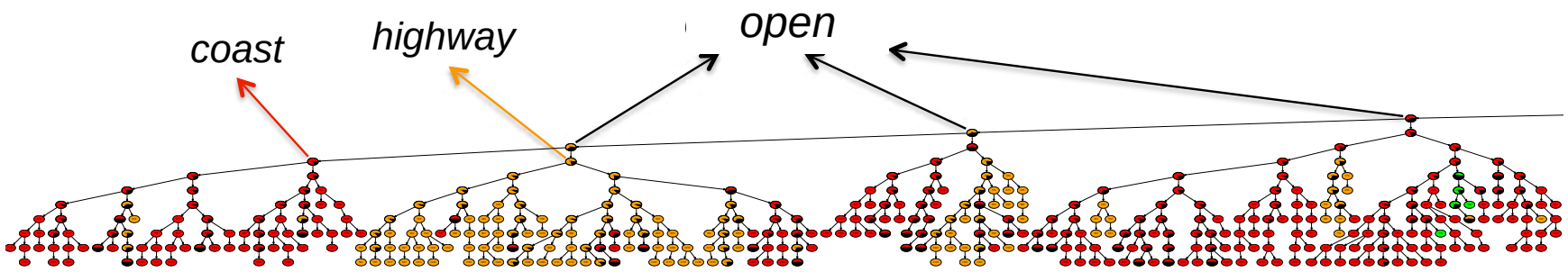


FIGURE 5.1. Subtree that represents the concept *open* and then transitions to *coast* and *highway*.

image represented by a d-dimensional feature vector:

$$X_c = \begin{pmatrix} x_{1,1} & x_{2,1} & \cdots & x_{s,1} \\ x_{1,2} & x_{2,2} & \cdots & x_{s,2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1,d} & x_{2,d} & \cdots & x_{s,d} \end{pmatrix}$$

The data is mean centered, $\hat{X}_c = X_c - \bar{X}_c$, and the covariance matrix of $\hat{X}_c$ is decomposed and represented by its eigenvectors V_c using singular value decomposition:

$$\begin{aligned} V_c \, \Lambda_c \, V_c^{-1} &= SVD(Cov(\hat{X}_c)) \\ &= SVD\left(\frac{\hat{X}_c \hat{X}_c^{\,T}}{s-1}\right) \end{aligned}$$

This representation of internal structure focuses on the direction of variance in the data. Given that the diagonal entries of Λ_c are sorted in descending order, the first eigenvector,v_{c1}, in V_c provides the axes of maximum variance for c.

Local structural change is found by comparing the internal structure of c to its parent p (this relationship can be seen in Figure 5.2). Specifically, the comparison is made by calculating the angle between the first eigenvectors, v_{c1} and v_{p1}, of c and p respectively. Larger angles indicate greater differences in directions of variance, and are used to represent the interestingness of each split in $\mathcal{H}$. Formally, interestingness derived from structural change for group c is defined as the cosine distance,

$$\Delta(c) = 1.0 - \langle v_{c1}, v_{p1} \rangle, \tag{2}$$

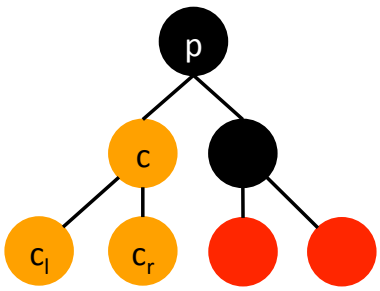


FIGURE 5.2. Illustration that depicts the relationships for group c in a local neighborhood of $\mathcal{H}$, including its parent p and left and right children, c_l and c_r.

which yields values on the interval of $[0.0, 1.0]$ with large Δ values representing large angles.

Most groups in $\mathcal{H}$ have at least some structural difference from their parent, but notice in Figure 5.1 that it often takes several splits before a change in concept occurs. To better detect these transitions, HCGL looks for large changes in structure followed by a lack of structural change in local neighborhoods of $\mathcal{H}$. In other words, if the structural change of c is a local maximum with respect to p and its children, c_r and c_l (relationship illustrated in Figure 5.2), it is added to $\mathcal{S}$. Formally, local maxima selection is defined as

$$\mathcal{S} = \{c \,|\, \Delta(c) > \Delta(p)\,,\, \Delta(c) > \Delta(c_{r,l})\}. \tag{3}$$

$\mathcal{S}$ has two important properties. First, groups in $\mathcal{S}$ are not necessarily disjoint because every image belongs to many related groups in the hierarchical structure. Second, selecting peaks in structural change does not guarantee that every image will be represented in $\mathcal{S}$. We will discuss the first property in context with the group labeling order. The second property may result in only a fraction of the training data being assigned labels, which is analyzed in the experimentation.

5.2. Group Labeling Order

Partitional grouping approaches form a set of disjoint groups, label each group, and then train a classifier with the collected data. HCGL is different because groups in $\mathcal{S}$ are not necessarily disjoint. The ordering of groups in $\mathcal{S}$ is meaningful because if a group is given a class label from $\mathcal{Y}$, all descendants of this group (according to the structure in $\mathcal{H}$) inherit that label, and thus no longer need to be labeled by an annotator.

There are many ways $\mathcal{S}$ can be ordered. Since groups have already been selected as meaningful based on their interestingness score (Equation 2), by default HCGL ranks groups in descending order by their Δ value. The idea is to order groups by strength of their potential for concept transition. During labeling, when a group is given a class label from $\mathcal{Y}$, any descendants that inherit the label and exist in $\mathcal{S}$ are removed. Thus, the total labeling effort of HCGL is not equal to $|\mathcal{S}|$, but depends on the labeling order and the number of inherited labels. We will discuss variants of this labeling order in our experiments.

The algorithm outline of HCGL can be found in Algorithm 1. We now present a set of experiments to demonstrate the potential of this thesis research. Later we discuss future work that expands upon the currently described model.

Algorithm 1 Hierarchical Cluster Guided Labeling

Require: $\mathcal{H}$
1: $S = \{\}$
2: **for all** $c \in \mathcal{H}$ **do**
3: $\quad relatives = \{p, c_l, c_r\}$
4: $\quad$ **if** $\Delta(c) > \Delta(r)$, $\forall\ r \in relatives$ **then**
5: $\quad\quad \mathcal{S} = \mathcal{S} \cup \{c_i\}$
6: $\mathcal{S} = sort(\mathcal{S}, \Delta)$
7: **while** $\mathcal{S} \neq \emptyset$ **do**
8: $\quad$ label query $\rightarrow \mathcal{S}[0]$
9: $\quad update(\mathcal{S})$

5.3. Evaluation

5.3.1. Experiment Setup

We compare HCGL to several state of the art group-based labeling techniques on two benchmark datasets, 13-Scenes and MSRC. In these experiments we test our hypothesis that moving to hierarchical clustering, evaluating hierarchical relationships and assigning labels from a hierarchical label set allows HCGL to discovery underlying concepts and balance labeling efficiency and label noise to collect labeled data that trains higher performing visual classifiers than existing techniques. These experiments are performed to show the effect of labeling effort, our independent variable, on multiple dependent variables: classification accuracy, discovery rate, labeling rate and label accuracy.

Each experiment is averaged over 10 trials of random training/testing partitions. The specific partitions for each dataset are discussed in later sections. Error bars in figures show the standard deviation of the results across the 10 experiment trials. Any significance testing is performed using the Wilcoxon signed-rank test.

Comparisons to HCGL are made using a set of diverse labeling frameworks that require different types of labeling interactions. These methods include:

- **SAC** (Spectral Active Clustering [12]) - active clustering approach that queries for 20 binary constraints per iteration to improve one-to-one clustered output, followed by majority labeling
- **SG** (Selective Guidance) - our initial hierarchical clustering approach that models group coherency for selection and only labels groups of images that represent a single class

- **MKML** (Multiple Kernel Metric Learning [16]) - iterative clustering approach that labels the largest subset of samples with the dominating class and removes images that do not match this label

We note that SG was previously shown to outperform partitional clustering and active learning techniques [51], so any performance improvements that HCGL displays over SG can also be inferred as performance improvements to these other techniques.

The label assignment for HCGL is automated in these experiments by simulating human labeling responses using the ground truth associated with each dataset. However, these datasets do not have hierarchical ground truth associated with them. We derive a small set of hierarchical labels for the 13-Scenes dataset, but only evaluate classification accuracy relative to the classifier label set $\mathcal{Y}$. We simulate label assignment in the following way. If more than 50% of a group's images represent a visual concept in $\mathcal{Y}$, it is given the majority class label. When a group does not have a majority of images that map to a label in $\mathcal{Y}$ it assigned the most relevant coarser hierarchical concept when this information is available. If a hierarchical concept does not make up the majority of images or no hierarchical label set is available the group is assigned the label *irrelevant*. As with SG, this query counts towards the total level of effort, but provides no new label information to the system. We discuss how hierarchical labels are used in a later chapter.

Since it is easy to simulate human interactions automatically with benchmark datasets, labeling systems can be run to completion even if the total effort cannot be supplied during real-wold applications. For this reason, we focus our evaluation on the performance achieved in the earlier stages of labeling effort. This allows us to evaluate how well systems perform when labeling resources are scarce and fully labeling the data may not be feasible.

5.3.2. Scene Labeling and Classification

The first experiment is performed using the 13-Scenes dataset [17]. Each experiment trial is derived by a random partition of the training data where 80% is used as training and the remaining 20% is used for testing the trained classifier. For classification, we train an SVM classifier using the same parameters found in the existing efficient labeling literature [5]. Using publicly available code, we directly compare HCGL to SAC and SG using the same 10 trial splits.

Classification accuracy with respect to labeling effort for the 13-Scenes dataset can be seen in Figure 5.3. The most glaring distinction is the high labeling effort required by SAC before a large improvement in classification accuracy is seen. In fact, SAC exceeds the effort required of a fully supervised instance-based labeling approach, indicated by the vertical dashed line at labeling effort equal to 1.0. Recall, however, that SAC queries for binary constraints which provide a single bit of information. Thus, it is not surprising that many binary interactions are required to collect sufficient information to drastically refine the partitioned output.

The classification performance of HCGL and SG with respect to labeling effort is a much closer comparison. HCGL appears to outperform SG when averaging all experiment trials, but the high standard deviation in performance seen in the far left of the plot suggests that the two techniques are not significantly different during the initial queries. Table 5.1 includes the p-values from the Wilcoxon signed-rank test performed on the classification results of HCGL and SG at various labeling efforts. We display the results for the first 25 labeling interactions and every 25th interaction following. The level of significance is evaluated at $\alpha = 0.01$, and the p-values confirm that results are not significantly different early in the

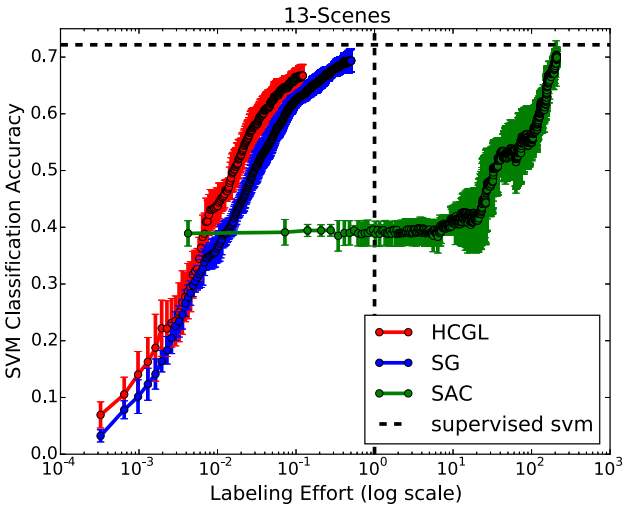


FIGURE 5.3. Comparison of classification accuracy versus labeling effort on the 13-Scenes dataset.

labeling process. However, classification results achieved after labeling effort equal to 0.0068 are significantly different between HCGL and SG. This significance shows the performance improvements that HCGL provides over these existing techniques.

Overall, Figure 5.3 shows that with only one-tenth of the fully supervised labeling effort, HCGL trains classifiers that approach supervised performance (indicated by the dashed horizontal line). SG also approaches supervised performance and eventually exceeds the classification performance of HCGL, but does so with up to three times more labeling effort.

The MKML code was not made available by the authors, but we simulate their largest subset labeling on this dataset with the HCGL framework. For every group selected by HCGL, a dominating class label was provided followed by the removal of label noise. Figure 5.4 shows the additional effort required by largest subset labeling, denoted as HCGL-largest subet, and the classification gap between it and our majority labeling approach, HCGL-majority.

As was seen in Chapter 3, the addition of label noise has very little impact on the overall classification accuracy. We see minimal improvement in overall classification accuracy using

TABLE 5.1. P-values after running Wilcoxon signed-rank test to compare the significance between classification results achieved by HCGL and SG at various levels of labeling effort.

Labeling Effort	p-value
0.0003	0.0098
0.0006	0.0499
0.0010	0.0926
0.0013	0.0593
0.0016	0.0926
0.0019	0.0166
0.0023	0.0367
0.0026	0.0469
0.0029	0.3329
0.0032	0.1688
0.0036	0.1688
0.0039	0.2845
0.0042	0.3329
0.0045	0.2845
0.0049	0.1688
0.0052	0.1141
0.0055	0.1141
0.0058	0.0469
0.0062	0.0093
0.0065	0.0125
0.0068	0.0069
0.0071	0.0051
0.0075	0.0069
0.0078	0.0051
0.0081	0.0051
...	
0.0162	0.0051
0.0243	0.0051
0.0324	0.0051
0.0405	0.0051
0.0486	0.0051
0.0567	0.0051
0.0648	0.0051
0.0729	0.0051
0.0810	0.0051
0.0891	0.0051
0.0972	0.0051
0.1053	0.0051
0.1134	0.0069

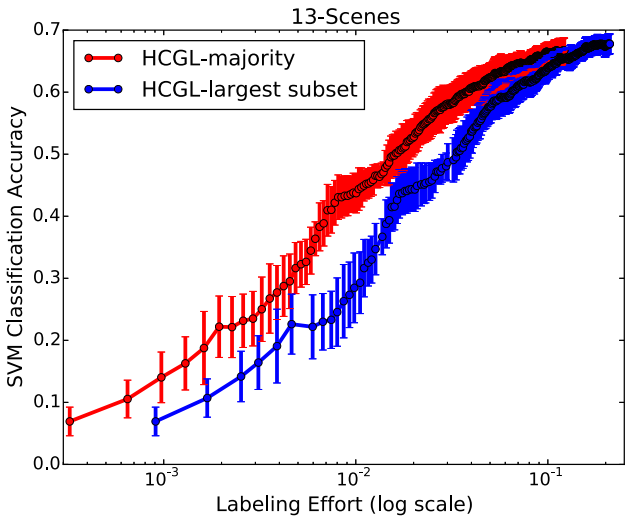


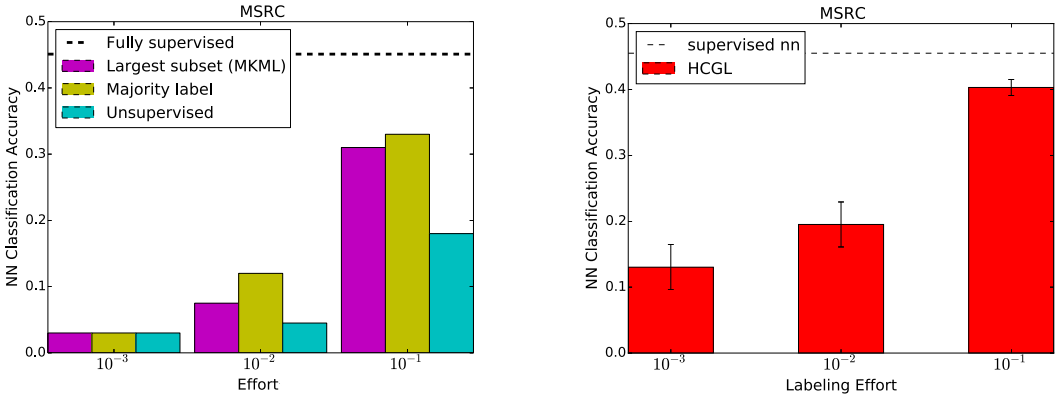
FIGURE 5.4. Largest subset versus majority labeling on 13-Scenes dataset.

noise-less labels for more labeling effort. This continues to support the fact that the addition of minimal (to be seen in Figure 5.6(b)) label noise does not negatively impact the end classifier in a severe manner.

5.3.3. Object Labeling and Classification

Our second experiment replicates the experimental protocol used in the MKML paper [16] on the MSRC-v2 dataset. In the original experiment, the authors use a 40/60 data partition. The 40% split was used to extract regions and features representing 5 classes that were presumed known to act as a seed to their system. HCGL assumes no known knowledge while collecting labels and therefore does not use this data for seeding. The other 60% is used to perform the grouping and label collection for 16 unknown classes. Using the collected labels, classification is performed only on the 16 classes that were presumed unknown, which means the 40% split can also be used at the testing set.

The multi-concept MSRC images are segmented into regions using the publicly available segmentation and appearance feature extraction code used by Lee and Grauman [7]. While



(a) Extracted MKML results

(b) Corresponding HCGL results

FIGURE 5.5. Comparison of classification accuracy for three degrees of minimal labeling effort on the MSRC dataset.

the image region set is not identical to the set used by MKML, we achieve the same supervised nearest neighbor classification performance as MKML, indicating that the sets of training data are effectively equivalent.

A comparison between HCGL and MKML can be seen in Figure 5.5. The MKML results, Figure 5.5(a), are an alternative view of the authors' original presentation (Figure 10 [16]). The three bars correspond to their proposed largest subset labeling technique, a majority labeling technique intended to emulate an incremental labeling system [15] and an unsupervised baseline. Further, as mentioned earlier the focus of comparison is on the classification results achieved during the earliest stages of labeling effort.

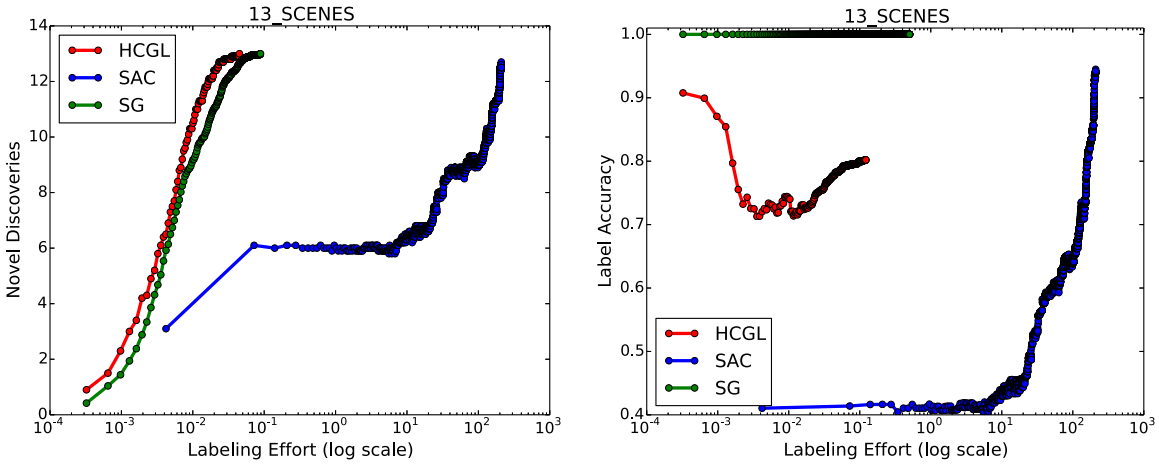
The results are separated in side by side plots because the definition of labeling effort used by MKML is slightly different than what is defined in Section 3.1. In particular, MKML defines effort as the fraction of images that are removed from a group because they do not match the largest subset label. It does not include the effort required to provide the label of the largest subset.

When focusing on results achieved with minimal labeling effort, the majority labeling technique in Figure 5.5(a) outperforms MKML with largest subset labeling, which is noted by the authors. However, the majority labeling technique still performs significantly worse than the fully supervised approach. Our approach uses a similar majority labeling scheme, but outperforms all techniques from the MKML paper. The performance gap suggests that groups in $\mathcal{S}$ are more coherent than those selected in the MKML approach. More label noise requires more effort by MKML before it can achieve reasonable classification performance.

5.3.4. Evaluation of Problem Objectives

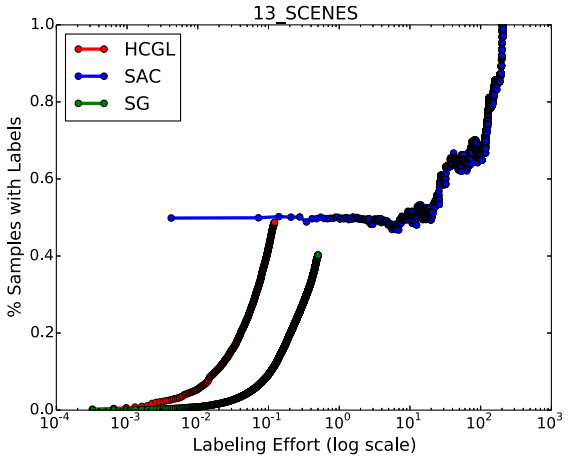
Performance on the objectives defined at the beginning of this thesis ultimately influences classifier learning. To break down the overall performance we perform a secondary evaluation that looks at the discovery rate, labeling rate and label accuracy with respect to labeling effort. Figure 5.6 compares results of HCGL, SAC and SG on these three objectives using the 13-Scenes dataset.

Concept discovery is important since classifiers can only recognize concepts that exist in the labeled training data. Techniques that collect labels for all K classes the fastest will likely see the fastest classification performance boost. Figure 5.6(a) shows that HCGL has the best rate of discovery with SG performing competitively in the first several iterations of effort until the discovery gap widens. The poor performance of SAC can be attributed to two things: 1) groups may be too incoherent to label or 2) the same scene class may be dominating multiple groups. Each time this occurs in a one-to-one mapping, one class goes undiscovered. Notice that the rate of discovery and classification performance (Figure 5.3) have similar trends, reinforcing the claim that discovery is of significant importance for bottom-up grouping techniques.



(a) Rate of discovery

(b) Label accuracy



(c) Percentage of samples labeled

FIGURE 5.6. Comparison of label collection objective performance for HCGL, SG and SAC on the 13-scenes dataset.

The label accuracy maintained by all the labeling techniques can be seen in Figure 5.6(b). As mentioned previously, many labeling techniques have emerged that focus on how to collect accurate labels even at the cost of more effort. SG enforces this constraint by only assigning labels to groups that consist of images from exactly one class, and although SAC never reaches perfectly accurate labels, the addition of binary constraints is intended to allow a clustering algorithm to eventually perfectly partition the data by class. On average, HCGL

maintains accurate labels for about 70-80% of all labeled samples which does not appear to negatively impact classification performance significantly.

HCGL takes the middle ground between perfect label accuracy, like SG, and poor label accuracy seen in one-to-one mapping, like SAC. Also notice that HCGL has a faster labeling rate than SG, but slower than SAC in Figure 5.6(c). This shows the balance of label accuracy and labeling efficiency that HCGL was designed on. Most importantly, this balance allows HCGL to train higher performing classifiers than the other techniques that focus only on either label accuracy or labeling efficiency.

Finally, Figure 5.6(c) also reinforces that labeling all of $\mathcal{S}$ does not guarantee that HCGL assigns a label to all training samples. This is a similar feature of SG as it also does not label all training samples even though it is designed to run until all data are labeled. This indicates that many SG queries result in a "mixed" label where no labeled samples are collected. Overall, these experiments illustrate the importance of discovery, and how the balance of label accuracy and labeling efficiency may be more important than emphasizing any single criterion.

CHAPTER 6

Selection Ordering

The initial HCGL implementation of cluster selection emphasized interestingness (defined in Chapter 5. This was done by identifying groups in $\mathcal{H}$ whose structural change value was a local maximum, and labeling these groups in order of this change, largest first. This was shown to perform well against other state-of-the-art labeling techniques, but does not explicitly address the label collection objectives defined at the beginning of this thesis. Further, while using local maxima helps narrow down the number of groups to label, its selections do not necessarily select groups with the highest interestingness scores. In this chapter we present and compare new selection and ordering criteria for HCGL that more explicitly addresses the problem objectives.

6.1. Local Maxima Inconsistencies

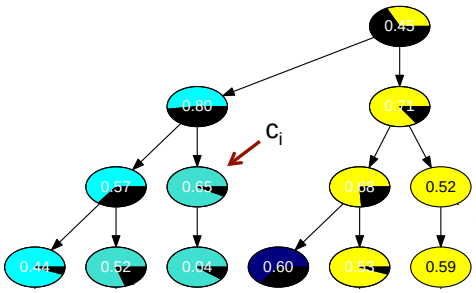
In the previous chapter, Equation 3 constructs $\mathcal{S}$ by identifying the subset of clusters whose interestingness scores are local maxima in the hierarchy. However, recall that the interestingness score was designed to locate large changes in pattern structure to indicate concept transitions. Unfortunately, selecting only local maxima has two inconsistencies with this original design goal. These inconsistencies are illustrated in Figure 6.1, which shows two subtrees from a hierarchy constructed from the Digits dataset. Node colors map to the dominating digit class of the group's data, and black wedges indicate the percentage of samples that do not represent the dominating digit class. The interestingness score of each group is labeled in the node.

First, not all groups with high interestingness scores are local maxima, meaning some locations with potential for concept transition may never be displayed and labeled by a human annotator. Cluster c_i (denoted with an arrow) in Figure 6.1(a) is an example of a group with a high interestingness score that is not a local maximum. This group consists mostly of images of the digit *1*, and has high structural change from its parent which consists mostly of images of the digits *1* and *8*. However, it is the parent of c_i that is the local maximum in this neighborhood. In this case, the local maximum is seen after a split from a group of three classes (the root of the subtree) to a group of two classes (the parent of c_i). Consequently, this means that even though c_i also undergoes high structural change from its parent, HCGL will never select this group to be labeled by the human annotator.

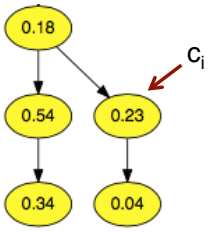
Second, local maxima are not guaranteed to have high interestingness scores compared to other groups in the hierarchy. This is inconsistent with the underlying goal of HCGL since it allows clusters that have low likelihood of providing new label information to be selected and labeled by an annotator. Cluster c_i in Figure 6.1(b) is a local maximum and an example of this type of inconsistency. Relative to other interestingness scores in $\mathcal{H}$, c_i does not indicate that a large structural change has occurred.

Finally, as mentioned in the previous chapter, labeling only local maxima does not guarantee that the entire training set will be labeled. In some cases the entire dataset may not need to be labeled or may not be feasible to label it in its entirety for certain applications. However, if not enough local maxima exist in the hierarchy then classification performance could be significantly compromised because of insufficient label collection.

To address these inconsistencies, selection of local maxima (defined in Equation 3) is dropped and $\mathcal{S}$ is constructed to contain clusters with the highest structural change values as



(a) Example of a cluster with high structural change, i.e., interestingness, that is not a local maximum.



(b) Example of a cluster with low structural change that is a local maximum.

FIGURE 6.1. Illustrations of inconsistencies in local maxima selection. The examples show that not all clusters with high interestingness values are in fact local maxima, whereas some clusters with low interestingness scores are local maxima.

defined in Equation 2. If $\mathcal{S}$ is sorted and labeled in order of these interestingness scores, then all clusters can be added to $\mathcal{S}$. However, later in this chapter we discuss and compare different ordering criteria that should operate only on a subset of the most interesting unlabeled clusters from $\mathcal{H}$.

To identify a subset of clusters, the mean and standard deviation of structural change values of all unlabeled clusters in $\mathcal{H}$ are found:

$$\bar{\Delta} = \frac{1}{|\mathcal{U}|} \sum_{\forall c_i \in \mathcal{U}} \Delta(c_i) \tag{4}$$

$$\sigma_\Delta = \sqrt{\frac{1}{|\mathcal{U}|} \sum_{\forall c_i \in \mathcal{U}} (\Delta(c_i) - \bar{\Delta})^2} \tag{5}$$

We refer to clusters with structural change values at least one standard deviation beyond the mean as outliers and they are added to $\mathcal{S}$:

$$\mathcal{S} = \{c_i \,|\, \Delta(c_i) > \bar{\Delta} + \sigma_\Delta\}. \tag{6}$$

The contents of $\mathcal{S}$ are iteratively updated after each labeling query. Labeled groups are removed, $\bar{\Delta}$ and σ_{Δ} are recomputed, and any new clusters that fall beyond this threshold are added to $\mathcal{S}$.

We do not make any direct comparisons between local maxima and outlier selection because ultimately their constructions of $\mathcal{S}$ capture similar information and achieve similar performance with minimal human effort. However, the two criteria are used in the next two sections, and it is clear that outlier selection allows for a larger percentage of $\mathcal{T}$ to be labeled given no labeling time constraints.

6.2. Other Group Labeling Order Criteria

This thesis identified four relevant objectives for the label collection problem: concept discovery, labeling efficiency, sufficient label collection and label accuracy. Evaluation of the problem objectives performed in the previous chapter showed that HCGL balanced efficiency and accuracy better than existing methods, but the initial selection and ordering criteria do not explicitly emphasize all objectives. Additional ordering techniques are introduced to evaluate the importance of each individual objective. Ordering is based on the following three heuristic criteria:

(1) Interestingness - the degree of structural change seen after a split

(2) Exploitation - the number of samples that would receive labels

(3) Exploration - the likelihood a group is different from those previously labeled

Interestingness is the original ordering from Equation 2 that ranks groups by their likelihood of visual concept transition.

Exploitation ordering is based on the number of unlabeled samples in a cluster. This criterion is designed to label larger clusters first to emphasize the efficiency objective and collect

a large set of labels very quickly. Note that this value is not the size of a cluster because any descendants that belong to the set of previously labeled clusters, $\mathcal{L} = \{(c_1, y_1), (c_2, y_2), \dots\}$, have already assigned some samples a label. Formally, the exploitation score for c_i is:

$$\text{exploitation}(c_i) = |c_i| - |\{x_i \,|\, x_i \in c_j \,,\, c_j \in \mathcal{L} \,,\, c_j \subset c_i\}| \tag{7}$$

Exploration ordering spreads group labeling throughout $\mathcal{H}$ to better explore the feature space as a way to discover groups that represent concepts yet to be labeled. Exploration values iteratively change throughout the labeling process as they are computed with respect to $\mathcal{L}$. Specifically, the exploration score for cluster c_i is the shortest path in the hierarchy between it and all labeled clusters,

$$\text{exploration}(c_i) = \min_{c_j \in \mathcal{L}} \text{path-length}(c_i, c_j), \tag{8}$$

where path length between c_i and c_j is the combined number of direct connections up the tree until their first common ancestor is reached. For example, c_l and c_r in Figure 5.2 have a path length of two where their first common ancestor is c. Exploration ordering labels clusters with the longest path length first, i.e., groups that are least similar to what has already been discovered and labeled.

6.2.1. Experiment Setup

To compare the three heuristic criteria we run HCGL as defined in Algorithm 1 from Chapter 5, with interestingness, exploitation and exploration ordering of $\mathcal{S}$ (line six of the algorithm). We use the same experimental protocol from Chapter 5 with the 13-Scenes and MSRC dataset. The comparison results display the average performance after running 10

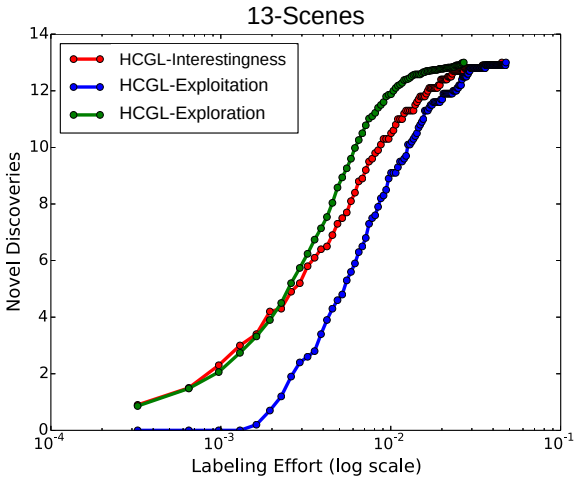
trials of random training/testing partitions. Error bars denote the standard deviation of the results across the 10 trials. Again, we use the Wilcoxon signed-rank test to determine significance at the level of $\alpha = 0.01$. Since we are comparing multiple different techniques and the Wilcoxon test compares pairs of techniques, we run significance testing for pair of label orderings used in HCGL.

We look at three dependent variables, discovery rate, labeling rate and label accuracy with our independent variable labeling effort. These dependent variables are the motivation of our additional objective ordering criteria. Thus, we hypothesize that our evaluation will show that each objective ordering criterion will excel, with respect to the other criteria, in one but not all of the dependent variables being measured.

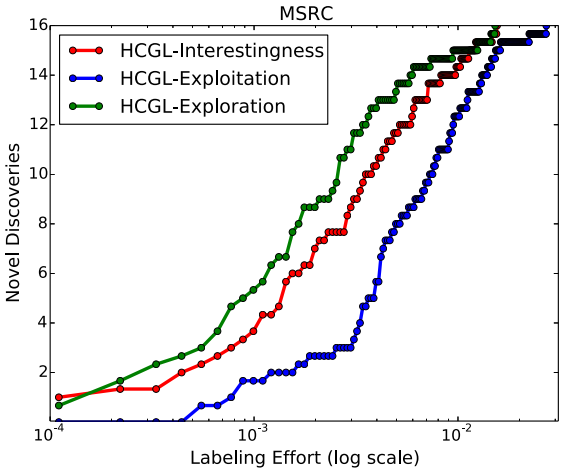
6.2.2. Comparison of Ordering Criteria

Rate of discovery, labeling rate and label accuracy are evaluated on the 13-Scenes and MSRC datasets as a function of labeling effort. This comparison illustrates the emphasis of each of the ordering techniques. As designed, Figures 6.2(a) and (b) show that exploration ordering provides the best rate of discovery, followed by the interestingness ordering. Figures 6.2(c) and (d)) show that exploitation ordering produces more labeled samples faster than other orderings after it gets past its initial selections. These first several selections produce no labeled training data because the groups represent concepts coarser than those labels in $\mathcal{Y}$. However, exploitation ordering does result in lower label accuracy than the other techniques throughout much of the labeling process to achieve its labeling rate dominance seen in Figures 6.2(e) and (f).

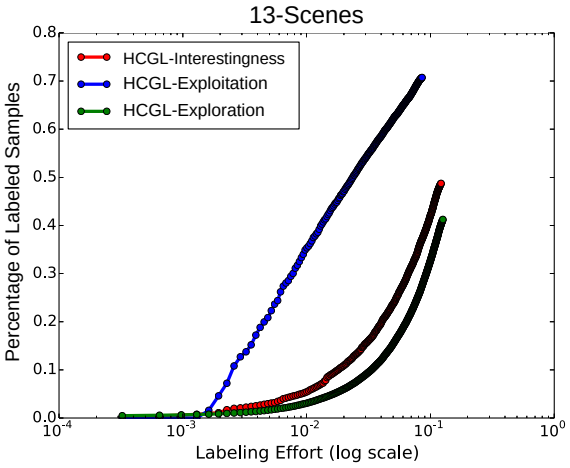
These orderings emphasize different objectives of the label collection process, and Figure 6.3 shows the impact of this emphasis in terms of classifier learning on two datasets.



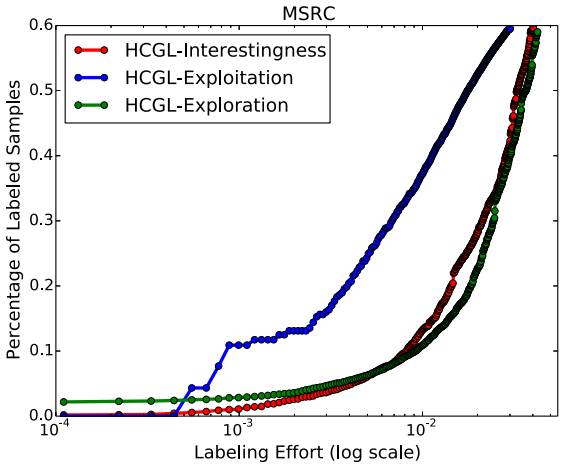
(a) Discovery rate: 13-Scenes



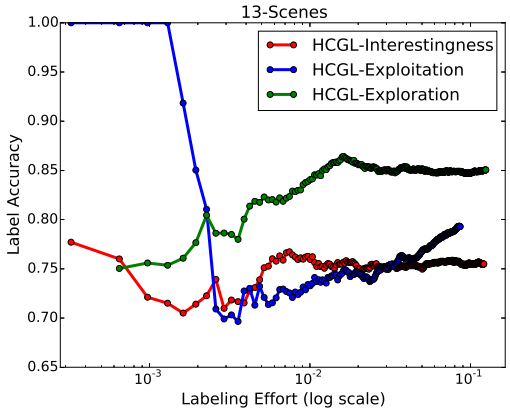
(b) Discovery rate: MSRC



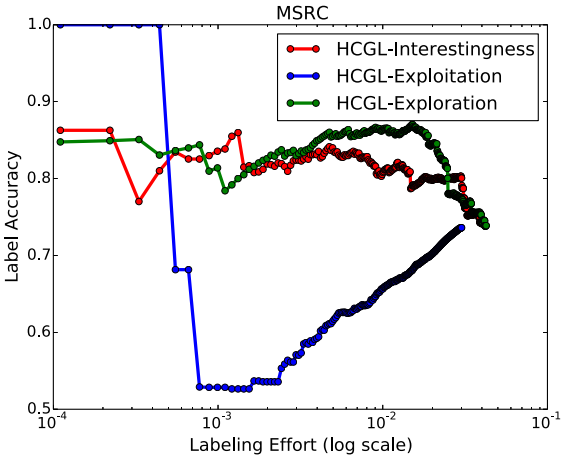
(c) Labeling rate: 13-Scenes



(d) Labeling rate: MSRC

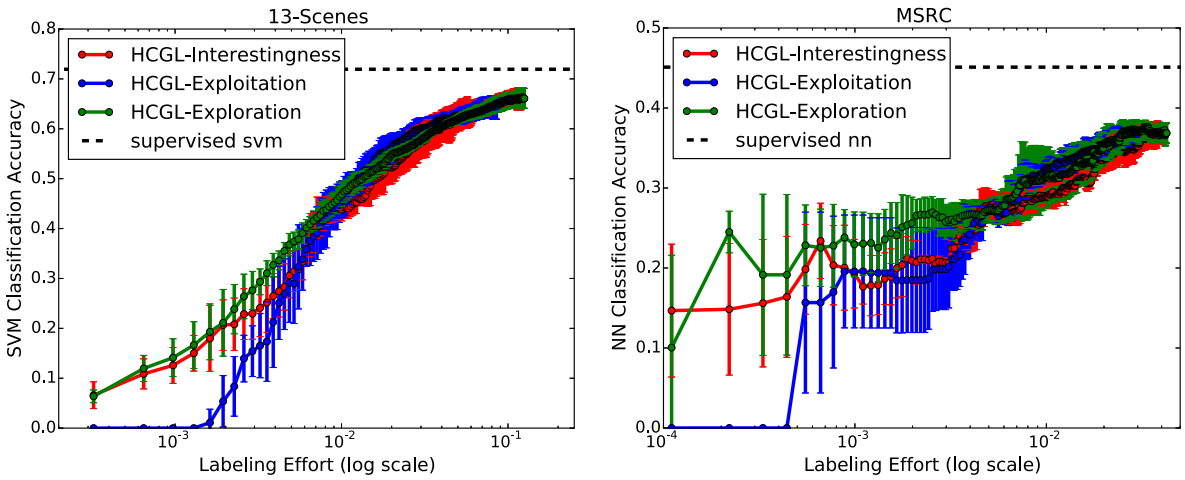


(e) Label accuracy: 13-Scenes



(f) Label accuracy: MSRC

FIGURE 6.2. Comparison of discovery rate, labeling rate and label accuracy for three labeling objective orderings.



(a) Classification Accuracy: 13-Scenes

(b) Classification Accuracy: MSRC

FIGURE 6.3. Comparison of classification accuracy as a function of labeling effort for three labeling objective orderings of $\mathcal{S}$.

There is no single labeling order that significantly outperforms the other two orderings throughout the entire labeling process. However, Table 6.1 shows the significant differences in classification on the 13-Scenes dataset (Figure 6.3(a)) between each pair of labeling orders based on the p-values computed after running the Wilcoxon signed-rank test. Significant differences in classification occur at four different times during the labeling process. The most obvious significant difference seen in Figure 6.3 for both datasets occurs in the earliest stages of labeling. Interestingness and exploration significantly outperform exploitation, which initially is unable to classify any of the testing set because it selects large groups representing concepts too coarse to be assigned a label from $\mathcal{Y}$. P-values less than 0.01 are also seen on the MSRC dataset through the first 30 labeling interactions (~ 0.003 labeling effort). Thus, quickly labeling all visual concepts is extremely important even if this results in a much slower labeling rate, as in the exploration and interestingness orderings.

After around 1% labeling effort, exploitation ordering discovers nearly all concepts in the dataset, and starts to classify more similarly to the other techniques. This is around the same

time that there is a significant performance gap between the interestingness and exploration ordering. Eventually, the large number of samples labeled by the exploitation ordering criterion enhances classifier learning and displays significant improvements over exploration ordering around 2% labeling effort. Overall, these significant differences are short lived during the labeling process. These results suggest that initially focusing on discovery of all visual concepts is particularly important, but that a small set of accurately labeled data and a larger set of less accurately labeled data train similar performing classifiers.

6.2.3. Combined Ranking Comparison

Each ordering criteria excelled at different problem objectives which benefited overall classification performance at various times throughout the labeling process. To balance the benefits of all ordering criteria, a multi-objective rank combination ordering is defined.

Clusters in $\mathcal{S}$ are ranked according to each criteria individually, and linearly combined to produce the multi-objective ordering score,

$$\begin{aligned} \text{rank-score}(c_i) = \beta_1 \, &\text{ranking}(\text{exploitation}(c_i), \{\text{exploitation}(c_j) \,|\, c_j \in \mathcal{S}\}) \\ + \beta_2 \, &\text{ranking}(\text{exploration}(c_i), \{\text{exploration}(c_j) \,|\, c_j \in \mathcal{S}\}) \\ + \beta_3 \, &\text{ranking}(\Delta(c_i), \{\Delta(c_j) \,|\, c_j \in \mathcal{S}\}), \end{aligned} \tag{9}$$

where each β_i is a weight for its ordering objective such that $\beta_1 + \beta_2 + \beta_3 = 1.0$. For all experiments in this section, the objectives are weighted evenly, $\beta_1 = \beta_2 = \beta_3 = \frac{1}{3}$. The ranking function is passed the cluster's score and the set of scores from clusters in $\mathcal{S}$ eligible to be labeled, and returns the cluster's rank with respect to the set. The cluster with the highest rank score is selected as the next labeling query.

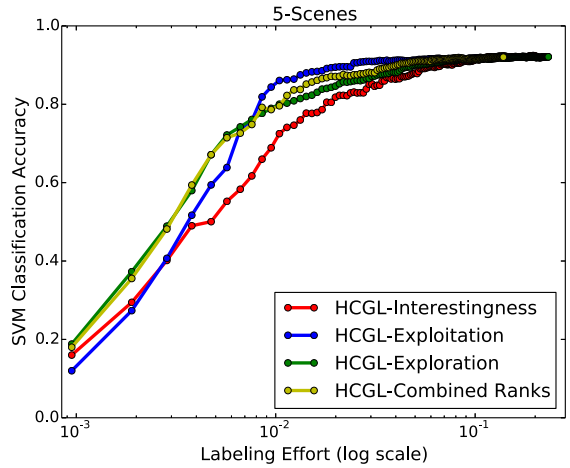
We compare this ranked-based combination ordering with the three criteria used individually. Comparisons are made using the new outlier selection criterion instead of local maxima. As mentioned earlier, classification results are not noticeably different after this change, but the percentage of labeled samples increases because more labeling queries can be issued to the annotator. Algorithm 2 outlines the procedure of HCGL when using the outlier selection criterion with any labeling order criteria.

Algorithm 2 Hierarchical Cluster Guided Labeling-v2

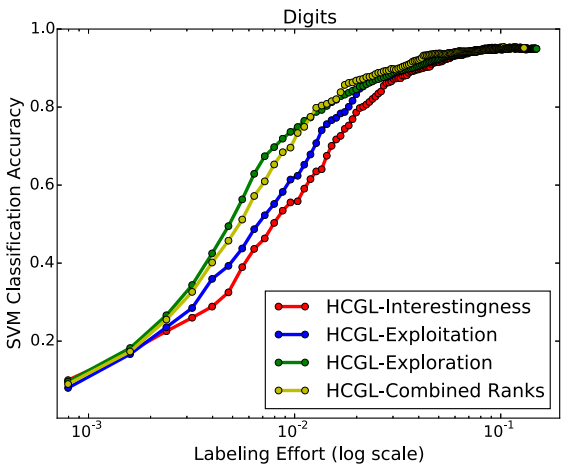
Require: $\mathcal{H}$
1: $\mathcal{U} = \{c_i \,|\, c_i \in \mathcal{H}\}$
2: **while** $\mathcal{U} \neq \emptyset$ **do**
3: threshold $= \bar{\Delta} + \sigma_\Delta$
4: $\mathcal{S} = \{c_i \,|\, \Delta(c_i) >$ threshold, $c_i \in U\}$
5: *update*(ordering-scores)
6: $\mathcal{S} = sort(\mathcal{S}$, ordering-scores)
7: label query $\rightarrow \mathcal{S}[0]$
8: *update*($\mathcal{U}$)

Figure 6.4 shows classification results for the individual objective criteria and the multi-objective combination criteria for several datasets. Error bars are omitted from these figures for easier viewing, but a discussion on significant differences is provided in their place. Overall, the multi-objective combination typically performs as well or better than the individual orderings with the exception of the 13-Scenes experiment (seen in Figure 6.4(c)). Early in the labeling process the combined rank ordering tends to favor the exploitation objective which has the worst performance of all the techniques for this dataset.

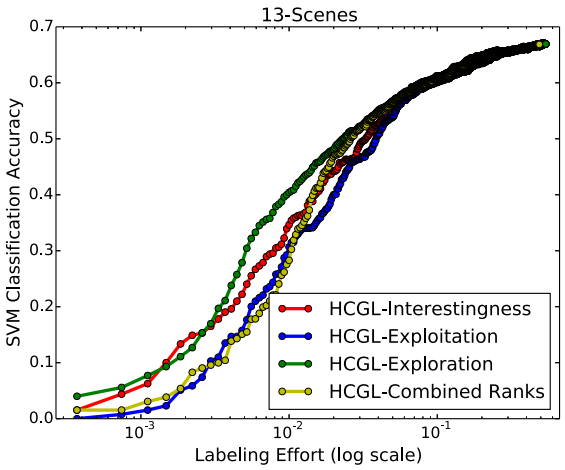
The most interesting and important result can be seen on the MSRC dataset in Figure 6.4(d). In this case, the multi-objective combination ordering criterion improves classification and outperforms the criteria individually during most of the labeling process. The significance of these results are show in Figure 6.5, which shows the p-values when comparing the classification accuracy of the multi-objective labeling combination with the three



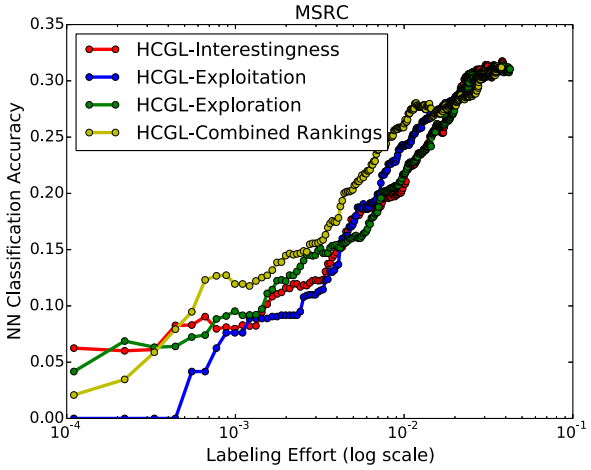
(a) Classification accuracy on 5-Scenes: uniform distribution of 5 scene classes



(b) Classification accuracy on Digits: uniform distribution of 10 digit classes



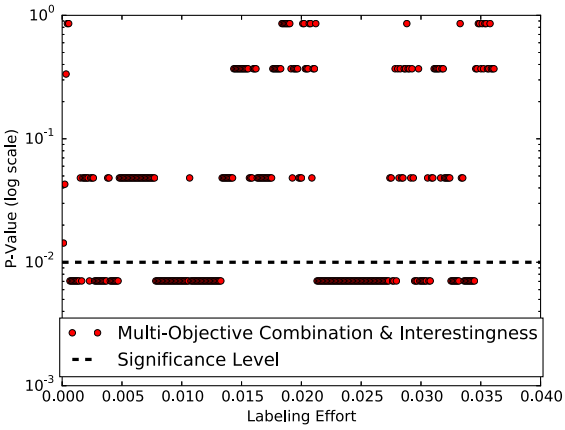
(c) Classification accuracy on 13-Scenes: uniform distribution of 13 scene classes



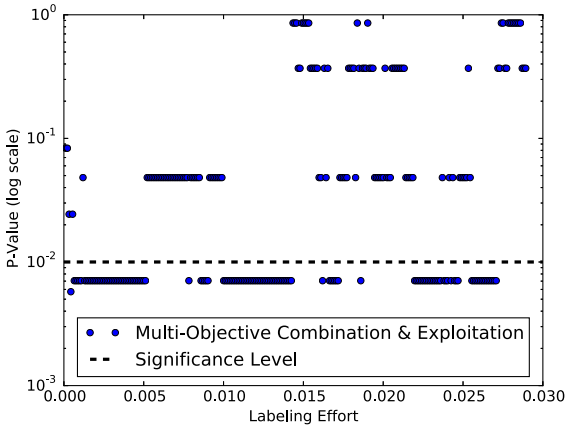
(d) Classification accuracy on MSRC: non uniform distribution of 16 object classes

FIGURE 6.4. Comparison of classification accuracy for the individual objective ordering criteria and the multi-objective combination of these criteria. Experiments are performed on three datasets with a uniform distribution of classes and one dataset with a non uniform distribution of classes.

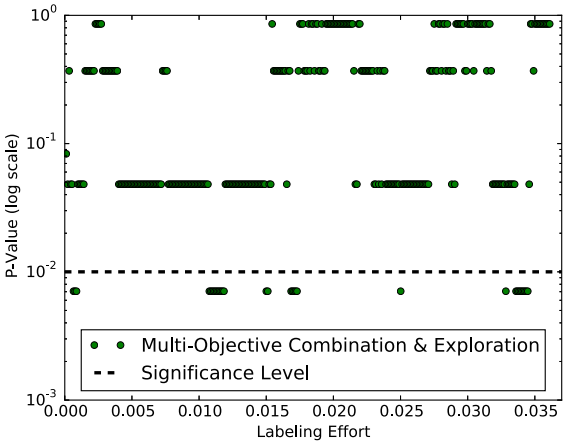
objective criteria used individually. For most of the analysis multi-objective combination ordering significantly outperforms both interestingness and exploitation ordering. There are fewer ranges of labeling effort where difference in classification is significant compared to the exploration ordering, but it is never outperformed significantly by the exploration objective ordering criterion.



(a) P-values when comparing multi-objective combination ordering with interestingness ordering.



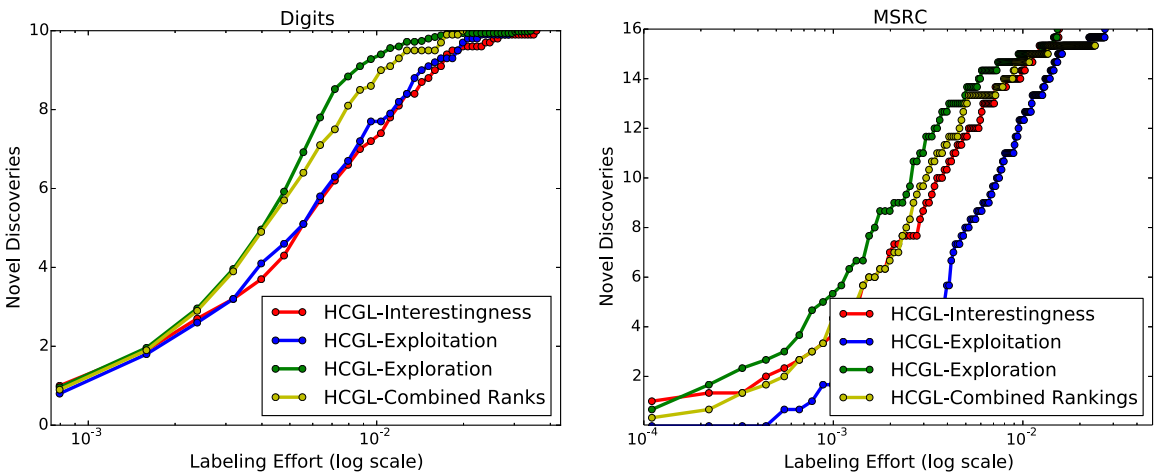
(b) P-values when comparing multi-objective combination ordering with exploitation ordering.



(c) P-values when comparing multi-objective combination ordering with exploration ordering.

FIGURE 6.5. P-values when comparing classification accuracy of multi-objective combination ordering with individual objective ordering criteria as a function of labeling effort. Results are for the MSRC dataset, which is the only dataset where multi-objective combination displays an improvement in classification accuracy over the individual criteria.

This change in trend from the MSRC dataset and the other three datasets in Figure 6.4 is best explained by one major difference. The datasets differ in their number of classes and degree of difficulty, but the MSRC dataset is particularly different because it is the only set of data that has a non uniform distribution of classes. This suggests that balancing the ordering criteria is particularly important when some classes appear less often than others.



(a) Discovery rate for the 10 class Digits dataset (b) Discovery rate for the 16 class MSRC dataset

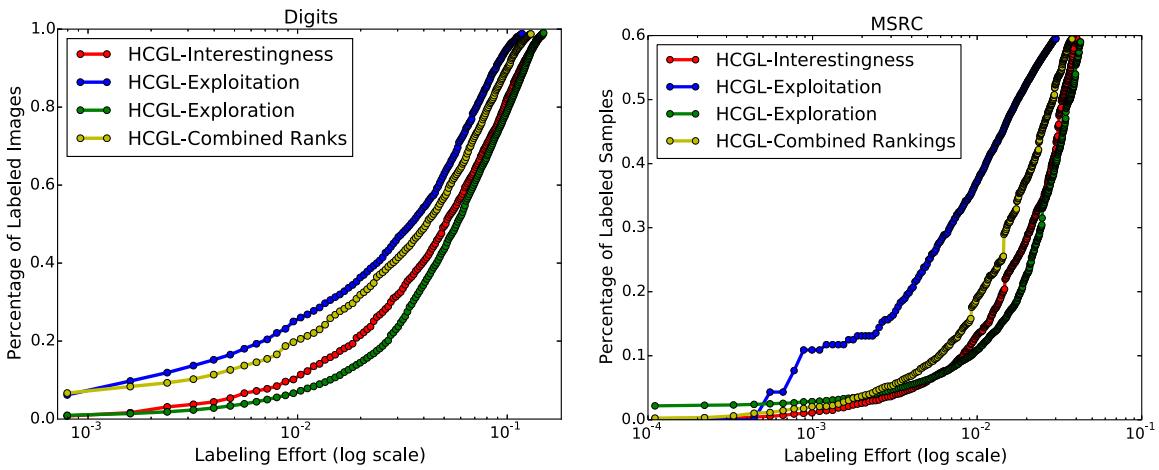
FIGURE 6.6. Comparison of the number of novel discoveries with respect to labeling effort for the individual objective ordering criteria and the multi-objective combination of these criteria.

This is an important result because the real-world application we address in Chapter 8 also exhibits a non uniform distribution of classes.

Even though multi-objective combination ordering did not outperform all individual criteria on the three datasets with evenly distributed classes, it did balance the ordering criteria well. This can be seen by comparing the labeling rate and discovery rate of all ordering techniques. Figures 6.6 and 6.7 show that multi-objective combination has the second fastest discovery and labeling rate. Of course it falls second to the individual ordering criteria designed to emphasize these objectives. We show this balance for a few example datasets, but the trend follows for all datasets seen in the classification results.

6.3. Discussion

We have yet to thoroughly investigate different β_i weights for the multi-objective combination technique. However, the comparisons in this chapter lead us to believe that dynamically adjusted weights throughout the labeling process may create a better combination of



(a) Labeling rate for the 10 class Digits dataset (b) Labeling rate for the 16 class MSRC dataset

FIGURE 6.7. Comparison of the labeling rate for the individual objective ordering criteria and the multi-objective combination of these criteria.

objectives. For example, the exploration objective may be weighted more heavily at the beginning of the labeling process to quickly discovery all classes, whereas exploitation may be weighted more heavily later in the process to collect more labeled instances of these classes.

The new outlier selection and ordering criteria introduce iterative updates to the HCGL framework during labeling. Although these updates require processing time, it is minimal compared to re-training classifiers or re-clustering data. Exploitation and exploration updates are only required for a subset of clusters in $\mathcal{H}$ that are in the neighborhood of the previously labeled cluster. $\mathcal{S}$ can be updated in $\mathcal{O}\log(n)$ time complexity by maintaining a sorted list of structural change values and searching the list using the outlier threshold.

Use of HCGL in real-world applications with a human annotator is discussed and evaluated in Chapter 8. This evaluation demonstrates the real-world feasibility of the system by presenting labeling interaction timings. The speed of updates when using outlier selection and multi-objective combination ordering was fast enough that there was no observable latency between labeling queries.

TABLE 6.1. P-values for ranges of labeling effort where significant classification performance differences are found on the 13-Scenes dataset after running the Wilcoxon signed-rank test. Tests are run for each combination pair of label orderings. The pair of orderings that display significantly different results are marked in bold.

Labeling Effort	Interestingness & Exploitation	Interestingness & Exploration	Exploitation & Exploration
0.000324	**0.002700**	0.338724	**0.004376**
0.000648	**0.004509**	0.514670	**0.005034**
0.000972	**0.005034**	0.959354	**0.005062**
0.001296	**0.005062**	0.798859	**0.005062**
0.001620	**0.007686**	0.721277	**0.005062**
0.001944	**0.006910**	0.386271	**0.005062**
0.002268	**0.006910**	0.507624	**0.005062**
0.002592	0.012515	0.114128	**0.005062**
0.002915	0.046853	0.021824	**0.006910**
0.003239	0.046853	0.028417	**0.009344**
...			
0.011662	0.074462	**0.006910**	0.575062
0.011986	0.074462	**0.009344**	0.575062
0.012310	0.092601	**0.006910**	0.575062
0.012634	0.059336	**0.006910**	0.959354
0.012958	0.092601	**0.005062**	0.959354
0.013282	0.046853	**0.005062**	0.959354
0.013929	0.028417	**0.005062**	0.959354
0.014253	0.036658	**0.006910**	0.721277
...			
0.028183	0.059336	0.798859	**0.006910**
0.028507	0.074462	0.878482	**0.006910**
0.028831	0.114128	0.646462	**0.009344**
0.029155	0.092601	0.798859	**0.005062**
0.029478	0.139414	0.959354	**0.005062**
0.030126	0.046853	0.798859	**0.009344**
0.030450	0.036658	0.721277	**0.009344**
0.030774	0.028417	0.575062	**0.009344**
...			
0.034661	0.139414	0.878482	**0.009344**
0.035309	0.139414	0.959354	**0.009344**
0.036929	0.114128	0.721277	**0.005062**
0.037577	0.139414	0.878482	**0.005062**
0.038873	0.092601	0.721277	**0.005062**
0.039845	0.168807	0.959354	**0.006910**
0.040492	0.139414	0.959354	**0.009344**
0.041464	0.114128	0.878482	**0.006910**
0.042760	0.202622	0.959354	**0.009344**

CHAPTER 7

Group Stability

Interestingness in the hierarchy was previously modeled using the primary direction of variance in the features of each group. Although good performance was seen during evaluation, this model does not generalize to all data representations. For example, kernels for unordered feature sets [60, 61] produce a similarity matrix, not a d-dimensional feature vector, by comparing sets of local features with different cardinalities.

To generalize to a larger set of data representations, we evaluate the use of stability measures to identify interesting splits in the hierarchy. We use stability to describe the consistency of data grouping under slight problem variations. The idea is that samples representing the same class will more consistently group together under problem variations producing different data partitions than those representing different classes. Different partitions can be achieved by grouping with random forests, data subsets or different feature representations. Like many cluster validation measures [62, 63, 64], stability has been used on flat data partitions to identify the "best" number of groups to represent the unlabeled samples [65]. We use stability in the hierarchical structure as a replacement for structural modeling in HCGL.

7.1. Proximity Forest

In this thesis, stability is derived from the Proximity Forest data structure [66]. This structure is a set of T randomized metric trees designed for fast Approximate Nearest Neighbor (ANN) search in general metric spaces. All that is required to build a Proximity Forest

is a distance function. This allows our approach to generalize to data represented by features in vector space and similarity matrices.

Each tree in the forest is constructed by incrementally adding training samples in random order. A splitting threshold size, τ, determines when nodes in the tree have reached their maximum size and should split into two children. To split, a random sample from a node is selected as the pivot element. Distances between the pivot and all other samples in the node are computed. The left child receives samples with a distance less than or equal to the median, and the right child receives those with distance greater than the median. Once the root splits, added samples traverse down to a leaf node using the splitting distance of internal nodes. Construction is complete when all samples have been added to a leaf node in the tree. Each tree in the forest represents a different randomized partitioning of the training data, and leaf nodes encode approximate nearest neighborhoods.

7.2. Measuring Stability

We present and discuss two variants of stability using the Proximity Forest. Just as with the original structural modeling, each $c_i \in \mathcal{H}$ receives a stability score. These scores are computed independent of $\mathcal{H}$ but are later compared within local neighborhoods of the hierarchy to represent the change in stability that occurs from each split.

7.2.1. Leaf Node Stability

The Proximity Forest was designed to do fast ANN search by retrieving samples from leaf nodes in $\mathcal{O}(\log n)$ time complexity. Using leaf node neighborhoods to evaluate pairwise stability is a natural extension, where the number of trees in which two samples share the same leaf node determines stability. Using leaf nodes, the stability between any two samples

x_i and x_j is

$$\text{pairwise-stability}(x_i, x_j) = \frac{1}{T}\sum_{t=1}^{T}\gamma_t(x_i, x_j), \tag{10}$$

where the function γ_t returns 1 if x_i and x_j exist in the same leaf node of tree t, and 0 otherwise. Pairwise stability is normalized by the number of trees in the forest to produce a stability score on the range of $[0.0, 1.0]$. Higher scores indicate greater stability.

We define the stability for a cluster in $\mathcal{H}$ as the average pairwise stability of its samples. Stability of a sample with respect to its cluster, $x_i \in c_i$ is

$$\text{sample-stability}(x_i, c_i) = \frac{1}{|c_i|}\sum_{\forall x_j \in c_i}\text{pairwise-stability}(x_i, x_j), \tag{11}$$

and the overall stability using leaf node neighborhoods in the forest is:

$$\text{leafnode-stability}(c_i) = \frac{1}{|c_i|}\sum_{\forall x_i \in c_i}\text{sample-stability}(x_i, c_i). \tag{12}$$

One disadvantage of this measure is that sample stability as defined in Equation 11 suffers from mismatched cardinalities between leaf nodes in the forest and $c_i \in \mathcal{H}$. Leaf nodes in the forest have a size on the range of $[\tau/2, \tau - 1]$. This means x_i can maximally be stable with $\tau - 1$ other data points. However, sample stability for x_i is averaged across all other $x_j \in c_i$. This creates a bias so clusters lower in the hierarchy (i.e., those more closely matching the cardinality of leaf nodes in the forest) appear to be more stable than groups near the top.

7.2.2. kNN Stability

To reconcile the cardinality mismatch from leaf node stability, we define a variant that retrieves the k approximate nearest neighbors from a tree in the forest, where $k = |c_i|$. When

$|c_i| > [\tau/2, \tau - 1]$, the internal ancestor nodes in a tree serve as larger approximate nearest neighborhoods from which stability is computed. The pairwise stability function for this variant is now defined as

$$\text{pairwise-stability}(x_i, x_j, k) = \frac{1}{T} \sum_{t=1}^{T} \zeta_t(x_i, x_j, k), \tag{13}$$

where ζ_t returns 1 if x_j is one of the k nearest neighbors of x_i found in tree t. Recall that since we are using the Proximity Forest, the set of k samples is actually the approximate nearest neighbors. This variant is also different because pairwise stability between x_i and x_j is no longer symmetric as was the case in pairwise stability computed from leaf nodes in the forest.

With this change, sample stability with respect to its cluster is

$$\text{sample-stability}(x_i, c_i) = \frac{1}{|c_i|} \sum_{\forall x_j \in c_i} \text{pairwise-stability}(x_i, x_j, |c_i|), \tag{14}$$

and every x_i is capable of achieving a maximum sample stability score of 1.0. Finally, the k-NN stability of c_i is also the average stability of its samples,

$$\text{knn-stability}(c_i) = \frac{1}{|c_i|} \sum_{\forall x_i \in c_i} \text{sample-stability}(x_i, c_i). \tag{15}$$

Unfortunately, this variant causes an inverse stability bias to that of leaf node stability. That is, k-NN stability measures tend to decrease down the tree. The pairwise redundancy

in $\mathcal{H}$ creates a ranking order bias. We use three example clusters from $\mathcal{H}$ to discuss this bias,

$$\begin{aligned} c_i &= \{x_i, x_j\} \\ c_j &= \{x_i, x_j, x_k, x_l\} \\ c_k &= \{x_i, x_j, x_k, x_l, x_m, x_n, x_o, x_p\}, \end{aligned}$$

where $c_i \subset c_j \subset c_k$. The pairwise stability between x_i and x_j for the three clusters is based on values of $k = 2, 4$ and 8 used in Equation 13. If we ignore that x_i will always be returned in its k-nearest neighbors, we see that x_j has the greatest likelihood of being stable with x_i when evaluating k-nn stability with respect to c_k. That is, x_j must be in a set of seven non-trivial ANNs with respect to c_k, whereas c_i and c_j only create sets of one and three non-trivial ANNs, respectively. The orderless sets used for k-nn stability causes larger clusters in the hierarchy to appear to be more stable than their descendants.

7.2.3. Change in Stability

To evaluate interestingness at all splits in the hierarchy, stability values (just like the structural representation) are compared for every cluster and its parent. Since both stability measures exhibit biases based on cluster size, we normalize the change in stability by the average stability score in the local neighborhood $\mathcal{N}$, where $\mathcal{N}$ is the set of all clusters in the hierarchy one path length away from either c_i or p. Formally, change in stability (either leaf node stability or kNN stability) is the normalized absolute difference between stability scores of c_i and p:

$$\Delta_S = \frac{|\text{stability}(c_i) - \text{stability}(p)|}{\frac{1}{|\mathcal{N}|}\sum_{\forall c_j \in \mathcal{N}} \text{stability}(c_j)} \tag{16}$$

Larger changes in stability indicate greater interestingness in the hierarchy.

7.3. Evaluation

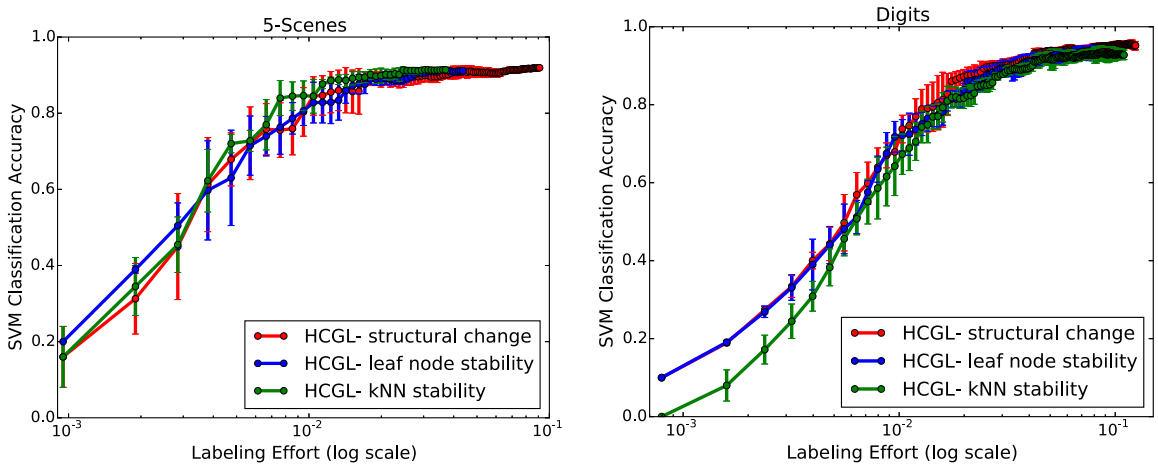
7.3.1. Experiment Setup

In these experiments, we hope to determine if change in stability provides a better model of interestingness than modeling changes in primary direction of variance. We use HCGL as defined in Algorithm 2 with the multi-objective linear combination of ranks from Equation 9. Both stability measures defined in this chapter and structural change from Equation 2 are used to model interestingness, and these three versions of HCGL are compared. Stability measures in these experiments are derived from a Proximity Forest constructed with 21 trees and $\tau = 25$. These parameter values were defined in previous work [67], and shown to work well for encoding approximate nearest neighborhoods for action recognition.

As in previous experiments in this thesis, evaluation are performed with the independent variable of labeling effort. We look at the effect of labeling effort on three dependent variables: classification accuracy, labeling rate and label accuracy.

7.3.2. Experiments

Figure 7.1 compares the classification performance achieved by the stability and structural change measurements on two datasets. Results on both datasets indicate no significant differences in classification performance after running ANOVA and evaluating at a significance level of $\alpha = 0.01$. However, the most distinguishing factor of the iterative classification results is that the stability measures complete the label collection task faster than the structural change measure. This can be seen by the shorter green and blue lines. So while



(a) 5-Scenes (b) Digits

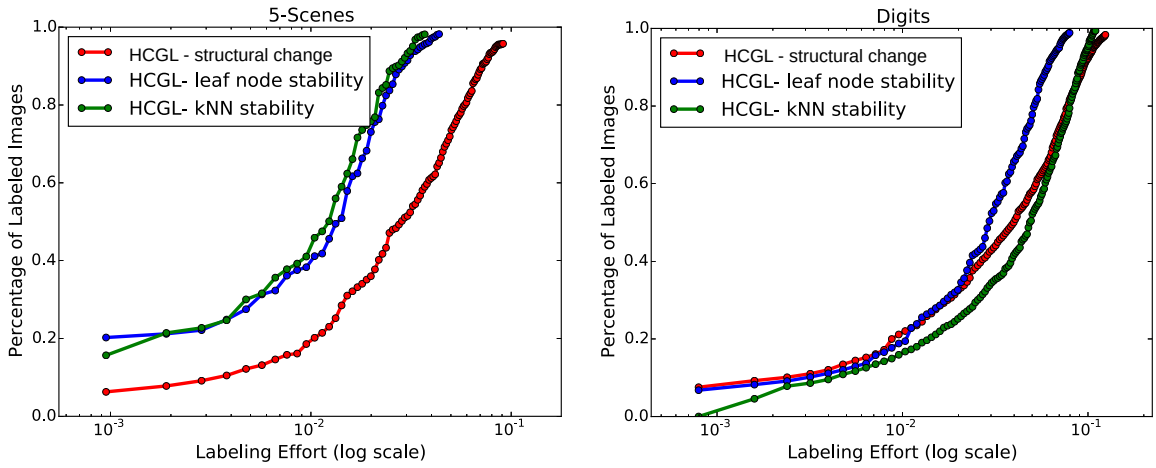
FIGURE 7.1. Comparison of classification accuracy between stability and structural change measures.

all techniques train similar performing classifiers, the stability measurements require less labeling effort.

This difference can be seen more directly by evaluating labeling rate as a function of labeling effort, seen in Figure 7.2. It is easy to see that each interestingness measure produces a unique construction of $\mathcal{S}$ based on the differing labeling rates and label accuracies (seen in Figure 7.3). The stability measures tend to incorporate groups from $\mathcal{H}$ with slightly more label noise which often maps to larger groups being labeled at any given iteration.

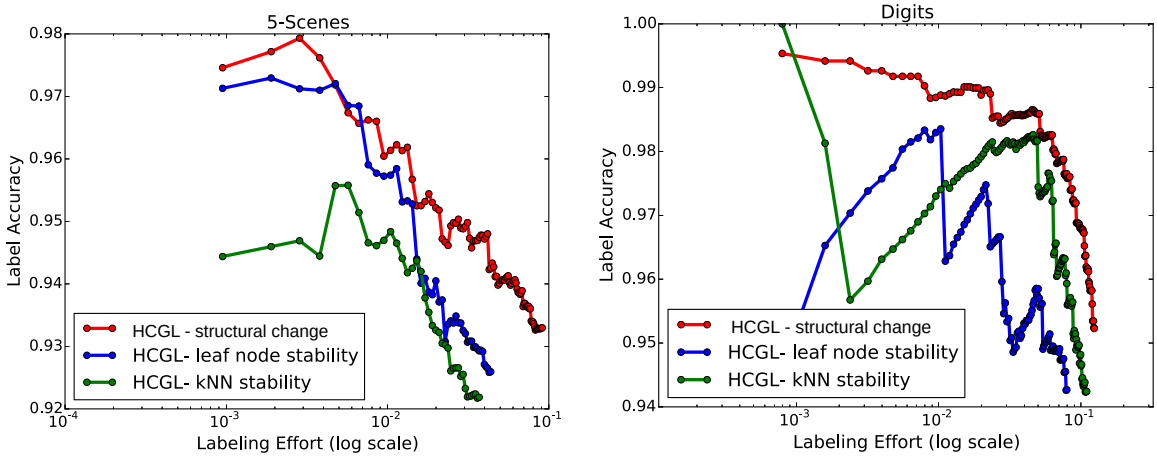
7.4. Discussion

One major disadvantage of the stability measure is the pairwise computational cost, resulting in k^2 comparisons to measure the stability of c_i. This pre-processing computation does not introduce latency during labeling, but does require more initial overhead startup compared to extracting the primary direction of variance for c_i. Randomly sampling elements from c_i can reduce the number of pairwise comparisons, but if the iterative classification



(a) 5-Scenes (b) Digits

FIGURE 7.2. Comparison of labeling rate between stability and structural change measures.



(a) 5-Scenes (b) Digits

FIGURE 7.3. Comparison of label accuracy between stability and structural change measures.

trend carries over to other vector datasets it is much faster overall to use structural change measurements in HCGL.

In this thesis we emphasize the evaluation of classification accuracy in the early stages of the labeling process because it is not always feasible to label the entire training set. However, if the end goal was to label the entirety of $\mathcal{T}$ the fastest with minimal label noise collection,

the leaf node stability measure showed the greatest success and consistency at this task. This measure was able to reduce labeling effort by over a factor of ten for both datasets.

CHAPTER 8

Applications in Mobile Robotics

We have demonstrated the efficiency of HCGL using single concept image benchmark datasets. In this chapter, we integrate HCGL into mobile robot applications to show the real-world feasibility of our labeling framework. We discuss the relevance of HCGL for perception systems in dynamic environments, and in collaboration with U.S. Army Research Laboratory, we evaluate trained visual classifiers used for environment perception and autonomous navigation tasks.

Accurate environment perception is critical for autonomous robots to plan paths on traversable terrain and avoid object collision during navigation. While many sensors have been used for environment perception (e.g., laser range finders [68], radar [69], and contact sensors [70, 71]), the speedup seen in image processing has allowed vision-based perception to emerge in mobile robots [72, 73, 74]. This is beneficial for path planning because visual data allows robots to perceive a large area of the environment at once.

To ensure the highest quality visual perception, training data is collected from the environment where the robot will be performing its navigation task. Thus, each domain change requires new collection and labeling of training data. We define the time required for the image annotation process of a new set of training data as ***adaptation latency***. This represents the time a robot is unable to perform navigation tasks autonomously in a new domain.

Adaptation latency has yet to be discussed in existing multi-concept visual perception systems used in robotics applications [72, 73, 74, 75]. Hand annotation of region boundaries followed by label assignment is simply performed as a necessary, but time consuming step to train supervised classifiers. However, beyond being an undesirable task, it may be infeasible

in some scenarios that have limited labeling time and resources. This evaluation is motivated by these scenarios that seek minimal adaptation latency without significantly compromising the ability to visually perceive the environment.

8.1. Related Work in Robotics

Vision is an emerging technique for environment perception [76] with applications in mobile robots such as gait and speed adjustments and path planning. Terrain classification in particular has received serious attention because of its importance in robot traversability. Offline pixel-wise evaluation of supervised terrain classifiers has been performed for images captured by aerial robots [72], and techniques that combine vision with other sensor modalities [69, 75]. Although ground truth datasets allow for quantitative evaluations, they provide less insight about visual perception in dynamic real-world environments than online task-based evaluation.

Online evaluation of visual terrain classification is commonly performed for gait change tasks [73, 74]. For this task, visual perception is often scoped down to a small local area in front of the robot (not the entire environment), and classification is performed on homogenous terrain regions. In path planning tasks, visual perception of environments requires both localization and classification of objects and terrains. Most real-time path planning experiments have been performed using non-visual sensors [68] or perform visual labeling and planning at a binary level, e.g., *path* and *non-path* [77]. Self-supervised techniques [69, 78, 79, 80] also have been used to learn binary concepts like traversability. These frameworks adapt online through multiple, complimentary sensor modalities without the time consuming labeling process. However, these do not extend to providing training data for more complex multi-class tasks such as verbal navigation commands between a human and robot [81, 82].

A major hardship of running multi-concept supervised visual perception for path planning tasks is the time commitment required to collect labeled training data to adapt to new environments. Haselich et al. [75] speak directly to this issue when discussing their laser-vision fusion approach, and its limited ability to adapt quickly because it requires the re-annotation of imagery for new environments.

8.2. Experiment Setup

In this chapter we compare HCGL to the supervised labeling baseline, LabelMe, where training images are labeled in random order. We do not make any direct comparisons to other existing efficient labeling frameworks because the previous chapter already demonstrated the superior performance of HCGL with respect to labeling effort. We use the version of HCGL defined in Algorithm 2 with rank-based combination scores (from Equation 9) to sort $\mathcal{S}$ as defined on line 6.

All evaluation metrics are presented as a function of labeling interaction time (adaptation latency) to show the speed at which techniques can train visual classifiers for environment perception[6]. We begin by comparing labeling speed and pixel-wise classification accuracy. However, our main focus is on task-based evaluation. As shown earlier, HCGL achieves greater efficiency in exchange for a slight degradation in label accuracy. This trade-off is beneficial for navigation tasks because although label noise may impact classifier learning, perfect pixel-wise classification is often not necessary to plan paths for successful navigation.

[6]Material submitted by author to ICRA 2016 [83].

TABLE 8.1. Details of environment datasets.

Environment	# Training Images	Label Set ($\mathcal{Y}$)
A	274	*asphalt,building,concrete-floor grass,gravel,object,sky,tree*
B	1,982	*building,car,grass,object road,sidewalk,sky,tree*
C	268	*building,car,curb,grass,object road,sidewalk,sky,tree*

8.2.1. Environment Datasets

We use three real-world environments to demonstrate the speed and performance of HCGL when collecting labels for multi-concept imagery for visual perception. The environments are outdoor urban training facilities that include multiple types of terrain, buildings, cars and other objects. Training data for environment A was collected with a high dynamic range camera at a previous experiment performed in October 2012. Images were taken at 5 different time blocks over two days from 53 locations in the environment [82]. Training data for environment B and C is captured via teleoperation using the robot described in Section 8.2.4. This data was collected several days before live navigation experiments were performed in environment B and several weeks before experiments in environment C. The training set from Environment B is much larger than the other environments because it is the combination of data collected on three consecutive days. Performance on this dataset shows how HCGL is able to scale with increasing training set sizes. An overview of the datasets is provided in Table 8.1, and example images are seen in Figure 8.1.

Since HCGL assumes that each training sample represents a single concept, environment data must be segmented into multiple regions. We over-segment images into approximately 150 regions using SLIC [21]. Over-segmentation is performed to better ensure that true region boundaries are observed in the training samples. Each segment represents an individual

FIGURE 8.1. Example images from the three environment training sets.

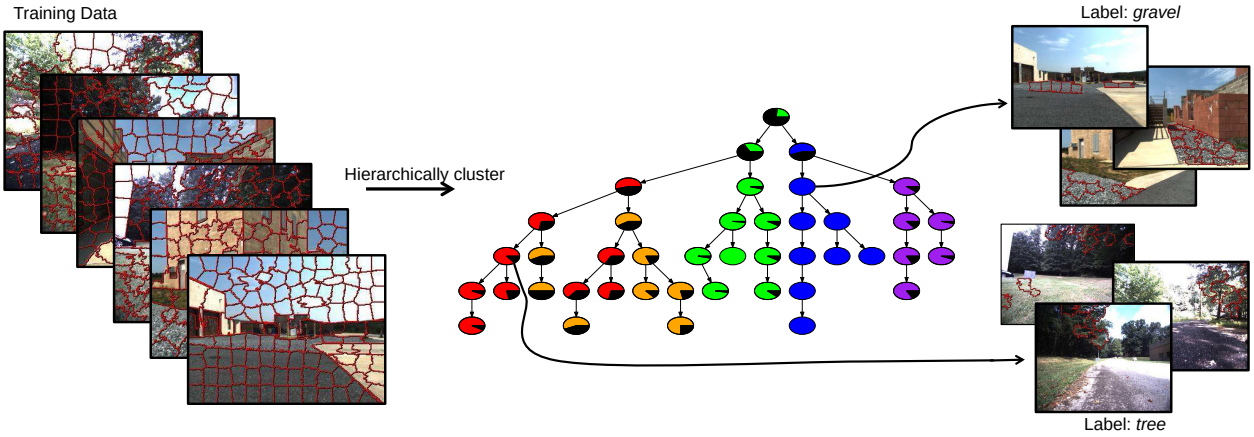


FIGURE 8.2. Visualization of HCGL process on multi-concept environment data. Training data is over-segmented, segments are hierarchically clustered, and clusters from the hierarchy are selected, displayed to the user and assigned the majority concept label.

training sample that is clustered and represented using LAB color histograms, local binary patterns [84], a 200-dimensional SIFT [37] codebook and normalized region coordinates. The HCGL framework with over-segmentation of environment data is depicted in Figure 8.2. Node colors map to a class in the label set, $\mathcal{Y}$, and black wedges represent the percentage of noise in each cluster (images not representing the dominating class).

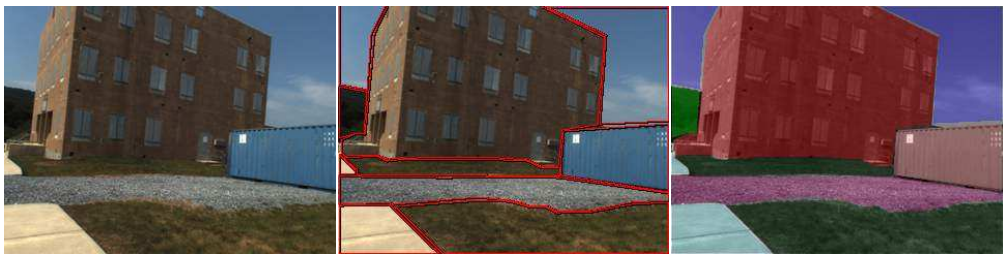


FIGURE 8.3. Supervised labeling input, required outlining and label output.

8.2.2. Supervised Labeling

Labeling single-concept images in a supervised manner can be time consuming when training sets are large. Multi-concept images are even more demanding for human annotators because regions of interest must be hand outlined before being assigned a label. Publicly available multi-concept image annotation tools such as LabelMe [33] have emerged in the research community to facilitate supervised labeling. LabelMe allows annotators to precisely outline, via mouse clicks, and assign a label to each distinct region in an image. Figure 8.3 shows the supervised labeling input (left), the required outlining (middle) and labeled output (right - see class/color legend in Figure 8.13) for a single image.

Although this technique produces high quality labeled data, supervised annotation requires significant human effort. A fully annotated training set of 250 images requires well over 20 hours of labeling effort (discussed further in Section 8.3). This labeling latency is introduced during domain changes and inhibits fast adaptation. We use LabelMe as a baseline labeling technique for experiments in this chapter, and precisely record interaction time when using the framework for environments B and C. Interaction time for environment A is empirically estimated because the dataset comes fully annotated without any timing information available. This was done by having two human annotators label a set of 34

images independently. Timing results suggest that on average a single image is labeled in about six minutes.

8.2.3. Visual Classifier

To test environment perception using labels from HCGL and LabelMe, we train a Hierarchical Inference Machine (HIM) [85], an approach for scene parsing and region classification. HIMs incorporate both feature descriptors and contextual cues computed at multiple scales within the scene. Images are decomposed into a hierarchy of nested superpixel regions [20], where regions at the bottom provide localized discriminative information and those at the top provide global context. The predictor is a decision forest regressor with 10 trees, where the label distribution of superpixel regions at a coarse level of segmentation are used to refine predictions at a finer level of segmentation with greater spatial locality. Features extracted from superpixels include SIFT [37], LAB colorspace statistics, texture information and statistics on the size and shape of superpixel regions.

The HIM processes a 640×384 image in approximately 2 seconds on a dedicated quad-core i7-3615QM at 2.3 GHz, with feature extraction being the dominant cost, which allows us to run navigation experiments in real-time. This algorithm has been rigorously tested during other field experiments; for a detailed description please refer to [82].

8.2.4. Hardware

The trained HIM is run on a Clearpath Husky seen in Figure 8.4. This robot is a wheeled platform that is limited to a maximum velocity of 1 m/s. The Husky collects images using a Prosilica GT2750C, a 6 megapixel CCD color camera. Further information about the Husky used in these experiment can be found in [83].

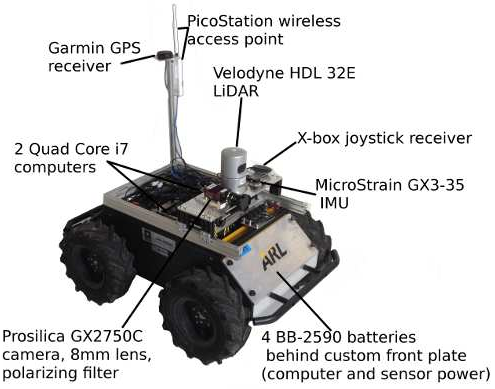


FIGURE 8.4. The hardware configuration of the Clearpath Husky robot.

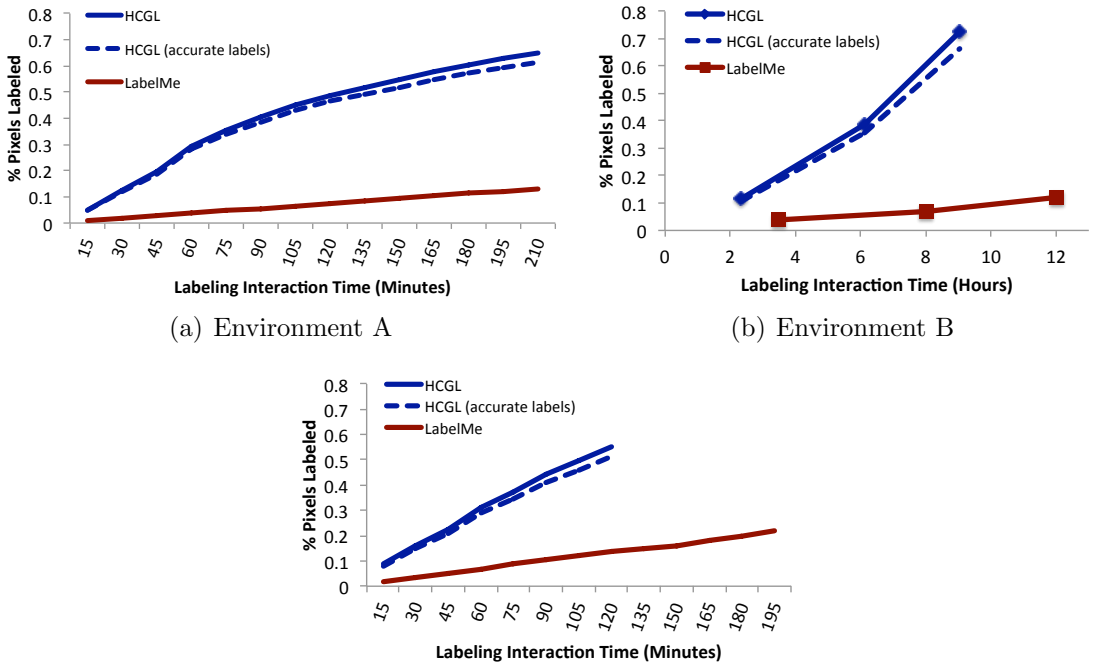
8.3. Labeling Speed Evaluation

Our first evaluation looks at how fast HCGL and LabelMe assign labels to pixels of the training images. Figure 8.5 shows the percentage of labeled pixels as labeling interaction time increases. There is a large performance gap seen across all datasets. Given limited labeling time, HCGL is able to collect around six to seven times the amount of label information as LabelMe. Labeling interaction time for environment B is on the order of hours because each of the three days of training data were labeled separately and then combined.

Assigning labels quickly is important, but recall that to achieve this speed, HCGL trades some label accuracy via majority labeling. The dashed blue lines in the plots show the percentage of pixels that received accurate labels from HCGL (determined using the labels collected by LabelMe). This line represents $\sim 5-10\%$ pixel label noise: a small fraction for a large gain in performance.

8.4. Pixel-Wise Classification Evaluation

Just as in previous chapters, labels collected from HCGL and LabelMe are compared by training visual classifiers and evaluating pixel-wise classification accuracy of the classifiers on a disjoint testing set. Pixel-wise classification accuracy is compared on a testing set from



(a) Environment A

(b) Environment B

(c) Environment C

FIGURE 8.5. Rate of label assignment to pixels in the training sets, and accurate label assignment of HCGL (dashed lines).

Environment B which consists of 265 images. This is the only environment dataset with a large, labeled testing set [82]. Classification evaluation is performed incrementally after every 15 minutes in which a user assigns labels to the training set. Figure 8.6 shows the overall pixel accuracy for environment A. Even though HCGL introduces small amounts of label noise, the larger volume of labeled training data allows HCGL to train higher performing classifiers than LabelMe up to 210 minutes of labeling interaction. HCGL labeling is terminated at this point to depict scenarios that have limited time to devote to label collection. So although LabelMe eventually reaches and surpasses the classification performance of HCGL, it may require time that is not feasible in all scenarios.

Overall pixel classification accuracy can be skewed by classes that have a higher distribution of pixels in the images. Thus, we run per-class classification accuracy and find that HCGL performs similarly or better than LabelMe for all classes but one, seen in Figure 8.7.

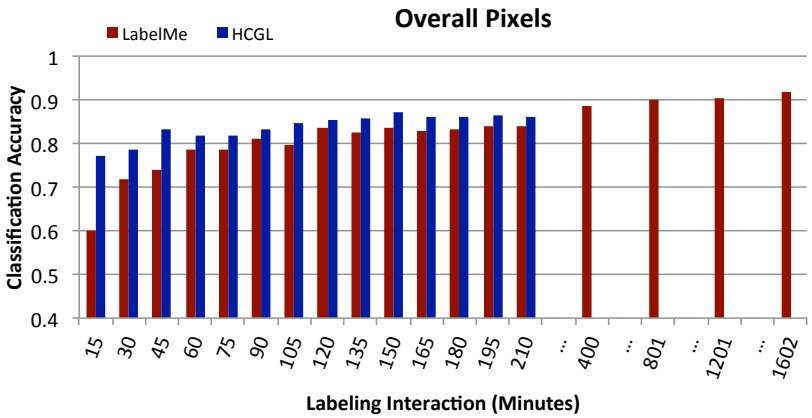


FIGURE 8.6. Overall pixel classification accuracy for environment A.

The *object* class is the least represented in the data and composed of many diverse things, e.g., light poles, traffic cones and cargo boxes. The low intra-class similarity made it difficult for all object samples to group together in HCGL. These factors caused few labels to be collected and achieved lower accuracy than LabelMe. However, this was a difficult class for LabelMe as well. With a fully labeled training set (1,602 minutes), LabelMe achieves only $\sim 18\%$ classification accuracy for the *object* class.

8.4.1. Exploitation Versus Rank-Based Ordering

In applications with limited time for label collection, it can be tempting to run HCGL with exploitation ordering to collect as many labels as possible in the allotted time. However, the environment data used in this chapter displays a large skew in classes distribution. Figure 8.8 shows the class distribution breakdown across all pixels in the training set for Environment A data. We compare HCGL with exploitation ordering and rank-based ordering to show the importance of balancing all ordering criteria during the labeling process.

As designed, HCGL with exploitation based ordering focuses on labeling large groups of data quickly. This results in a small number of classes actually receiving labels early in the labeling process because most of the training samples represent either *sky*, *grass* or *building*.

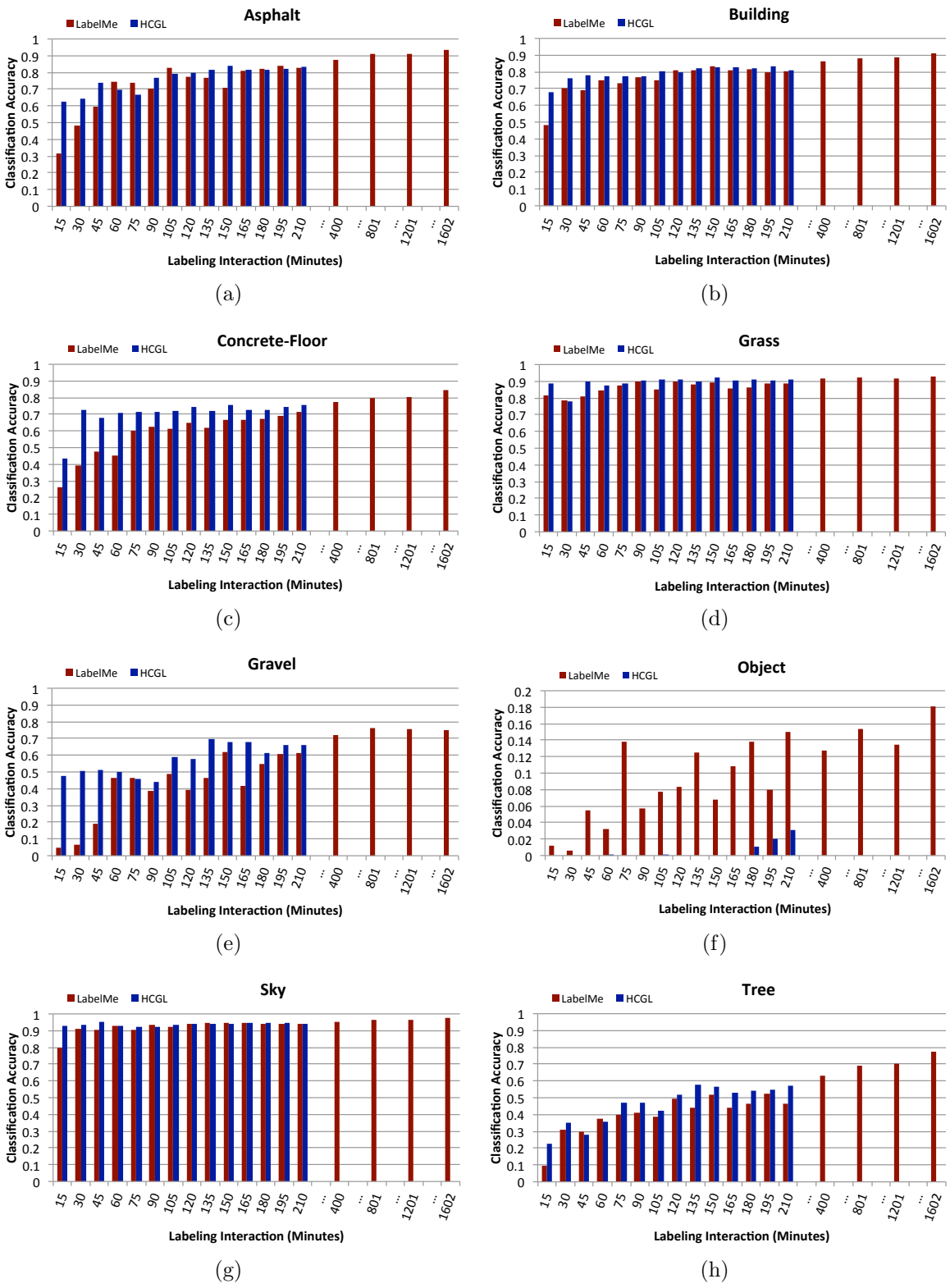


FIGURE 8.7. Class specific classification accuracy for environment A.

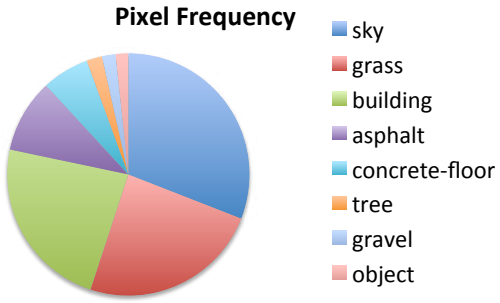


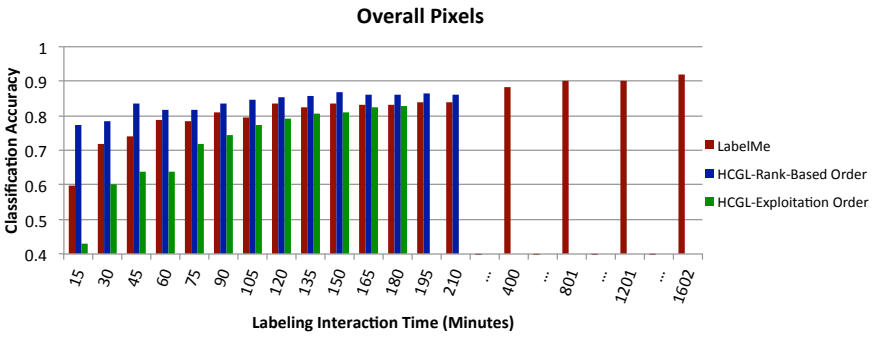
FIGURE 8.8. Breakdown of class distribution across all pixels in the training set of Environment A.

This ultimately produces far worse classifiers through the first 90 minutes of labeling than rank-based ordering and LabelMe. These results are shown in Figure 8.9. We only compare rank-based ordering to exploitation ordering on the real-world data to show the significance of class distributions on label collection.

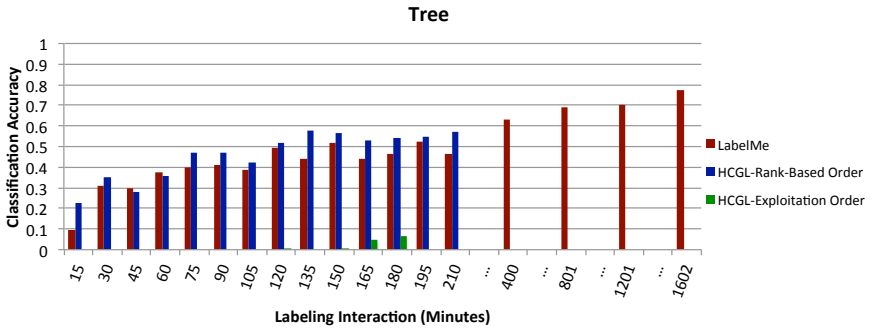
8.4.2. Qualitative Evaluation

A large labeled test set from environment B is not available, so instead we provide a qualitative comparison of HCGL and LabelMe. Figure 8.10 shows classified images from the two classifiers. There are two types of terrain in this environment, *road* and *grass*. Even though the LabelMe classifier is trained with more human interaction time, HCGL collects significantly more label information and trains higher performing classifiers.

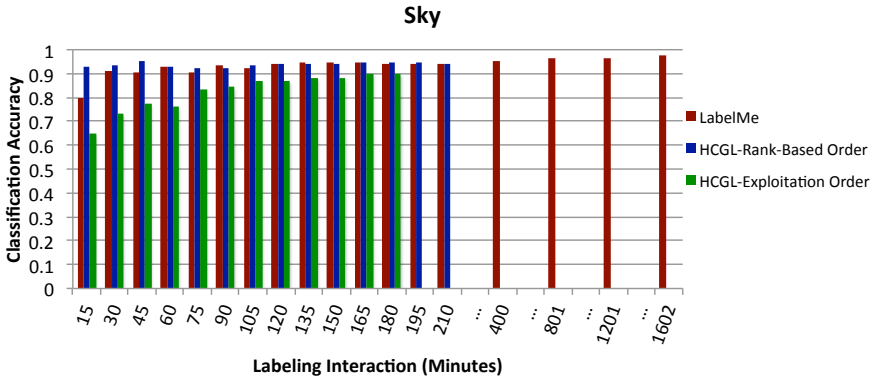
The testing examples show instances where both classifiers perform similarly, an example where HCGL performs slightly worse than LabelMe (column three), and the last three columns are examples of common mistakes made by the classifier trained using the LabelMe data. In particular, the LabelMe classifier has a hard time correctly identifying terrain that is not directly in front of the robot. While this may allow a robot to make immediate decisions it would negatively impact long term path planning tasks that we will discuss next.



(a) Overall pixels



(b) Class with small distribution of pixels.



(c) Class with large distribution of pixels.

FIGURE 8.9. Classification accuracy for environment A using different HCGL selection ordering.

Qualitatively it can also be seen that HCGL commonly misclassifies trees and certain objects as sky.

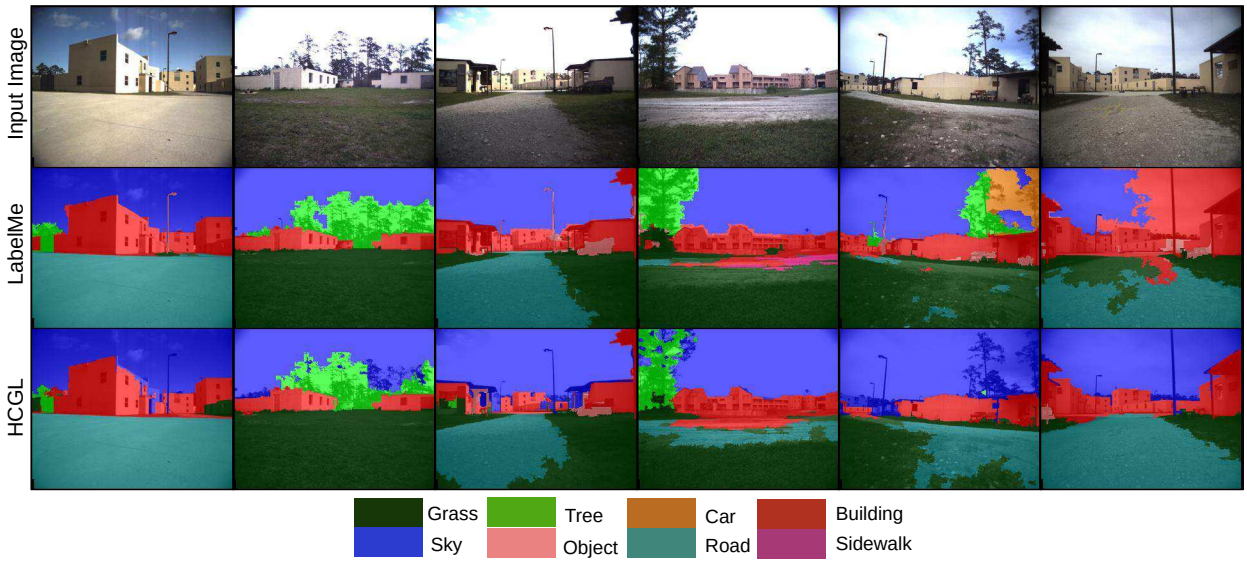


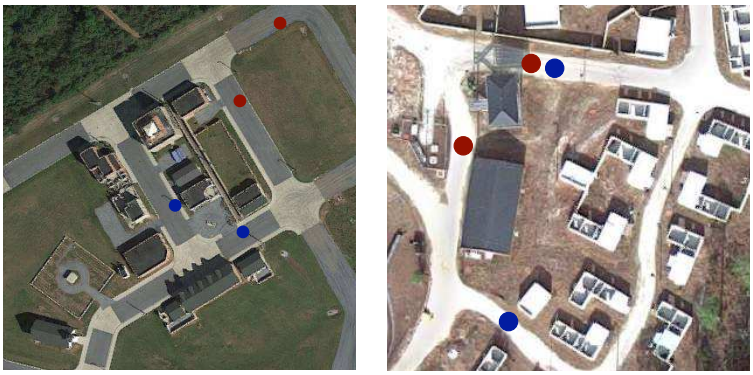
FIGURE 8.10. Classified test images from environment B.

8.5. Real-Time Autonomous Navigation Experiments

While standard evaluation protocols provide an easy way to compare performance on static data, task-based evaluation judges the quality of perception relative to our end goal of successful navigation in dynamic environments. In many cases, perfect pixel-wise accuracy is not required to accomplish navigation tasks. This section compares several visual classifiers, trained using labels collected from HCGL and LabelMe, and their ability to provide perception information to the mapping and navigation framework for real-time autonomous navigation tasks.

8.5.1. Task Description

Our live navigation task requires the robot to use visual perception to plan paths and traverse between waypoints using road terrain only. We restrict the robot to road traversal because roads are designed to provide navigation guidance to vehicles. For example, roads direct vehicles around difficult terrains like sand, hazards like bodies of water and non-public areas like front yards. Our experiments emulate these scenarios by defining waypoints as



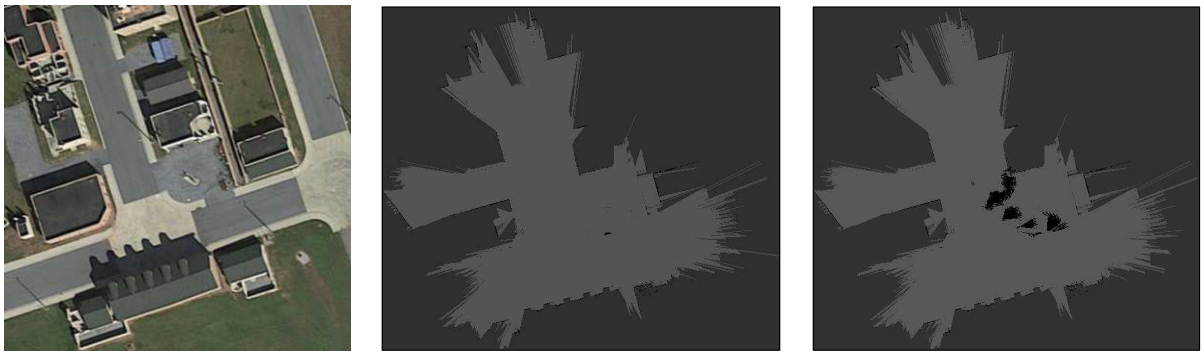
(a) Environment A (b) Environment B

FIGURE 8.11. Navigation waypoint maps for environments.

GPS coordinates (seen in Figure 8.11) such that the most direct path to the goal is not along a road.

Labeling models and the associated trained classifiers are compared using the successes and failures seen throughout multiple trials of a waypoint navigation task, where outcomes are defined as follows:

- **Success** - the robot autonomously traverses between waypoints using only road terrain without hitting objects
- **Success with Minor Errors** - the robot traverses between waypoints but either, 1) traverses on non-road terrain for a short duration before re-planning a road terrain route or 2) requires operator intervention at least once but no more than twice for small adjustments in location or direction due to potential object collision or planner failure
- **Failure** - the robot cannot plan and execute a road traversal even with minimal operator intervention; visual perception has major false-positive errors indicating no path of road terrain or planner constantly updates without making progress towards the goal



(a) Environment (b) LiDAR map (c) Vision map

FIGURE 8.12. Example obstacle maps for location two in environment A. Darker regions indicate obstacles and non-traversable terrain.

Visual perception via HIM is run online during the navigation tasks. To plan paths the robot maintains an obstacle map that depicts objects and terrains that should be avoided during path planning. This is updated after every image processed by HIM. An example obstacle map without visual perception of traversable terrain is seen in Figure 8.12(b), whereas Figure 8.12(c) shows the obstacle map using visual perception. Details pertaining to the mapping and navigation software on the robot are beyond the scope of this thesis, but can be found in the collaborative research with U.S. Army Research Laboratory [83].

8.5.2. Navigation Results - Environment A

Environment A is the primary location used for comparative evaluation since LabelMe was used to label the entire training set [82]. Four classifiers are trained and compared. We compare the labeling techniques given the same amount of labeling interaction time. HCGL-150 and LabelMe-150 represent classifiers trained after 150 minutes of labeling, which reflects scenarios where limited time is available for label collection. This is just under one-tenth of the estimated total time, 1,602 minutes, required to label the entire training set with

TABLE 8.2. Summary of navigation results for location one (red waypoints) in environment A.

	# Successes		
Label Model	No Errors	Minor Errors	# Failures
HCGL-150	3	0	3
LabelMe-150	2	1	3
LabelMe-1602	1	1	2
HCGL-150+30	**7**	**1**	0

LabelMe. To demonstrate results given no time restrictions, we also train a classifier using the entire training set, denoted as LabelMe-1602.

The final classifier is meant to show that adaptation is not only relevant to domain changes, but also locally within an environment as it undergoes changes over time. We supplement the existing training data (collected several years ago), with an additional 231 images collected from environment A during these experiments (but not at the specific testing locations). We spend 30 minutes using HCGL on this new set of images and assign labels to $\sim 27\%$ of the pixels. This set of labeled training data is combined with the labeled training data from HCGL-150 to train the final classifier, denoted as HCGL-150+30.

The navigation task is performed at two locations in the environment. The first location is illustrated by the red waypoints in Figure 8.11(a). Path planning must avoid grass terrain (the shortest path between waypoints) and several objects near where the grass and road meet. Road terrain is defined as *gravel*, *concrete* and *asphalt*. Each trial represents a traversal from one waypoint to the other, and are performed in both directions. Trials were run across multiple days, at different times of the day to capture performance under varying environment conditions. Table 8.2 compares the performance of each classifier at this location.

HCGL-150 and LabelMe-150 perform similarly with a lot of inconsistency. LabelMe-1602 has a similar failure rate as the 150 minute labeling models, which suggests that the

environment has changed since the collection of the training data. For example, we notice that the length of the grass is significantly longer than it was in the training data. HCGL-150+30 on the other hand, performs the navigation task very reliably across both days of testing. The single trial marked as success with minor errors involved the robot trying to plan a shortest path through the grass, entering the grass for a brief moment before backing out and successfully planning a road traversal route. We perform more trials for this method to show its consistency. These results demonstrate the impact of rapid label collection when new training data needs to be collected to adapt and improve visual perception.

Qualitative evaluation of visual perception (examples shown in Figure 8.13) shows that the labeling models produce classifiers that make different mistakes. HCGL-150 tended to confuse road terrains, whereas LabelMe-150 often misclassified road terrain as *building* and *object* (seen in columns three and five). LabelMe-1602 has cleaner results than the 150 minutes models, but often still misclassified *gravel* as *object*, seen in column three. Although still not perfect pixel-wise classification, HCGL-150+30 displays the most accurate results compared to the ground truth, which is expected given its superior navigation success.

The second location is shown in Figure 8.11(b) as the blue waypoints. At this location, road terrain is defined only as *concrete* and *asphalt*, whereas *gravel* terrain (shortest path between waypoints) is defined as non-road. Along the shortest road path are two objects (traffic cones) that the robot must also avoid. This location was chosen to test terrain classification for classes with much higher inter-class similarity.

Experimental trials are limited for this location due to time constraints and were performed in a single afternoon. Further, more trials are performed using HCGL during this limited testing time to better evaluate HCGL in real-world applications. Comparisons are

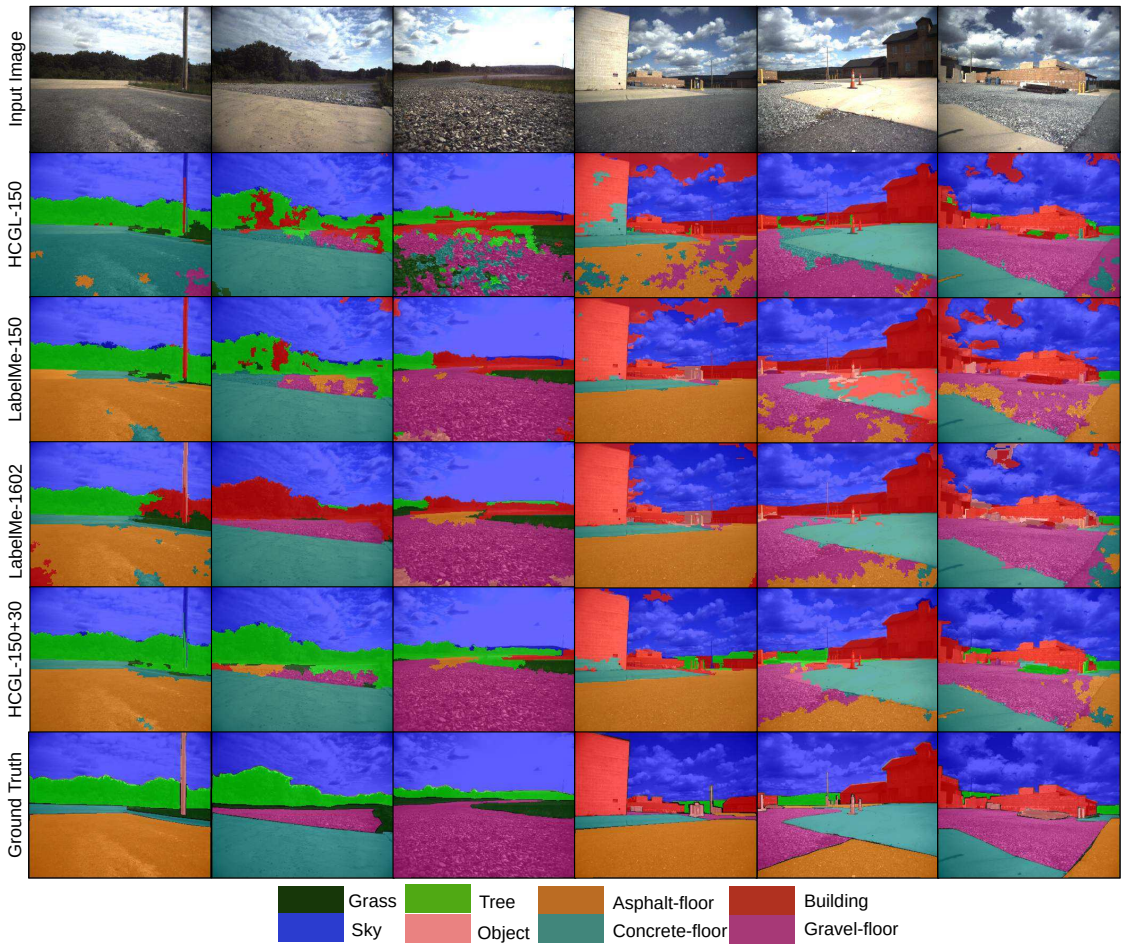


FIGURE 8.13. Visual perception examples for each of the labeling models. The first three columns are images taken are from the first location (red waypoints), and the last three columns are images taken from the second location (blue waypoints).

provided for LabelMe-1602 and HCGL-150+30: the most successful models at the first location in terms of successes and qualitative evaluation. Results are summarized in Table 8.3, and indicate that this navigation task is much more challenging. However, HCGL-150+30 is still able to successfully navigate the majority of the time with minimal errors. Most failures and errors at this location were caused by misclassifying *asphalt* as *gravel*. This can be seen in the last three columns of Figure 8.13.

TABLE 8.3. Summary of navigation results for location two (blue waypoints) in environment A.

	# Successes		
Label Model	No Errors	Minor Errors	# Failures
LabelMe-1602	0	0	3
HCGL-150+30	**3**	**2**	3

8.5.3. Navigation Results - Environment B

We use environment B to further test the label collection of HCGL in new domains. Since terrain classes in this environment are limited to *road* and *grass* we define *grass* as non-traversable. We chose to focus our navigation trial experiments on labels collected using HCGL to show the consistency of the system across multiple environments.

We ran over 15 navigation trials between both sets of waypoints without any failure cases. Only minor path planning errors in a few trials caused the robot to traverse on the edge of the grass where it meets the road. These successes are used to confirm that the small amount of label noise collected by HCGL, in exchange for fast label collection, does not negatively impact road navigation and path planning. Figure 8.14 shows visual perception from three different trials (one per row) during 20, 40, 60 and 80 percent through the trial traversal (columns). The class-color key is the same as that seen in Figure 8.10.

Overall perception is very strong, hence the successful navigation. Live navigation experiments also show that trees are commonly misclassified as sky, but have little impact on the navigation task. It can also be seen that some buildings further in the distance are misclassified as sky, e.g., column one of trial one. While these errors are likely due to label noise introduced by HCGL we see that the robot is still able to accomplish the defined task.

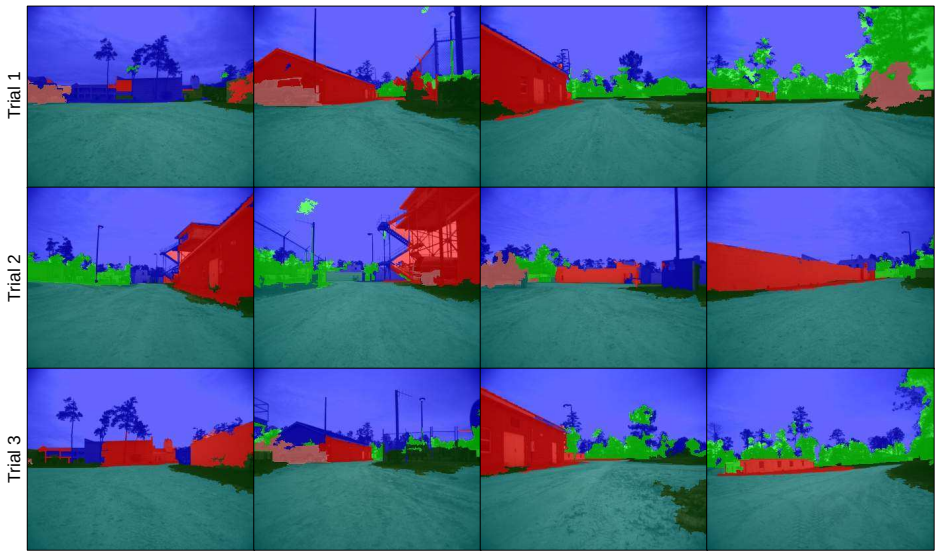


FIGURE 8.14. Visual perception examples from live navigation trials in environment B.

8.6. Summary

Real-time visual perception for mobile robots is only as useful as its ability to quickly adapt to changing environments. These experiments demonstrated the speed of label collection that HCGL can bring to real-world applications and multi-concept environment data. New training data can be collected, labeled and used to train a classifier in a single day. HCGL is a good candidate for task-based visual classification because while HCGL trades some label accuracy for reduced adaptation latency, perfect pixel-wise classification is not necessary to achieve quality visual perception for navigation.

CHAPTER 9

Exploiting Hierarchical Label Sets

When labeling with HCGL the user is permitted and encouraged to assign labels that best match the concept discovered by the clustering algorithm, even if these labels are more general than the final classifier label set. In Chapter 2, this was defined as the label set $\hat{\mathcal{Y}}$, a superset of labels from those in $\mathcal{Y}$. The hierarchical label set was primarily motivated by coarse-grained concepts found in the 13-Scenes dataset, and was designed to help reduce the amount of label noise introduced by majority group labeling. In this chapter we discuss $\hat{\mathcal{Y}}$ with respect to the real-world environment datasets used in the previous chapter. We discuss how much additional information is available about an environment when also considering coarse label assignment from the hierarchical label set, and how this information can be exploited to provide additional labeled training data to visual classifiers.

9.1. Coarse-Grained Label Assignment

Since all three environments were used for the same navigation task, the classifier label set, $\mathcal{Y}$, is similar for all environments with the exception that environment A has more specific terrain types. Four coarse-grained labels were commonly given by the user when labeling the environment datasets: *ground*, *vegetation*, *concrete* and *hard ground*. Figure 9.1 depicts the hierarchical relationships between these coarse-grained concepts and labels in $\mathcal{Y}$.

During the labeling process coarse-grained labels are collected and maintained but not used to train classifiers. Thus, the information available about the environment is actually denser than the labeled training set the classifier learns from. Figure 9.2 shows the additional label information collected by HCGL on the three environment datasets when crediting all

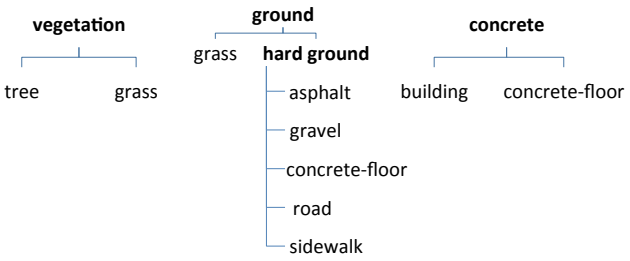
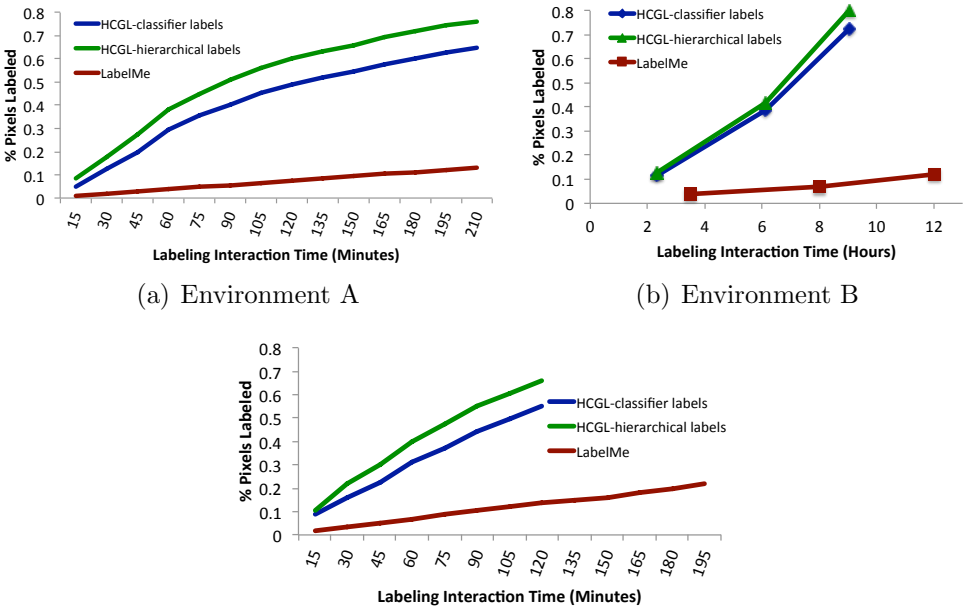


FIGURE 9.1. Relationship of coarse grained labels and classifier labels for environment data. Coarse-grained labels are in bold.



(a) Environment A

(b) Environment B

(c) Environment C

FIGURE 9.2. Rate of hierarchical label assignment to pixels in the training sets.

label assignments using $\hat{\mathcal{Y}}$ (green line). In environments A and C there is about a 10% increase in pixel information. Environment B has fewer hierarchical labels to contribute to the overall labeling of the training data, but the high overall percentage of labeled pixels indicates that HCGL simply made more group selections that best represented labels in $\mathcal{Y}$ for this environment.

Since the relationships between coarse-grained and classifier labels are known, this knowledge can be paired with contextual information about a segment from a multi-concept image

to infer its label from $\mathcal{Y}$. If hierarchical labels can be inferred to more specific concepts, classifiers gain additional information to learn form. This inference can be performed without any additional human labeling effort. In the rest of this chapter we discuss a basic inference model used to establish a larger labeled training set for classifiers.

9.2. Hierarchical Label Inference

Row two in Figure 9.3 shows example training images from the labeling model HCGL-150 used for navigation experiments in environment A. Segments labeled during the 150 minutes of interaction time are colored according to their label assignment. Any part of the image shown as grayscale is still unlabeled. The class-color label key at the bottom of the figure now includes four additional colors to represent the coarse-grained label assignment. Since HCGL uses over-segmentation to create training samples from multi-concept images, true region boundaries (as would be defined using LabelMe) are likely split into multiple smaller segments, meaning neighboring segments often share the same label. Notice that nearly all segments assigned coarse-grained labels in row two have neighboring segments assigned a label from $\mathcal{Y}$. These neighboring segment labels serve as contextual information and provide evidence for possible classifier label inference.

The r segments produced after over-segmenting an image are represented as a symmetric adjacency matrix,

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1r} \\ a_{21} & a_{22} & \cdots & a_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ a_{r1} & a_{r2} & \cdots & a_{rr} \end{pmatrix}$$

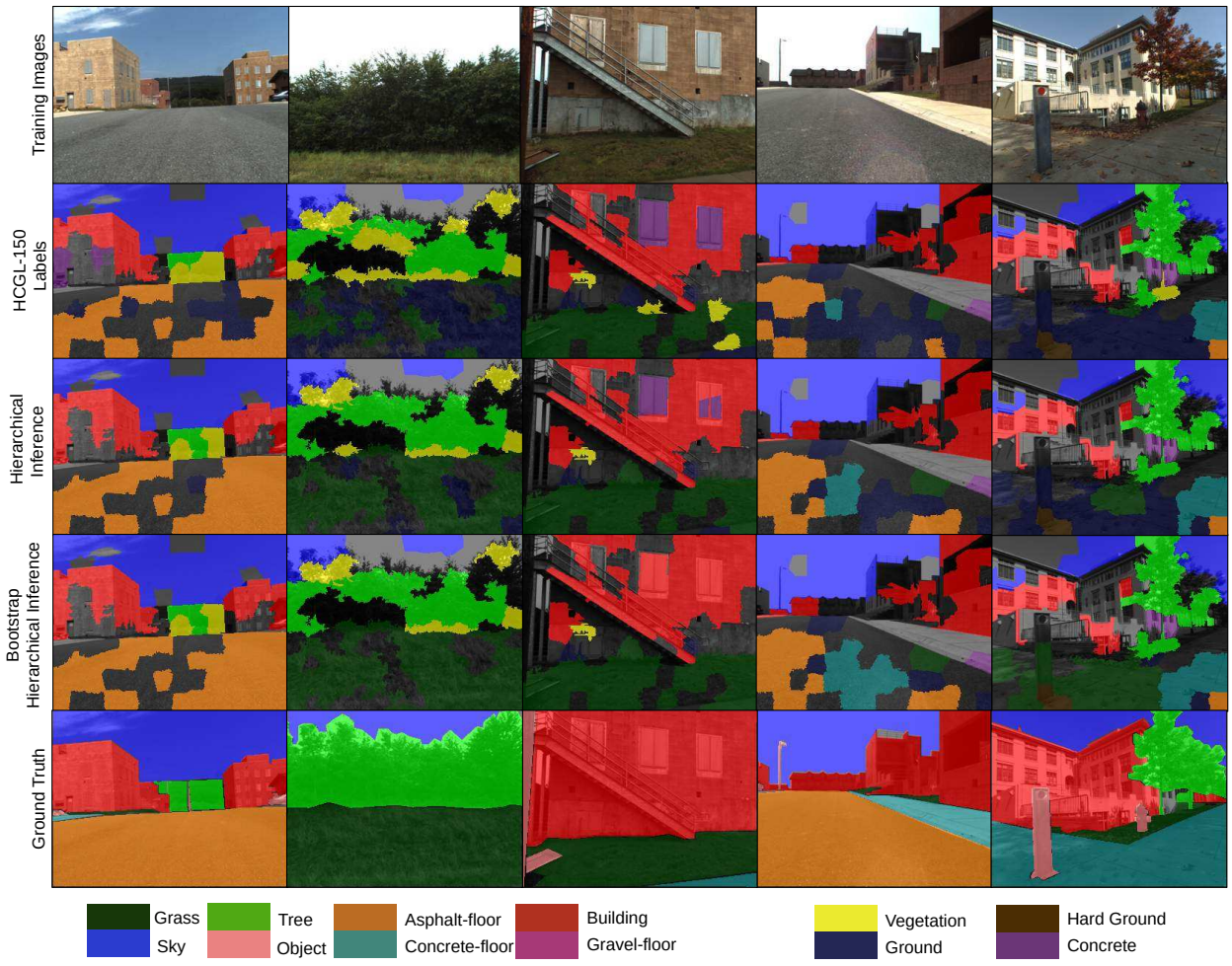


FIGURE 9.3. Five example training images and label assignment after human interaction and hierarchical label inference.

such that $a_{ij} = 1$ if segment i and segment j share at least one boundary pixel, and $a_{ij} = 0$ otherwise. After human labeling interaction, any segment i assigned the label *ground*, *vegetation*, *concrete* or *hard ground*, undergoes hierarchical label inference through a voting process. Every segment j where $a_{ij} = 1$ and $y_j \in \mathcal{Y}$ votes for the inferred label of i using its assigned label y_j. Segment i is assigned the inferred label if at least one neighboring segment voted, the set of votes represents only a single label from $\mathcal{Y}$ and the inferred label does not conflict with the hierarchical label of i, i.e., it is part of the same tree in Figure 9.1.

Row three in Figure 9.3 shows the re-labeling of images when label inference is performed on segments with coarse-grained labels. The percentage of pixels assigned a label from $\mathcal{Y}$

TABLE 9.1. Hierarchical label inference results.

Label Model	% Labeled	Label Accuracy	Classification Accuracy
HCGL-150	0.547	0.949	0.869
Hierarchical Inference	0.601	0.939	0.862
Bootstrap Hierarchical Inference	0.623	0.930	0.867

increases from $\sim 55\%$ to 60% (detailed in Table 9.1). This percentage increase does not come without error. Segments representing the 5% noise caused by majority labeling still vote and can propagate their incorrect label. This can be seen in column four where the *concrete-floor* noise is propagated to adjacent segments because there are no other correctly labeled segments to counteract this evidence. Inference noise is also introduced when a segment that lies on a true region boundary only has labeled neighbors representing a class on the other side of the boundary. This is seen in the last column when a correctly labeled *grass* segment is propagated to *concrete-floor* segments because these two terrains are adjacent. These errors cause overall label accuracy to drop by 1%.

Notice there are still segments with hierarchical labels because their classifier label could not be inferred. However, many of these segments now have new neighboring evidence because of the first inference iteration. Thus, hierarchical label inference can be bootstrapped to iteratively use the previously inferred segments as label votes for the next iteration. Labeling examples from the bootstrapped hierarchical inference can be seen in the fourth row. An additional 2% of the total pixels are assigned labels from $\mathcal{Y}$ at the cost of another small decrease in overall label accuracy.

We train new classifiers with the additional inferred label information. Classification accuracy on the test set from the previous chapter can be seen in the last column of Table 9.1. Unfortunately, the $5 - 9\%$ increase in labeled pixels after inference does not impact the overall classification accuracy. However, in the remainder of this chapter we discuss inference

extensions and why this initial label inference success is still important even though this experiment does not show improved classification performance.

9.3. Discussion

Although classification accuracy did not increase, the hierarchical label inference experiment demonstrated the information available in surrounding context in environment data. This is important for two reasons. First, there is still a classification performance gap when using HCGL-150, which is roughly a half labeled training set, and using a fully annotated training set. Second, since label inference was performed reliably there is potential to introduce this inference earlier in the labeling process and further reduce human interaction time.

9.3.1. Inference Extensions

This thesis mentioned earlier that more labeled training data is better than less. This is specifically true for the HIM used with the environment data since it also uses surrounding context to learn terrains and objects. Thus, the sparsely labeled training images produced by HCGL are not ideal for the HIM learning process. Most of the coarsely labeled segments were assigned a classifier label through the hierarchical inference model, but this inference technique can easily be extended to remaining unlabeled segments in images. The only difference is that there is no hierarchical label to reconcile with neighbor votes.

The five examples show that after hierarchical inference there are many unlabeled regions that are almost completely surrounded by label evidence. Label inference is run for unlabeled segments in a bootstrap fashion, and with no additional human effort the percentage of labeled pixels increases by over 20% to 0.83. As seen with hierarchical inference, this inference

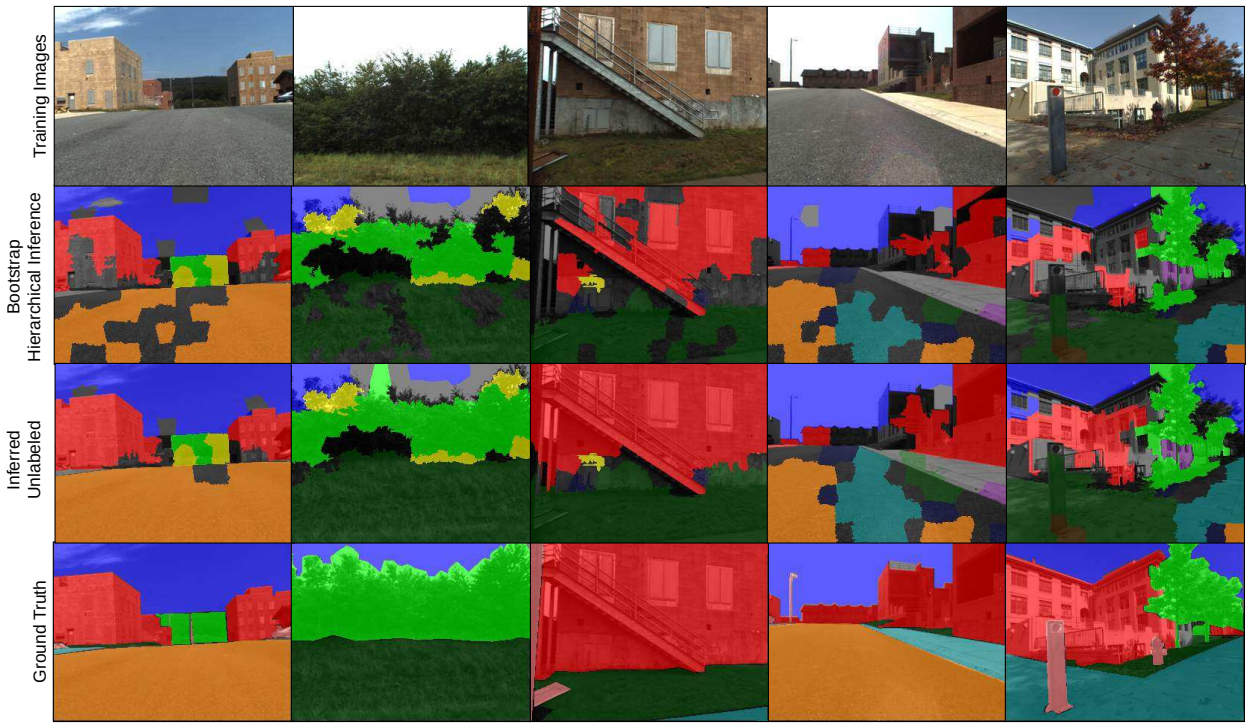


FIGURE 9.4. Five example training images and label inference on unlabeled segments.

comes at a cost and drops overall label accuracy to 0.89. Even with the much larger training set the overall classification accuracy drops slightly to 0.84, suggesting that label noise has reached a point where it outweighs the success seen from simply having a large training set.

Figure 9.4 shows the unlabeled inference results for the five example images. The first three columns show nearly fully labeled images from quality inference results with minimal error, whereas the last two columns look very dissimilar to the ground truth. These examples show that poor inference stems from areas that have little evidence (large areas of unlabeled or coarsely labeled segments) to guide inference logic. Whereas in the correctly inferred situations, segments had multiple adjacent segments with classifier labels. This suggests that if labels are distributed more evenly across images, providing more evidence for each unlabeled or coarsely labeled segment, a large percentage of segments could be labeled automatically with less label noise introduced.

9.3.2. Modifying Selections for Multi-Concept Images

HCGL was not specifically designed to work with multi-concept images and use surrounding context during the labeling process. The only modification required to allow HCGL to generalize to multi-concept images is performed as a pre-processing step: images are over-segmented to create a set of training samples. However, given that the preliminary label inference experiments produced high quality results when segments had multiple neighboring segments to use as evidence, the next logical step is to think about adapting the selection criteria for HCGL when working with multi-concept images.

The three current selection criteria are still relevant, but a fourth criteria, *inference evidence*, could help distribute labeled segments within an image allowing human labeling interaction to terminate earlier and labeling to be finished via inference. For this selection criteria, groups would be ordered by the number of classifier labeled neighboring segments, where those with no surrounding evidence are ranked higher for selection. An in depth analysis of label inference for multi-concept images is beyond the scope of this thesis, but future work will look at taking advantage of contextual information during the selection and labeling process.

CHAPTER 10

Conclusion

10.1. Summary of Research

Research in this thesis is motivated by the time consuming workload allocated to humans to collect large sets of labeled training data for visual classifiers. We presented an efficient label collection framework, Hierarchical Cluster Guided Labeling (HCGL), that reduces labeling workload through group-based labeling. The novelty of HCGL is seen through our hierarchical approach to grouping and label assignment, and our analysis of hierarchical structural change to identify interesting labeling queries.

We motivated the hierarchical framework by demonstrating the difficulty of creating an unsupervised partition of visual data, such that each group contains images representing the same concept. This difficulty arises from low intra-class similarity, high inter-class similarity and the challenge of defining global parameters. Instead, maintaining a hierarchical clustering of data encodes the spectrum of feature similarities found in and between classes in the data, and does not require any global parameters. Our novel hierarchical evaluation of parent to child relationships localizes interesting changes in the underlying feature patterns/structure encoded in the hierarchy. These changes were used to efficiently label a small subset of groups representing the hierarchical spectrum of visual concepts in the data.

Using our hierarchical evaluation for group selection, hierarchical label set assignment and majority group-based labeling allows HCGL to achieve high efficiency at the cost of a small percentage of label noise. We showed that this noise does not significantly compromise classifier learning. Overall, this leads to HCGL producing higher performing classifiers than other labeling techniques, with respect to labeling effort.

Finally, the speed of HCGL was demonstrated by integrating it into a real-world mobile robot application. We showed that HCGL could quickly collect label data for multi-concept images depicting a local environment used for path planning and navigation tasks. Classifiers trained with labels collected by HCGL produced quality visual perception cues for the mobile robot, resulting in successful execution of path planning and navigation along road terrain.

10.2. Future Directions

Although this thesis must come to an end, there are plenty of future directions in which this work can be taken. This section highlights extensions of the research we have presented.

10.2.1. Multi-Concept Images

Classifiers learning from multi-concept images are likely extracting and using the contextual information that surrounds objects in the scene. Thus, fully labeled multi-concept training images are important. In Chapter 9, we discussed ways to extend the label collection of HCGL to produce more densely labeled training data. Specifically, we discussed adding an additional ordering criteria that emphasizes a more evenly distributed labeling of superpixels in a multi-concept image. This extension is motivated by the redundancy captured in image over-segmentation, and seeks to infer the labels of many regions automatically after an initial set of label evidence is provided by a human.

10.2.2. Scaling to Larger Data Volumes

HCGL has been used on a variety of benchmark and real-world datasets of different sizes, but all on the order of thousands of images. With the success of deep learning algorithms [86, 87, 88] that use millions of images to differentiate thousands of object classes, it becomes

apparent that state-of-the-art classifiers used in other applications may require significantly more labeled data than that collected in our thesis experiments.

Unfortunately, in its current state HCGL would struggle with this volume of images. Maintaining a fully constructed hierarchical clustering would require significant memory, and iterative labeling updates (introduced in Chapter 6) with a structure that large would likely display noticeable labeling latency. Future work will look at modifications that can be made to HCGL to handle significantly larger datasets. Two modifications to consider include hierarchical pruning or online hierarchical construction. Both modifications limit the size of the hierarchical structure by either completely removing nodes or only constructing certain subtrees at any given time.

10.2.3. Online Discovery and Labeling

We integrated HCGL into a real-world mobile robot application to reduce labeling latency when visual classifiers must be re-trained to adapt to new environments. However, HCGL is still an offline process performed disjointly from the navigation task. HCGL can only provide labels for the classifier given the set of training data collected prior to robot deployment. In some cases training data may not capture everything in the environment which can cause uncertainty during the navigation task.

In these scenarios it would be ideal to have an online discovery and labeling version of HCGL that could query a human annotator sporadically during the navigation task when visual perception displayed high uncertainty about parts of the environment. This requires HCGL to be more tightly coupled with the visual classifier, but this information can be used to focus labeling attention at locations of the hierarchy were uncertain areas have been grouped together. An online labeling process allows visual classifiers to recognize new objects

or adapt to factors, e.g., occlusion, that were not fully captured in the training set without having the robot abort its task.

Bibliography

[1] P. Jain and A. Kapoor, "Active learning for large multi-class problems," in *Proceedings of Computer Vision and Pattern Recognition*, pp. 762–769, IEEE, 2009.

[2] X. Li and Y. Guo, "Adaptive active learning for image classification," in *Proceedings of Computer Vision and Pattern Recognition*, IEEE, 2013.

[3] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell, "Active learning with gaussian processes for object categorization," in *Proceedings of International Conference on Computer Vision*, pp. 1–8, IEEE, 2007.

[4] A. Holub, P. Perona, and M. C. Burl, "Entropy-based active learning for object recognition," in *Proceedings of Computer Vision and Pattern Recognition Workshops*, pp. 1–8, IEEE, 2008.

[5] A. J. Joshi, F. Porikli, and N. Papanikolopoulos, "Multi-class active learning for image classification," in *Proceedings of Computer Vision and Pattern Recognition*, pp. 2372–2379, 2009.

[6] D. Dai, M. Prasad, C. Leistner, and L. Van Gool, "Ensemble partitioning for unsupervised image categorization," in *Proceedings of European Conference on Computer Vision*, pp. 483–496, Springer, 2012.

[7] Y. J. Lee and K. Grauman, "Object-graphs for context-aware visual category discovery," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 2, pp. 346–358, 2012.

[8] T. Tuytelaars, C. H. Lampert, M. B. Blaschko, and W. Buntine, "Unsupervised object discovery: A comparison," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 284–302, 2010.

[9] A. Bosch, A. Zisserman, and X. Muñoz, "Scene classification via plsa," in *Proceedings of European Conference on Computer Vision*, pp. 517–530, Springer, 2006.

[10] J. Sivic, B. Russell, A. Efros, A. Zisserman, and W. Freeman, "Discovering objects and their location in images," in *Proceedings of International Conference on Computer Vision*, vol. 1, pp. 370–377 Vol. 1, Oct 2005.

[11] C. Wang, D. Blei, and F.-F. Li, "Simultaneous image classification and annotation," in *Proceedings of Computer Vision and Pattern Recognition*, pp. 1903–1910, IEEE, 2009.

[12] C. Xiong, D. M. Johnson, and J. J. Corso, "Spectral active clustering via purification of the k-nearest neighbor graph," in *Proceedings of European Conference on Data Mining*, 2012.

[13] A. Biswas and D. Jacobs, "Active image clustering: Seeking constraints from humans to complement algorithms," in *Proceedings of Computer Vision and Pattern Recognition*, pp. 2152–2159, IEEE, 2012.

[14] A. Gilbert and R. Bowden, "igroup: Weakly supervised image and video grouping," in *Proceedings of International Conference on Computer Vision*, pp. 2166–2173, 2011.

[15] Y. J. Lee and K. Grauman, "Learning the easy things first: Self-paced visual category discovery," in *Proceedings of Computer Vision and Pattern Recognition*, pp. 1721–1728, IEEE, 2011.

[16] C. Galleguillos, B. McFee, and G. Lanckriet, "Iterative category discovery via multiple kernel metric learning," *International Journal of Computer Vision*, vol. 108, no. 1-2, pp. 115–132, 2014.

[17] L. Fei-Fei and P. Perona, "A bayesian hierarchical model for learning natural scene categories," in *Proceedings of Computer Vision and Pattern Recognition*, vol. 2, pp. 524–531, IEEE, 2005.

[18] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International journal of computer vision*, vol. 42, no. 3, pp. 145–175, 2001.

[19] 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 European Conference on Computer Vision*, pp. 1–15, Springer, 2006.

[20] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, 2004.

[21] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[22] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[23] B. Alexe, T. Deselaers, and V. Ferrari, "Measuring the objectness of image windows," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 11, pp. 2189–2202, 2012.

[24] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[25] I. Endres and D. Hoiem, "Category independent object proposals," in *Computer Vision–ECCV 2010*, pp. 575–588, Springer, 2010.

[26] B. C. Russell, W. T. Freeman, A. A. Efros, J. Sivic, and A. Zisserman, "Using multiple segmentations to discover objects and their extent in image collections," in *Proceedings of Computer Vision and Pattern Recognition*, vol. 2, pp. 1605–1614, IEEE, 2006.

[27] B. Settles, "Active learning literature survey," *University of Wisconsin, Madison*, 2010.

[28] B. Siddiquie and A. Gupta, "Beyond active noun tagging: Modeling contextual interactions for multi-class active learning," in *Proceedings of Computer Vision and Pattern Recognition*, pp. 2979–2986, June 2010.

[29] 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*, pp. 86–98, Springer, 2008.

[30] A. J. Joshi, F. Porikli, and N. Papanikolopoulos, "Breaking the interactive bottleneck in multi-class classification with active selection and binary feedback," in *Proceedings of Computer Vision and Pattern Recognition*, pp. 2995–3002, IEEE, 2010.

[31] J. Deng, W. Dong, L.-J. Socher, Richard Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proceedings of Computer Vision and Pattern Recognition*, IEEE, 2009.

[32] A. Sorokin and D. Forsyth, "Utility data annotation with amazon mechanical turk," in *Computer Vision and Pattern Recognition Workshops*, 2008.

[33] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: a database and web-based tool for image annotation," *International journal of computer vision*, vol. 77, no. 1-3, pp. 157–173, 2008.

[34] S. Vijayanarasimhan and K. Grauman, "Large-scale live active learning: Training object detectors with crawled data and crowds," *International Journal of Computer Vision*, vol. 108, no. 1-2, pp. 97–114, 2014.

[35] S. Vijayanarasimhan, P. Jain, and K. Grauman, "Far-sighted active learning on a budget for image and video recognition," in *Proceedings of Computer Vision and Pattern Recognition*, pp. 3035–3042, IEEE, 2010.

[36] L. B. Chilton, G. Little, D. Edge, D. S. Weld, and J. A. Landay, "Cascade: Crowdsourcing taxonomy creation," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1999–2008, ACM, 2013.

[37] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[38] A. Biswas and D. Jacobs, "Active subclustering," *Computer Vision and Image Understanding*, vol. 125, no. 0, pp. 72 – 84, 2014.

[39] Z. Li and J. Liu, "Constrained clustering by spectral kernel learning," in *Proceedings of International Conference on Computer Vision*, pp. 421–427, IEEE, 2009.

[40] N. Grira, M. Crucianu, and N. Boujemaa, "Semi-supervised image database categorization using pairwise constraints," in *Proceedings of International Conference on Image Processing*, vol. 3, pp. III–1228, IEEE, 2005.

[41] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by latent semantic analysis," *JAsIs*, vol. 41, no. 6, pp. 391–407, 1990.

[42] T. Hofmann, "Probabilistic latent semantic indexing," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 50–57, ACM, 1999.

[43] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[44] D. M. Blei, "Probabilistic topic models," *Communications of the ACM*, vol. 55, no. 4, pp. 77–84, 2012.

[45] D. Liu and T. Chen, "Unsupervised image categorization and object localization using topic models and correspondences between images," in *Proceedings of International Conference on Computer Vision*, pp. 1–7, IEEE, 2007.

[46] J. Deng, O. Russakovsky, J. Krause, M. S. Bernstein, A. Berg, and L. Fei-Fei, "Scalable multi-label annotation," in *Proceedings of Human Factors in Computing Systems*, pp. 3099–3102, ACM, 2014.

[47] E. Bart, M. Welling, and P. Perona, "Unsupervised organization of image collections: taxonomies and beyond," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 11, pp. 2302–2315, 2011.

[48] L.-J. Li, C. Wang, Y. Lim, D. M. Blei, and L. Fei-Fei, "Building and using a semantivisual image hierarchy," in *Proceedings of Computer Vision and Pattern Recognition*, pp. 3336–3343, IEEE, 2010.

[49] A. Mahmood, A. Mian, and R. Owens, "Semi-supervised spectral clustering for image set classification," in *Proceedings of Computer Vision and Pattern Recognition*, IEEE, 2014.

[50] J. Sivic, B. C. Russell, A. Zisserman, W. T. Freeman, and A. A. Efros, "Unsupervised discovery of visual object class hierarchies," in *Proceedings of Computer Vision and Pattern Recognition*, pp. 1–8, IEEE, 2008.

[51] M. Wigness, B. A. Draper, and J. R. Beveridge, "Selectively guiding visual concept discovery," in *Proceedings of the Winter Conference on Applications of Computer Vision*, IEEE, 2014.

[52] D. Nettleton, A. Orriols-Puig, and A. Fornells, "A study of the effect of different types of noise on the precision of supervised learning techniques," *Artificial Intelligence Review*, vol. 33, no. 4, pp. 275–306, 2010.

[53] S. Sukhbaatar and R. Fergus, "Learning from noisy labels with deep neural networks," *arXiv preprint arXiv:1406.2080*, 2014.

[54] J. Bootkrajang and A. Kabán, "Label-noise robust logistic regression and its applications," in *Machine Learning and Knowledge Discovery in Databases*, pp. 143–158, Springer, 2012.

[55] T. Leung, Y. Song, and J. Zhang, "Handling label noise in video classification via multiple instance learning," in *International Conference on Computer Vision*, pp. 2056–2063, IEEE, 2011.

[56] B. Frénay and M. Verleysen, "Classification in the presence of label noise: a survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 845–869, 2014.

[57] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *Computer Vision and Image Understanding*, vol. 106, no. 1, pp. 59–70, 2007.

[58] J. H. Ward Jr, "Hierarchical grouping to optimize an objective function," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 236–244, 1963.

[59] M. Wigness, B. A. Draper, and J. R. Beveridge, "Efficient label collection for unlabeled image datasets," in *Proceedings of Computer Vision and Pattern Recognition*, IEEE, 2015.

[60] K. Grauman and T. Darrell, "The pyramid match kernel: Efficient learning with sets of features," *The Journal of Machine Learning Research*, vol. 8, pp. 725–760, 2007.

[61] B. Póczos, L. Xiong, D. J. Sutherland, and J. Schneider, "Nonparametric kernel estimators for image classification," in *Proceedings of Computer Vision and Pattern Recognition*, pp. 2989–2996, IEEE, 2012.

[62] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971.

[63] G. W. Milligan and M. C. Cooper, "An examination of procedures for determining the number of clusters in a data set," *Psychometrika*, vol. 50, no. 2, pp. 159–179, 1985.

[64] E. Dimitriadou, S. Dolničar, and A. Weingessel, "An examination of indexes for determining the number of clusters in binary data sets," *Psychometrika*, vol. 67, no. 1, pp. 137–159, 2002.

[65] A. Ben-Hur and I. Guyon, "Detecting stable clusters using principal component analysis," in *Functional Genomics*, pp. 159–182, Springer, 2003.

[66] S. O'Hara and B. Draper, "Are you using the right nearest neighbor algorithm?," in *Proceedings of the Winter Conference on Applications of Computer Vision*, 2013.

[67] S. O'Hara, B. Draper, *et al.*, "Scalable action recognition with a subspace forest," in *Proceedings of Computer Vision and Pattern Recognition*, pp. 1210–1217, IEEE, 2012.

[68] R. Kümmerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard, "Autonomous robot navigation in highly populated pedestrian zones," *Journal of Field Robotics*, vol. 32,

no. 4, pp. 565–589, 2015.

[69] A. Milella, G. Reina, and J. Underwood, "A self-learning framework for statistical ground classification using radar and monocular vision," *Journal of Field Robotics*, vol. 32, no. 1, pp. 20–41, 2015.

[70] S. Manjanna, G. Dudek, and P. Giguere, "Using gait change for terrain sensing by robots," in *International Conference on Computer and Robot Vision*, pp. 16–22, IEEE, 2013.

[71] P. Giguere and G. Dudek, "Surface identification using simple contact dynamics for mobile robots," in *International Conference on Robotics and Automation*, pp. 3301–3306, IEEE, 2009.

[72] Y. N. Khan, A. Masselli, and A. Zell, "Visual terrain classification by flying robots," in *International Conference on Robotics and Automation*, pp. 498–503, IEEE, 2012.

[73] P. Filitchkin and K. Byl, "Feature-based terrain classification for littledog," in *International Conference on Intelligent Robots and Systems*, pp. 1387–1392, IEEE, 2012.

[74] S. Zenker, E. E. Aksoy, D. Goldschmidt, F. Worgotter, and P. Manoonpong, "Visual terrain classification for selecting energy efficient gaits of a hexapod robot," in *International Conference on Advanced Intelligent Mechatronics*, pp. 577–584, IEEE, 2013.

[75] M. Häselich, M. Arends, N. Wojke, F. Neuhaus, and D. Paulus, "Probabilistic terrain classification in unstructured environments," *Robotics and Autonomous Systems*, vol. 61, no. 10, pp. 1051–1059, 2013.

[76] A. Chatterjee, A. Rakshit, and N. N. Singh, *Vision Based Autonomous Robot Navigation: Algorithms and Implementations*, vol. 455. Springer, 2012.

[77] K. Konolige, M. Agrawal, R. C. Bolles, C. Cowan, M. Fischler, and B. Gerkey, "Outdoor mapping and navigation using stereo vision," in *Experimental Robotics*, pp. 179–190, Springer, 2008.

[78] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. R. Bradski, "Self-supervised monocular road detection in desert terrain.," in *Robotics: Science and Systems*, 2006.

[79] S. Zhou, J. Xi, M. W. McDaniel, T. Nishihata, P. Salesses, and K. Iagnemma, "Self-supervised learning to visually detect terrain surfaces for autonomous robots operating in forested terrain," *Journal of Field Robotics*, vol. 29, no. 2, pp. 277–297, 2012.

[80] A. Lookingbill, J. Rogers, D. Lieb, J. Curry, and S. Thrun, "Reverse optical flow for self-supervised adaptive autonomous robot navigation," *International Journal of Computer Vision*, vol. 74, no. 3, 2007.

[81] D. Summers-Stay, T. Cassidy, and C. R. Voss, "Joint navigation in commander/robot teams: Dialog & task performance when vision is bandwidth-limited," *V&L Net*, p. 9, 2014.

[82] C. Lennon, B. Bodt, M. Childers, R. Camden, A. Suppé, L. Navarro-Serment, and N. Florea, "Performance evaluation of a semantic perception classifier," Tech. Rep. ARL-TR-6653, Army Research Labs, 2013.

[83] M. Wigness, J. G. Rogers III, A. Suppe, L. E. Navarro-Serment, and B. A. Draper, "Reducing adaptation latency for visual perception and navigation." Submitted to 2016 Robotics Conference; Available Upon Request.

[84] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[85] D. Munoz, *Inference Machines: Parsing Scenes via Iterated Predictions*. PhD thesis, The Robotics Institute, Carnegie Mellon University, June 2013.

[86] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, pp. 1–42, 2015.

[87] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[88] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of Computer Vision and Pattern Recognition*, IEEE, June 2015.

APPENDIX A

Variable Reference

This appendix serves as a reference to variables used throughout this thesis which are summarized in Table A.1.

Table A.1. Terminology and variable key

Variable	Description
$\mathcal{Y}$	Label set to be learned by a classifier
K	Assumed number of concepts a classifier will learn
$\hat{\mathcal{Y}}$	Hierarchical label set
x_i	Training sample
$\mathcal{T}$	Set of training data
n	Number of samples in $\mathcal{T}$
m	Number of groups learned by a technique
y_i	Label given to training sample x_i
$\mathcal{H}$	Hierarchical grouping of data
$\mathcal{S}$	Subset of groups from $\mathcal{H}$ to label
$\mathcal{L}$	Set of groups from $\mathcal{H}$ that have been labeled
$\mathcal{U}$	Set of groups from $\mathcal{H}$ that have not been labeled
c	Group in $\mathcal{H}$
p	Parent group of c according to $\mathcal{H}$
c_l, c_r	Left and right children of c according to $\mathcal{H}$
T	number of trees in a Proximity Forest
τ	Splitting threshold for a Proximity Tree
t	Tree in the Proximity Forest
s	Size of a cluster
r	Number of segments in a multi-concept image

APPENDIX B

Selective Guidance: Modeling Coherency Using Classifier Label Set

This appendix includes the details of a group-based labeling technique called Selective Guidance (SG) [51]. Results of SG were used to motivate the work in this thesis, but the overall view of the labeling process is quite different. First, SG uses hierarchical clustering only as a way to establish a set of groups to label, and the relationships encoded in $\mathcal{H}$ are largely ignored. Second, SG seeks to collect noise-less data by modeling features of perfectly coherent groups, i.e., groups of images from exactly one visual concept. Third, labeling is restricted to the end classifier label set, $\mathcal{Y}$. This model uses the stability score PFC described in Chapter 7.

B.1. Coherency Model

The coherency model in SG is used to select groups in $\mathcal{H}$ to label based on a group's predicted likelihood of containing images from a single visual concept. The model is iteratively updated on-line during the labeling process using the PFC scores and sizes of groups that were labeled in previous iterations. As groups are iteratively labeled, the label information is used to help determine the ranges of PFC scores and sizes that are most likely to exhibit coherency. In this model, coherency is modeled in a binary fashion. A group with images from exactly one visual concept is considered coherent, and all other groups are not.

This information is modeled using a 10x10 uniform grid of Gaussian radial basis functions (RBF). One axis of the grid represents the range of PFC scores and the other represents the range of cluster sizes. Each axis is normalized to $[0.0, 1.0]$ with an even distribution of grid

point centers along these axes. Each grid point is modeled as the average weighted coherency of the set of labeled groups. That is, after t labeling queries, grid point g_i has a modeled coherency value of

$$p(g_i) = \frac{1}{t} \sum_{i=0}^{t} \phi(r_i) * v_i, \tag{17}$$

where r_i is the distance between the grid point center and the group queried at iteration i. v_i is 0 if the cluster was not coherent and labeled "mixed" or 1 if it was given a non-mixed label. $\phi(r_i)$ is the Gaussian RBF formally defined as $\exp^{-(r_i/\sigma)^2}$, which weights clusters closer to the center of g_i more heavily than clusters further the center of g_i. The σ model parameter is set to 0.1 for experiments.

The coherency likelihood for group c, is calculated from the RBF grid as

$$p_c = \phi(r_c) * p(g_i), \tag{18}$$

where g_i is selected as the grid point that minimizes r_c.

B.2. Group Selection

Group selection is based on information gain, where information gain is represented by two factors. The exploitation factor seeks to label large numbers of samples quickly, making group coherency and group size important factors when calculating information gain. The exploitation score for cluster c is defined as

$$exploit(c) = p_c * l_c, \tag{19}$$

where p_c is the likelihood that c is coherent and l_c is the number of unlabeled samples in c. Note that l_c is not the size of c since some samples may already be labeled if a descendant of c was selected for labeling in a previous iteration.

The exploitation score describes the expected number of samples that will receive labels if a cluster is given a non-mixed label, and emphasizes labeling as many samples in as few queries as possible. Focusing solely on exploitation, however, favors the discovery of visual concepts that are easy to group and that dominate the data set, possibly disregarding less common concepts.

The exploration factor seeks to discover different visual concepts. Exploration is modeled with the assumption that different visual concepts are located in different areas of feature space. Thus, when selecting the next group to label, it should be far away from coherent groups that have already been labeled, denoted as the set $\mathcal{L}$, to try and identify a new visual concept.

The exploration selectivity score is based on a distance value and defined as

$$explore(c) = \min_{\forall c_i \in \mathcal{L}} d(c_i, c), \tag{20}$$

where d is the Euclidean distance between two group centroids. After two non-mixed labeling queries, unlabeled groups will have multiple distances between labeled groups. The minimum distance from c to any cluster in $\mathcal{L}$ is used since it represents the difference between c and its most similar labeled neighbor. The unlabeled cluster with maximum exploration score represents the cluster that is most dissimilar to its nearest neighbor and expected to be most likely to represent a visual concept that has not been discovered yet.

These selection factors are combined to provide an overall selection criteria score. The two terms are combined using a weight α that has a range $[0, 1]$. Formally, the combination is defined as

$$SG(c) = \alpha * exploit(c) + (1 - \alpha) * explore(c), \tag{21}$$

and the group with the highest selection score is selected to be labeled. For all experiments, the exploitation and exploration terms are weighted evenly by setting $\alpha = 0.5$.