

Copyright  
by  
Adriana Ivanova Kovashka  
2014

The Dissertation Committee for Adriana Ivanova Kovashka  
certifies that this is the approved version of the following dissertation:

## **Interactive Image Search with Attributes**

Committee:

---

Kristen Grauman, Supervisor

---

J. K. Aggarwal

---

Devi Parikh

---

Raymond Mooney

---

Matthew Lease

**Interactive Image Search with Attributes**

by

**Adriana Ivanova Kovashka, B.A.; M.S.Comp.Sci.**

**DISSERTATION**

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

**DOCTOR OF PHILOSOPHY**

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2014

Dedicated to my family.



## Acknowledgments

I would like to thank my adviser Professor Kristen Grauman for helping me achieve all that I've achieved during my PhD career, for encouraging me to apply for a faculty position, and for her continued understanding, patience, and support. I have learned everything that I know about computer vision, writing papers, giving talks, and being professional, from working with Kristen and taking her classes. She has been the best supervisor and mentor I could ever dream of.

My thesis committee members, Professors Ray Mooney, Devi Parikh, Jake Aggarwal, and Matt Lease, have been extremely helpful in brainstorming ideas with me, helping me see new aspects of the problems that my thesis tackles, and offering valuable advice during my job search process.

I have learned a lot from my current and former labmates (Sudheendra, Yong Jae, Jaechul, Sung Ju, Chao-Yeh, Sunil, Suyog, Dinesh, Aron, Bo, Lu Zheng, and Viktoria), and from my ACL co-author Sindhu. I am blessed to have worked in such a friendly, intellectually stimulating environment.

I am also thankful to the Pomona College Computer Science department, where during my undergraduate years I felt like a part of a family, and where I first got the spark of wanting to do research in artificial intelligence. Professor Margaret Martonosi at Princeton University and the CRA-W Dis-

tributed Mentor Program gave me an opportunity to experience research as an undergraduate, and that was instrumental in inspiring me to pursue a PhD.

I am also grateful to the wonderful staff members of the UT Computer Science Department, and particularly to Stacy and Lydia, who have been so patient with my incessant questions and need for administrative help.

I am eternally grateful to my parents for their love, encouragement, and support. They have somehow managed to guide me in the right direction without imposing their beliefs or wishes on me. They have allowed me to make my own choices, and they never minded my occasional teenager moments because they always trusted me. I owe my ever-present belief that things will be alright to them. I am also grateful to my grandparents who helped raise me for the wonderful, carefree, blissful days of my childhood.

My last words of gratitude are reserved for my fiance Logan Gordon. Thank you for being so very patient and loving during my frequent moments of stress, and for never complaining about anything. You have shared all my random and ever-changing interests, and have allowed me to flourish. It is thanks to the happiness you have inspired that I was able to finish this PhD.

# Interactive Image Search with Attributes

Publication No. \_\_\_\_\_

Adriana Ivanova Kovashka, Ph.D.  
The University of Texas at Austin, 2014

Supervisor: Kristen Grauman

An image retrieval system needs to be able to communicate with people using a common language, if it is to serve its user’s information need. I propose techniques for interactive image search with the help of visual attributes, which are high-level semantic visual properties of objects (like “shiny” or “natural”), and are understandable by both people and machines. My thesis explores attributes as a novel form of user input for search. I show how to use attributes to provide relevance feedback for image search; how to optimally choose what to seek feedback on; how to ensure that the attribute models learned by a system align with the user’s perception of these attributes; how to automatically discover the shades of meaning that users employ when applying an attribute term; and how attributes can help learn object category models.

I use attributes to provide a channel on which the user of an image retrieval system can communicate her information need precisely and with as little effort as possible. One-shot retrieval is generally insufficient, so interactive retrieval systems seek feedback from the user on the currently retrieved

results, and adapt their relevance ranking function accordingly. In traditional interactive search, users mark some images as “relevant” and others as “irrelevant”, but this form of feedback is limited. I propose a novel mode of feedback where a user directly describes how high-level properties of retrieved images should be adjusted in order to more closely match her envisioned target images, using relative attribute feedback statements. For example, when conducting a query on a shopping website, the user might state: “I want shoes like these, but *more formal*.” I demonstrate that relative attribute feedback is more powerful than traditional binary feedback.

The images believed to be most *relevant* need not be most *informative* for reducing the system’s uncertainty, so it might be beneficial to seek feedback on something other than the top-ranked images. I propose to guide the user through a coarse-to-fine search using a relative attribute image representation. At each iteration of feedback, the user provides a *visual comparison* between the attribute in her envisioned target and a “pivot” exemplar, where a pivot separates all database images into two balanced sets. The system actively determines along which of multiple such attributes the user’s comparison should next be requested, based on the expected information gain that would result. The proposed attribute search trees allow us to limit the scan for candidate images on which to seek feedback to just one image per attribute, so it is efficient both for the system and the user.

No matter what potentially powerful form of feedback the system offers the user, search efficiency will suffer if there is noise on the communica-

tion channel between the user and the system. Therefore, I also study ways to capture the user’s true perception of the attribute vocabulary used in the search. In existing work, the underlying assumption is that an image has a single “true” label for each attribute that objective viewers could agree upon. However, multiple objective viewers frequently have slightly different internal models of a visual property. I pose *user-specific attribute learning* as an adaptation problem in which the system leverages any commonalities in perception to learn a generic prediction function. Then, it uses a small number of user-labeled examples to adapt that model into a *user-specific* prediction function. To further lighten the labeling load, I introduce two ways to extrapolate beyond the labels explicitly provided by a given user.

While users differ in how they use the attribute vocabulary, there exist some commonalities and groupings of users around their attribute interpretations. Automatically discovering and exploiting these groupings can help the system learn more robust personalized models. I propose an approach to discover the latent factors behind how users label images with the presence or absence of a given attribute, from a sparse label matrix. I then show how to cluster users in this latent space to expose the underlying “shades of meaning” of the attribute, and subsequently learn personalized models for these user groups. Discovering the shades of meaning also serves to disambiguate attribute terms and expand a core attribute vocabulary with finer-grained attributes.

Finally, I show how attributes can help learn object categories faster. I

develop an active learning framework where the computer vision learning system actively solicits annotations from a pool of *both* object category labels and the objects' shared attributes, depending on which will most reduce total uncertainty for multi-class object predictions in the joint object-attribute model. Knowledge of an attribute's presence in an image can immediately influence many object models, since attributes are by definition shared across subsets of the object categories. The resulting object category models can be used when the user initiates a search via keywords such as "Show me images of cats" and then (optionally) refines that search with the attribute-based interactions I propose.

My thesis exploits properties of visual attributes that allow search to be both effective and efficient, in terms of both user time and computation time. Further, I show how the search experience for each individual user can be improved, by modeling how she uses attributes to communicate with the retrieval system. I focus on the modes in which an image retrieval system communicates with its users by integrating the computer vision perspective and the information retrieval perspective to image search, so the techniques I propose are a promising step in closing the semantic gap.

# Table of Contents

<b>Acknowledgments</b>	<b>v</b>
<b>Abstract</b>	<b>vii</b>
<b>List of Tables</b>	<b>xv</b>
<b>List of Figures</b>	<b>xvi</b>
<b>Chapter 1. Introduction</b>	<b>1</b>
1.1 Comparative Relevance Feedback using Attributes . . . . .	5
1.2 Actively Guiding the User’s Relevance Feedback . . . . .	7
1.3 Accounting for Differing User Perceptions of Attributes . . . . .	11
1.4 Discovering Attribute Shades of Meaning . . . . .	14
1.5 Using Attributes To Help Learn Object Category Models . . . . .	17
<b>Chapter 2. Related Work and Background</b>	<b>22</b>
2.1 Visual Attributes . . . . .	22
2.1.1 Learning Visual Attribute Models . . . . .	23
2.1.2 Defining Attribute Vocabularies . . . . .	25
2.1.3 Discovering Non-Semantic Attributes . . . . .	26
2.1.4 Subjectivity in Attribute Perception . . . . .	26
2.1.5 Attributes for Object Recognition . . . . .	27
2.1.6 Attributes for Image Search . . . . .	28
2.2 Relevance Feedback in Image Search . . . . .	30
2.2.1 Active Selection of Solicited Feedback . . . . .	31
2.2.2 Personalization of Search . . . . .	32
2.3 Active Selection for Learning Object Categories . . . . .	33
2.3.1 Active Learning for Classification . . . . .	33
2.3.2 Active Testing and Twenty Questions . . . . .	35

2.4	Learning Visual Categories with Domain Adaptation . . . . .	36
2.5	Aggregating Crowd Labels . . . . .	37
2.6	Polysemy of Words in Images . . . . .	38
<b>Chapter 3. WhittleSearch: Image Search with Relative Attribute Feedback</b>		<b>40</b>
3.1	Background: Binary Relevance Feedback . . . . .	42
3.2	Learning to Predict Relative Attributes . . . . .	43
3.3	Relative Attribute Feedback . . . . .	47
3.4	Hybrid Feedback Approach . . . . .	52
3.5	Experimental Validation . . . . .	54
3.5.1	Experimental Design . . . . .	54
3.5.2	Impact of Iterative Feedback . . . . .	60
3.5.3	Impact of Amount of Feedback . . . . .	62
3.5.4	Impact of Reference Images . . . . .	63
3.5.5	Ranking Accuracy with User-Given Feedback . . . . .	64
3.5.6	Consistency of Relative Supervision Types . . . . .	66
3.6	Conclusions . . . . .	68
<b>Chapter 4. Active WhittleSearch: Attribute Pivots for Guiding Relevance Feedback in Image Search</b>		<b>69</b>
4.1	Attribute Binary Search Trees . . . . .	71
4.2	Predicting the Relevance of an Image . . . . .	72
4.3	Actively Selecting an Informative Comparison . . . . .	75
4.3.1	Entropy Reduction Objective . . . . .	75
4.3.2	User Response Likelihood . . . . .	76
4.3.3	Recap of Interaction Loop . . . . .	78
4.3.4	Discussion . . . . .	80
4.4	Conceptual Comparison of Free-form and Active WhittleSearch	81
4.5	Experimental Validation . . . . .	84
4.5.1	Experimental Design . . . . .	84
4.5.2	Baselines . . . . .	86
4.5.3	Results with Feedback by Simulated Users . . . . .	87



4.5.4	Results with Live Users . . . . .	92
4.5.5	Experimental Comparison of Free-form and Active Whit- tleSearch . . . . .	96
4.6	Conclusions . . . . .	100
<b>Chapter 5. Attribute Adaptation for Personalized Image Search</b>		<b>103</b>
5.1	Learning Adapted Attributes . . . . .	105
5.1.1	Adapting Binary Attribute Classifiers . . . . .	106
5.1.2	Adapting Relative Attribute Rankers . . . . .	108
5.1.3	Suitability for Adapted Attributes . . . . .	109
5.2	Personalized Image Search with Adapted Attributes . . . . .	111
5.3	Obtaining User-Specific Labeled Data . . . . .	111
5.3.1	Explicit Collection . . . . .	112
5.3.2	Implicit Collection . . . . .	114
5.4	Experimental Validation . . . . .	117
5.4.1	Experimental Design . . . . .	117
5.4.2	Baselines . . . . .	118
5.4.3	Adapted Attribute Accuracy . . . . .	120
5.4.4	Personalized Search with Adapted Attributes . . . . .	127
5.5	Conclusions . . . . .	130
<b>Chapter 6. Discovering Shades of Attribute Meaning with the Crowd</b>		<b>131</b>
6.1	Collecting Crowd Labels per Attribute . . . . .	134
6.2	Recovering Latent Factors for Attribute Labels . . . . .	137
6.3	Discovering Shades of Meaning . . . . .	139
6.4	Using Shades to Predict Perceived Attributes . . . . .	141
6.5	Discussion . . . . .	143
6.6	Experimental Validation . . . . .	144
6.6.1	Experimental Design . . . . .	145
6.6.2	Accuracy of Perceived Shade Predictions . . . . .	145
6.6.3	Improved Personalized Search with Shades . . . . .	149
6.6.4	Quantifying the Accuracy of Shade Formation . . . . .	151

6.6.5	Visualizing Attribute Shades of Meaning . . . . .	154
6.6.6	Exploiting Attribute Correlations for Transfer . . . . .	158
6.7	Conclusions . . . . .	161
<b>Chapter 7.</b>	<b>Actively Selecting Annotations Among Objects and Attributes</b>	<b>162</b>
7.1	Object-Attribute Model . . . . .	165
7.2	Active Learning with Objects and Attributes . . . . .	168
7.2.1	Annotation Set Definitions . . . . .	170
7.2.2	Entropy-Based Selection Function . . . . .	171
7.2.3	Updates to Labeled and Partially Labeled Sets . . . . .	173
7.3	Experimental Validation . . . . .	176
7.3.1	Experimental Design . . . . .	176
7.3.2	Baselines . . . . .	177
7.3.3	Results and Discussion . . . . .	179
7.4	Conclusions . . . . .	183
<b>Chapter 8.</b>	<b>Future Work</b>	<b>185</b>
<b>Chapter 9.</b>	<b>Conclusion</b>	<b>189</b>
	<b>Bibliography</b>	<b>192</b>
	<b>Vita</b>	<b>215</b>

## List of Tables

3.1	A list of the attribute names for the Shoes, OSR, and PubFig datasets . . . . .	55
3.2	Impact of the type of reference images in WhittleSearch . . . .	63
3.3	Errors for class- vs. instance-level attribute training . . . . .	67
4.1	Selection time for active WhittleSearch vs. the exhaustive active baseline . . . . .	91
5.1	The Shoes and SUN attributes used in my experiments . . . .	118
5.2	Personalized image search accuracy: Multi-attribute keyword search . . . . .	128
5.3	Personalized image search accuracy: Relative attribute search feedback . . . . .	128
5.4	The benefit of inferring implicit user-specific labels . . . . .	129
6.1	Discovering attribute shades of meaning: The 12 attribute definitions shown to annotators . . . . .	135
6.2	Accuracy of predicting perceived attributes using attribute shades	147
6.3	Multi-attribute query image search accuracy using shades . . .	150
6.4	Attribute shades: Illustration of significance of topic entropy reduction . . . . .	155
6.5	Accuracy of imputing missing labels using other attributes . .	160

## List of Figures

1.1	WhittleSearch allows users to refine their image search via relative attribute feedback . . . . .	6
1.2	The active version of WhittleSearch requests visual attribute comparisons between the user’s target and images selected by the system . . . . .	9
1.3	Visual attribute interpretations vary across viewers . . . . .	12
1.4	My attribute shade discovery method uses the crowd to discover factors responsible for an attribute’s presence, then learns predictive models based on those visual cues . . . . .	16
1.5	Object and attribute label requests affect object category models in different ways . . . . .	18
1.6	Overview of the work in this thesis . . . . .	20
3.1	Interface for image-level relative attribute annotations . . . . .	44
3.2	A toy example illustrating the intersection of relative attribute feedback constraints with $M = 2$ attributes . . . . .	50
3.3	Example images from the Shoes, OSR, and PubFig datasets . . . . .	56
3.4	Impact of iterative feedback in WhittleSearch . . . . .	61
3.5	Impact of the amount of feedback in WhittleSearch . . . . .	62
3.6	Ranking accuracy with user-generated feedback . . . . .	64
3.7	Example search result with relative attribute feedback . . . . .	65
3.8	Example search result with hybrid feedback . . . . .	66
4.1	Sketch of active selection interaction loop . . . . .	79
4.2	Comparison of the proposed models for the likelihood of a user’s response in active WhittleSearch . . . . .	89
4.3	Comparison of existing interactive search methods and active WhittleSearch . . . . .	90
4.4	Results from active WhittleSearch live searches done by MTurk users . . . . .	94

4.5	Example of an active WhittleSearch live search done by an MTurk user . . . . .	95
4.6	Quantitative comparison of free-form and active WhittleSearch: Entropy and rank of target . . . . .	97
4.7	Quantitative comparison of free-form and active WhittleSearch: Total time and response confidence . . . . .	98
4.8	Qualitative comparison of free-form and active WhittleSearch	101
5.1	MTurk qualification test for relative annotations on Shoes . .	113
5.2	Illustration of my idea for extracting implicit user-specific labels	116
5.3	Example images from the SUN dataset . . . . .	119
5.4	Adapted attribute prediction accuracy: Average performance and standard error over all attributes . . . . .	121
5.5	Adapted attribute prediction accuracy: Individual per-attribute per-user plots . . . . .	122
5.6	Example learned generic and user-specific attribute spectra . .	125
5.7	Adapted attribute accuracy as a function of task difficulty . .	126
6.1	Illustration of my attribute shade discovery approach . . . . .	139
6.2	Example label explanations that annotators provided . . . . .	146
6.3	Confusion matrices for multi-way shade classification . . . . .	148
6.4	Quality of discovered attribute shades . . . . .	153
6.5	Top words and images for two shades per attribute . . . . .	156
6.6	Image regions that are important for learning the attribute shades	158
7.1	Overview of active selection with object and attribute annotations	169
7.2	Example images from Animals with Attributes, a-Pascal, and a-Yahoo . . . . .	178
7.3	Active learning with objects and attributes: Representative learning curves from all three datasets . . . . .	180
7.4	Active learning with objects and attributes: Entropy of all examples with increasing number of labels . . . . .	181
7.5	Distribution of requests per label type (object/attributes). . .	182
7.6	Sample ⟨image, label⟩ requests for active learning with objects and attributes . . . . .	183

# Chapter 1

## Introduction

It is difficult to develop a machine vision system that perceives the world like a person does. However, an image retrieval system needs to be able to communicate with its user using a common language, if it is to serve the user's information need. When people perform a search, they usually have a very specific idea of what they want to retrieve, and this idea cannot be captured by simple tags or keywords, which are usually category labels. Users might want to see what a given celebrity looks like, in which case the celebrity can be treated as a category, and the search can be seen as a classification problem. However, in most cases, the traditional categories we use in computer vision are insufficiently descriptive of the user's information need since categories are too coarse-grained. For example, a user might want to buy shoes that satisfy certain properties like color, heel height, texture etc., and these properties cannot be captured by even the most fine-grained categories that might reasonably exist in the world. The user might also search for stock photography to include in a presentation, and she likely has a very specific idea of what the photograph she wants to include should look like.

Keywords alone are not sufficient to capture the user's mental picture

of what she is looking for. Even if all existing images were tagged to enable keyword search, it is infeasible to pre-assign tags sufficient to satisfy any future query a user may dream up. Furthermore, due to the well known semantic gap—which separates the system’s low-level image representation from the user’s high-level concept—one-shot retrieval performed by matching images to keywords is unlikely to get the right results. One of the solutions which retrieval systems employ to solve the problems caused by the semantic gap is to allow the user to iteratively provide feedback on the results retrieved in each round. The basic idea is to show the user candidate results, obtain feedback, and adapt the system’s relevance ranking function accordingly. In this interactive form of search, users mark some images as “relevant” and others as “irrelevant” [86, 23, 123, 182, 43, 152, 89]. Instead of requesting feedback on some user-chosen set of the current results, some methods perform active selection of the images to display for feedback, by exploiting the uncertainty in the system’s current model of relevance to find useful exemplars [152, 89, 23, 182, 43].

However, this form of feedback is limited as it forces the retrieval system to guess *what about* the images was relevant or irrelevant. For example, when Jane searches for “black shoes”, retrieves a pair of pointy high-heeled black shoes, and marks them as irrelevant, this might be because she did not want these shoes to be “pointy”, or because she wanted them to be “flat”. However, the system does not know which, and this uncertainty will negatively impact the next page of image results. Furthermore, existing methods which *actively*

*select* the images for feedback use an *approximation* to finding the optimal uncertainty reduction, whether in the form of uncertainty sampling [152] or by employing sampling or clustering heuristics [23, 43]. Such methods also only consider binary feedback (“this is relevant” / “this is irrelevant”), which is imprecise.

Recently, there has been work in the computer vision community on developing models for visual *attributes*, which are semantic properties of objects that often extend across category boundaries [87, 40, 45, 107]. Some examples of attributes are “furry”, “metallic”, “pointy”, “young”, and “smiling”. Some researchers have explored attributes for search [83, 142, 134], but even though one-shot attribute queries help a user more precisely state their goal, they are still a form of keyword search and do not allow refinement of the search results. Further, systems which retrieve results based on multi-attribute queries assume that all users mean the same thing when they make a certain attribute statement, and that one classifier is sufficient to capture all variability within a given attribute term [83, 142, 134]. However, other researchers and myself find that there is substantial disagreement between users regarding attribute labels [40, 110, 24], and that different groups of users have different “shades of meaning” in mind when they employ adjectives such as colors [37].

The central idea of my thesis is to explore visual attributes for semantic feedback in interactive image search. Attributes can be either binary or relative properties. In the first case, to learn an attribute model, we learn a binary classifier which predicts whether the attribute is present or not. In the second



case, we learn a ranking model which predicts the relative strength of the attribute in a given image. This model allows us to rank images on a spectrum from “least” to “most” having the attribute. I will consider both binary and relative attributes in this thesis.

Towards the broad goal of interactive search with attributes, I address a number of technical challenges. First, I use attributes to provide a channel on which the user can communicate her information need precisely and with as little effort as possible. I find that attributes enable more powerful relevance feedback for image search compared to traditional binary feedback (“This image is relevant; this one is not.”), and show how to further select this feedback so it is as informative as possible. I also investigate how users use the attribute vocabulary during search, and ensure that the models learned for each attribute align with how a user employs the attribute name, which is determined by the user’s individual perception of this attribute. I propose to automatically discover and exploit the commonalities that exist in user perceptions of the same attribute, to reveal the “shades of meaning” of an attribute and learn more robust models which are personalized for groups of users. Finally, I use attributes in a joint object-attribute model to efficiently learn object category models which can be used when a person initiates a search with keywords.

Unlike existing relevance feedback for image retrieval [86, 123, 151, 23, 182, 43, 47], the attribute-based feedback I propose allows the user to communicate to the retrieval system precisely *how* a set of results lack what the

user is looking for. Furthermore, unlike existing work in attribute-based search [83, 142, 134, 117], I ensure that the user input to the system will be interpreted as intended, by developing a new form of personalization that decreases the noise on the user-system communication channel. Further, my approach differs from existing work on attribute-based search that treats attributes as keywords [83, 142, 134], so visual properties are considered to be either present or not present, with no way to quantify to what extent or in what way they are present. In that work, once a search is performed, there is no way to refine the query. Instead I show how to use attributes to refine a search.

## 1.1 Comparative Relevance Feedback using Attributes

In the the first major component of my thesis, I propose a novel mode of feedback where a user directly describes how high-level properties of exemplar images should be adjusted in order to more closely match her envisioned target images. For example, when conducting a query on a shopping website, the user might state: “I want shoes like these, but *more formal*.” When browsing images of potential dates on a dating website, she can say: “I am interested in someone who looks like this, but with *longer hair* and *more smiling*.” When searching for stock photos to fit an ad, she might say: “I need a scene *similarly bright* as this one and *more urban* than that one.” See Figure 1.1. In this way, rather than simply state which images are (ir)relevant, the user employs semantic terms to say *how* they are so. I expect that such feedback will enable the system to more closely match the user’s mental model of the desired con-

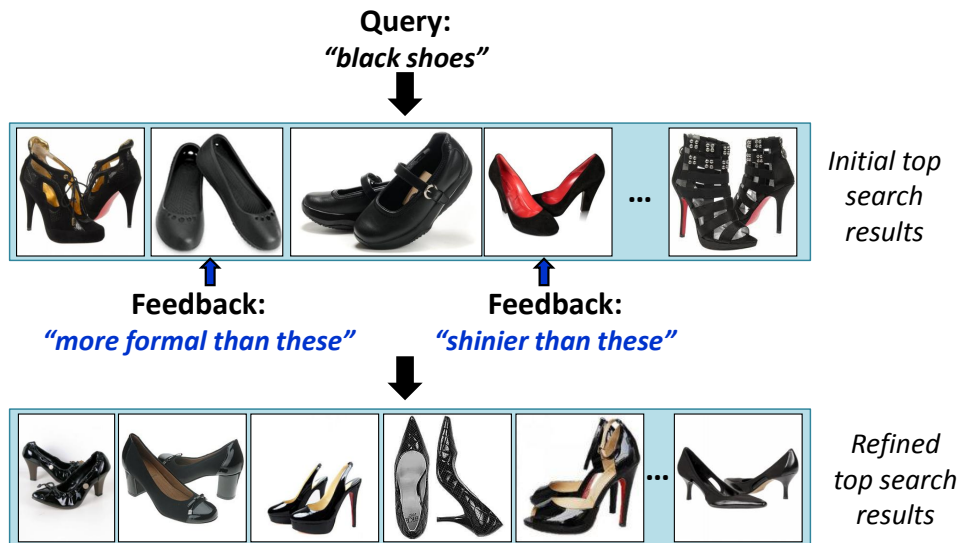


Figure 1.1: WhittleSearch allows users to give relative attribute feedback on reference images to refine their image search.

tent, and with less total interaction effort. I call the approach *WhittleSearch*, since it allows users to “whittle away” irrelevant portions of the visual feature space via precise, intuitive statements of their attribute preferences.

In order for a retrieval system to accept relative attribute feedback, it needs to know how images place along each attribute spectrum. Following [107], the system learns one ranking model per attribute, which is a one-time procedure that takes place before any search begins. The system then needs a method for updating its notion of relevance from relative attribute feedback. In Chapter 3, I propose a simple strategy which counts how many relative attribute feedback constraints each image satisfies, and then in Chapter 4, I extend this approach by computing the probability that an image satisfies a

given set of constraints, thereby accounting for uncertainty in the attribute predictions.

During search, a user sees a whole page of results, so feedback can be given on any image of the user's choosing. To match this scenario, the system presents the user with a set of reference images and allows her to pair any of these with any attribute in the vocabulary. This setup gives the user the most freedom to comment on exactly what she finds important for achieving good image results. In all but the first iteration, the presented reference images are those currently ranked best by the system, which has the additional advantage that the user is shown only those results which are relatively similar to (the system's estimate of) her target image. My results show that the proposed form of feedback via relative attributes is more powerful than traditional binary relevance feedback.

## 1.2 Actively Guiding the User's Relevance Feedback

However, the images believed to be most *relevant* need not be most *informative* for reducing the system's uncertainty. As a result, when the user is willing to cooperate with the system in order to achieve better final results, it might be more beneficial to leave the choice of reference images on which to seek feedback to the system. Therefore, in Chapter 4, I study how the system can best select the feedback it requests.

The goal of *actively* selecting some images for feedback is to solicit feedback on those exemplars that would most improve the system's notion of

relevance. Many existing methods exploit classifier uncertainty to find useful exemplars (e.g., [152, 89, 182]). However, traditional approaches have two main limitations. First, these methods elicit traditional binary feedback (“Image X is relevant; image Y is not.”) which is imprecise, as discussed above. This makes it ambiguous how to extrapolate relevance predictions to other images, which in turn clouds the active selection criterion. Second, existing active selection techniques add substantial computational overhead to the interactive search loop, since ideally they must scan all database images to find the most informative exemplars. This is why prior efforts to display the exemplar set that minimizes uncertainty were forced to resort to sampling or clustering heuristics due to the combinatorial optimization problem inherent when categorical feedback is assumed (e.g., [122, 23, 43]), or to the over-simplified uncertainty sampling [152] which does not guarantee global uncertainty reduction over the full dataset.

In the second major component of my thesis, I introduce a novel approach that addresses these shortcomings. I propose to actively guide the user through a coarse-to-fine search using a relative attribute image representation. At each iteration of feedback, the user provides a *visual comparison* between the attribute in her envisioned target and a “pivot” exemplar, where a pivot separates all database images into two balanced sets. In the previous form of relevance feedback I proposed, the user is presented with a full page of image results, and has the freedom to choose both the image to which she will compare her mental model, and the attribute along which the comparison will be



Figure 1.2: The active version of WhittleSearch requests feedback in the form of visual attribute comparisons between the user’s target and images selected by the system. To formulate the optimal question to ask next, it unifies an entropy reduction criterion with binary search trees in attribute space.

made. I now propose an additional form a feedback where the system makes this choice, so the user is presented with a single image and a single attribute and simply has to provide the value of the comparison (“more”, “less”, or “equally”). The system actively determines along which of multiple attributes the user’s comparison should next be requested, based on the expected information gain that would result. This ensures that the system receives useful feedback, and also can be advantageous if a user finds the choice of which images and attributes to comment on burdensome.

The approach works as follows. Given a database of images, my system first constructs a binary search tree for each relative attribute of interest (e.g., “pointiness”, “shininess”, etc.). Initially, the pivot exemplar for each attribute is the database image with the median relative attribute value. Starting at the roots of these trees, the system predicts the information gain that would result from asking the user how his target image compares to each of the current

pivots. To compute the expected gain, I introduce methods to estimate the likelihood of the user’s response given the feedback history. Then, among the pivots, the most informative comparison is requested, generating a question to the user such as, “Is your target image more, equally, or less pointy than this image?” Following the user’s response, the system updates its relevance predictions on all images. It also moves the current pivot down one level within the selected attribute’s tree (unless the response is “equally”, in which case we no longer need to explore this tree). The procedure iterates until the user is satisfied with the top-ranked results. Please see Figure 1.2 for an illustration of the key idea of this approach.

In technical terms, this problem setting demands repeatedly estimating the total expected error reduction over all unlabeled database images, as a function of requesting any possible comparison from the user. Whereas prior information-gain methods would require a naive scan through all database images for each iteration, the proposed attribute search trees allow us to limit the scan to just one image per attribute. Thus, the active selection method I propose is efficient both for the system (which analyzes a small number of candidates per iteration) and the user (who locates his content via a small number of well-chosen interactions). My results show that the proposed method retrieves more accurate results and makes its choices faster than relevant baselines.

### 1.3 Accounting for Differing User Perceptions of Attributes

No matter what potential power of feedback we offer a user, search efficiency will suffer if there is noise on the communication channel between the user and the system. In other words, if the user says “A” and the system understands “B”, most searches will be unsuccessful. In existing work, training an attribute predictor largely follows the same procedure used for training any image classification system: one collects labeled image exemplars, extracts image descriptors, and applies discriminative learning. The underlying assumption is that an image has a single “true” category label that objective viewers could agree upon. Yet, while this holds for objects (a horse is a horse, of course<sup>1</sup>), an attribute inherently has more leeway. Multiple objective viewers are bound to have slightly different internal models of a visual property. Indeed, researchers collecting attribute-labeled datasets report significant disagreement among human annotators [40, 110, 35].

The differences may stem from several factors: the words for attributes are imprecise (when is the cat “overweight” vs. “chubby”?), their meanings often depend on context (the shoe appears “comfortable” for a wedding, but not for running) and even cultures (languages have differing numbers of color words, ranging from two to eleven), and they often stretch to refer to quite distinct object categories (e.g., “pointy” pencil vs. “pointy” shoes). For all

---

<sup>1</sup>See [104] for work on predicting which entry-level noun users commonly agree upon and use to describe an object.





Figure 1.3: Visual attribute interpretations vary slightly from viewer to viewer. This is true whether the attributes are modeled as categorical or relative properties. For example, 5 viewers *confidently* declare a shoe as formal (left) or more ornamented (right), while 5 others confidently declare the opposite! I propose to adapt attribute models to take these differences in perception into account.

such reasons, people inevitably craft their own definitions for visual attributes. Notably, their definitions vary whether we consider binary or relative attributes (see Figure 1.3).

This variability has important implications for any application where a person uses attributes to communicate with a vision system, and particularly for image search. Failing to account for user-specific notions of attributes will lead to discrepancies between the user’s precise intent and the message received by the system. Yet, even when training labels are solicited from multiple annotators, existing methods learn only a single “mainstream” view of each attribute, forcing a consensus through majority voting. This is the case whether using binary [45, 87, 40] or relative [107] attributes. For binary properties, one takes the majority vote on the attribute present/absent label. For relative properties, one takes a majority vote on the attribute more/less label. Note that using relative attributes does not resolve the ambiguity problem. The point in relative attributes is that people may agree best on comparisons

or strengths, not binary labels. However, just like categorical attributes, relative attributes assume that there is some single, common interpretation of the property shared consistently by all viewers—namely, that a single ordering of images from least to most [attribute] is possible.

In the third major component of my thesis, I study ways to capture the user’s true perception of the attribute vocabulary used in the search. I propose to model attributes in a user-specific way, in order to capture the inherent differences in perception. How can this be done efficiently? The most straightforward approach—to learn one function per attribute and per user, from scratch—is certainly not scalable in most reasonable application settings, and ignores the reality that people do share *some* foundational definition of a visual property.

Instead, I pose user-specific attribute learning as an *adaptation* problem. First, my system leverages any commonalities in perception to learn a *generic* prediction function, namely, a classifier for a binary attribute (e.g., “pointy”) or a ranking function for a relative attribute (e.g., “pointier than”). Then, it uses a small number of user-labeled examples to adapt that model into a *user-specific* prediction function. In technical terms, this amounts to imposing regularizers on the learning objective favoring user-specific model parameters that are similar to the generic ones, while still satisfying the user-specific label constraints [177, 50].

To further lighten the user’s labeling load, I introduce two ways to extrapolate beyond the labels explicitly provided by a given user. In the

first, the system connects relative attribute statements given by the user on multiple different images to obtain new implicit constraints via transitivity. In the second, it detects discrepancies between the system’s generic attribute models and the user’s perception, and creates implicit constraints to correct the models. Both ideas serve to generate additional plausible user-specific labels without directly requesting more labels from the user.

While adapted attributes are applicable to any task demanding precise human-system communication about visual properties, I focus specifically on their impact for image search. I demonstrate the advantages of personalized retrieval when a user queries for images with multi-attribute keywords or uses attributes to provide relevance feedback on selected reference images. In this context, I demonstrate that a user’s search history offers a natural source of data for inferring user-specific labels. My results show that the learned user-adaptive models align more closely with individual users’ perceptions of attributes, and enable more accurate results to be retrieved, compared to both generic models and ones learned solely from the given user’s data.

## **1.4 Discovering Attribute Shades of Meaning**

So far, I have discussed generic attribute models, which assume that all users perceive the attribute in the same way; and user-specific models, which assume that each user’s perception is unique. However, one could also consider a middle ground between the universal and personalized models. While users differ in how they perceive and use attributes, it is likely that there are some

commonalities between users in terms of how they interpret and utilize the attribute vocabulary. In other words, there might be groupings among users with respect to how they use a given attribute term. If a system can automatically discover these groupings, which I refer to as “schools of thought”, then it can personalize attribute models towards these schools, rather than towards individual users. Relying on the perception of a school as opposed to of an individual user can help avoid over-personalization, which might be misled by noise in a user’s annotations. Focusing on the commonalities allows the system to learn the important biases that users have in interpreting the attribute, as opposed to minor differences in labeling which may stem from factors other than a truly different interpretation.

If schools of thought do exist among users, they will be based on the slightly different understanding or use of the attribute term that users in each school of thought might have. In other words, each school will subscribe to a slightly different “shade” of the attribute. Thus, by discovering schools of thought, a system also discovers the shades of meaning that a given attribute contains. For example, for the attribute “open” in Figure 1.4, it might discover that some users have peep-toed shoes in mind when they say “open”, while others might have flip-flops in mind when they use the same word.

The shade discovery method I propose allows for an attribute vocabulary to be expanded. Existing approaches for automatic attribute discovery are quite different from what I propose. Unsupervised discovery methods detect clusters or splits in the low-level image descriptor space [106, 96, 33, 116, 138,



Figure 1.4: My attribute shade discovery method uses the crowd to discover factors responsible for an attribute’s presence, then learns predictive models based on those visual cues. For example, for the attribute *open*, the method will discover shades of *meaning*, e.g., peep-toed (*open* at toe) vs. slip-on (*open* at heel) vs. sandal-like (*open* at toe *and* heel), which are three visual definitions of openness. Since these shades are not coherent in terms of their global descriptors, they would be difficult to discover using traditional image clustering.

178]. While they might discover finer-grained shades of *some* property, they need not be nameable by people (semantic). Furthermore, discovery methods are intrinsically biased by the choice of features. For example, the set of salient splits in color histogram space will be quite different than those discovered in a dense SIFT feature space. Similarly, unsupervised methods that cluster global descriptors have no way to intelligently focus on only localized regions of the image, yet an attribute may occupy an arbitrarily small part of an image.

In the fourth component of my work, my goal is to automatically discover the shades of an attribute. An attribute “shade” is a visual interpretation of an attribute name that one or more people apply when judging whether that attribute is present in an image. Similarly, if learning relative attributes, a shade is an interpretation when judging whether that attribute is present more in image A or image B. In order to discover shades, the approach I propose

relies on a set of sparse annotations from a large pool of users, and performs matrix factorization over these labels to discover the underlying factors contributing to the annotations. After finding these latent factors, the method can cluster users (to discover “schools of thought”) or images (to discover the images representative of a “shade”) in this latent factor space.

On two datasets, I find that not only are the discovered shades visually meaningful, but they are also well-aligned with annotators’ textual explanations of their labels. Most importantly, I show their practical utility to reliably estimate perceived attributes in novel images, which is crucial for any application relying on the descriptive nature of attributes (e.g., image search or zero-shot learning).

## **1.5 Using Attributes To Help Learn Object Category Models**

So far, I have studied techniques for providing precise relevance feedback for image search using attribute constraints, as well as ensuring the accuracy of this feedback by learning user-specific or shaded attribute models. In the final component of my thesis, I explore the role of attributes in interactive (human-in-the-loop) object category learning. I develop an active learning framework where the computer vision learning system actively solicits annotations from a pool of *both* object category labels and the objects’ shared attributes, depending on which will have the most impact for learning a joint object-attribute model. By simultaneously weighing requests in both label

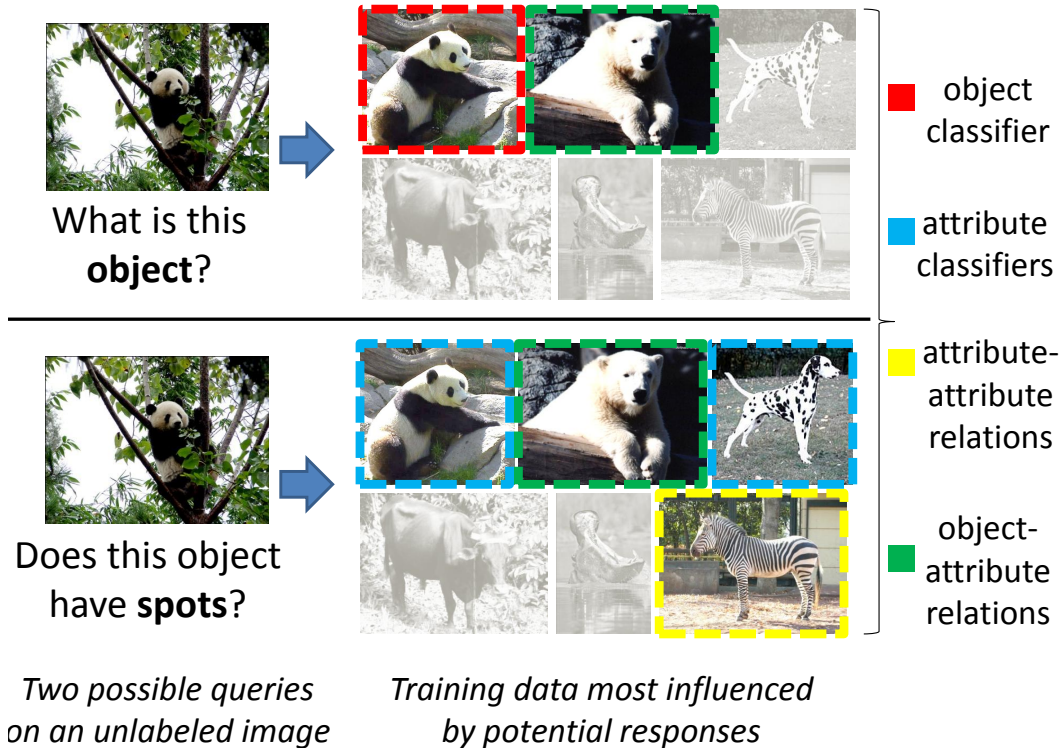


Figure 1.5: Object and attribute label requests affect an object category model’s understanding of each training image in distinct ways. This example illustrates how the different label requests about the image (left) will influence the different components of the learned models (right, color-coded by type of impact). For example, whereas getting the “panda” label may reduce uncertainty about that class and refine the model’s distinctions with other bear classes (top), getting the “spotted” label could have even greater influence, strengthening discriminability for the striped and spotted attributes alike.

spaces, the learner can more efficiently refine its object models because attributes are by definition shared across subsets of the object categories. Thus, knowledge of an attribute’s presence in an image can influence many object models. At the same time, attributes’ presence or absence in an image is often correlated, suggesting that many images do not require a full annotation of all attributes. See Figure 1.5. A novel aspect of the approach I propose is that it both weighs different annotation requests and also models dependencies between the target label space and a latent but human-describable label space. Only limited prior work explores either one or the other aspect [157, 143, 112]. The resulting object category models can be used when the user initiates a search via keywords such as “Show me images of cats” or “Show me images of sofas.”

**Roadmap** Figure 1.6 shows an overview of my work in this thesis. In the next chapter, I review related work on attributes, relevance feedback for image search, active learning, domain adaptation, collecting crowd labels and modeling users, and polysemy. In Chapter 3, I describe my work on providing relevance feedback using attributes (upper-right in Figure 1.6). In Chapter 4, I describe how such feedback can be actively requested by the retrieval system (lower-right in Figure 1.6). In Chapter 5, I propose techniques for learning user-specific attribute models (upper-center in Figure 1.6). In Chapter 6, I present a methodology for discovering attribute shades of meaning, and show how the corresponding schools of thought among users enable the learning of



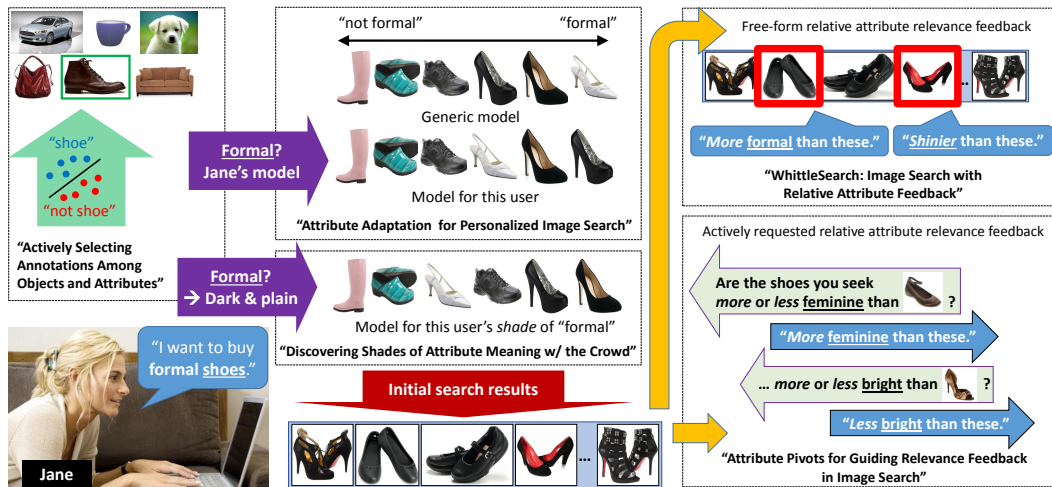


Figure 1.6: Overview of the work in this thesis. I show how to use relative attributes to provide relevance feedback (right), how to learn attribute models that align with each user’s perception (middle), and how to employ attributes to efficiently learn object categories (upper left), all of which are crucial for serving the user’s information need (lower left).

more robust attribute prediction models (center in Figure 1.6). In Chapter 7, I describe how attributes can be used to actively learn object category models, which can be employed when the user initializes a search with keywords (upper-left in Figure 1.6). In Chapter 8, I discuss some opportunities for future work that my thesis presents, and then conclude.

**Impact** Given the rate at which visual data is generated today, image retrieval systems cannot rely on pre-tagged images. Vision systems offer an approach to automatically analyze the content of images, and I demonstrate how we can best expose what the system has discovered about an image to the user, through a communication channel over visual attributes. Given the

abundance of data today, people frequently do not have time to simply browse results, so a retrieval strategy that pinpoints the most useful feedback a person can provide is particularly useful. Unlike existing work on visual attributes, my work pays close attention to the retrieval system’s users, by removing the assumption that “one model fits all” and modeling how a user applies attributes to communicate with the retrieval system. I focus on the modes in which an image retrieval system communicates with its users by integrating both the computer vision and the information retrieval perspective to image search, so the techniques I propose are a promising step in closing the semantic gap. Due to their computational efficiency, my methods are highly applicable in practice. I expect they will inspire a change in the interactions that image retrieval systems employ in the future.

## Chapter 2

### Related Work and Background

I first overview work on learning visual attributes, developing attribute vocabularies, and using attributes for object recognition and image retrieval. I then describe the state of the art of relevance feedback for interactive image search. Next, I discuss active selection approaches for object category learning and test-time prediction, as well as work on adapting object category models. Finally, I present some strategies for collecting annotations via crowdsourcing, and on handling polysemy in object names.

#### 2.1 Visual Attributes

Visual attributes are semantic properties of objects that serve as a middle ground between low-level features (e.g., color, texture) and high-level categories. Attributes describe the physical properties of objects such as materials, textures, shapes, and colors; the intended uses of objects or scenes; the habitat and behavior of animals; and other properties. Originally introduced in [45, 87, 40], attributes offer a semantic representation shared among objects. Attributes may be expressed categorically, as a property that is either present or absent [40, 87, 83, 142, 134, 110]. If so, they are called binary attributes.

Parikh and Grauman propose to model attributes relatively, as a comparative property that is present with a certain strength [107]. Sadovnik et al. study whether an attribute should be treated as a binary or relative property [127].

### 2.1.1 Learning Visual Attribute Models

Frequently attribute models are learned in fashion similar to object models for object category recognition. First, training data is collected, usually using non-experts annotators on crowdsourcing platforms such as Amazon Mechanical Turk (MTurk). Features that describe the training images are extracted using computer vision techniques. A computational model (such as a classifier) is then learned using the features and label annotations. This model can now predict the label for a novel image, using its automatically extracted features.

In order to obtain attribute annotations, Lampert et al. and Parikh and Grauman request labels from trusted annotators [87, 107]. Alternatively, labels can be collected redundantly from a number of untrained users on MTurk. Farhadi et al. perform a majority vote to arrive at the final attribute label [40], and in Chapters 3 and 4 I follow this practice too. Patterson and Hays use the count of users who voted for each possible label value to judge the strength of attribute presence [110], while Kumar et al. and Russakovsky and Li use this count to determine on which labels annotators unanimously agree and discard the rest [85, 124]. If attributes are assumed to be binary, annotators are usually presented with a page of images and asked to click on those which have the

attribute (e.g., as in Patterson and Hays [110]), or shown a single image and asked to check off the attributes that are present (e.g., as in Endres et al. [35]). In the case of relative attributes, annotators are asked to determine which of two images (or categories) has the attribute to a greater extent, effectively making a statement of the kind “A coast (or *this coast*) is more open than a forest (or *this forest*)” (as in Parikh and Grauman [107]).

Several datasets have been released which provide attribute labelings. Lampert et al. and Branson et al. provide datasets that capture the attributes of animals [87, 17], Kumar et al. provide a dataset of human face attributes [84], Berg et al. provide a dataset of shoes [11] which I augment with attributes (see Chapter 3), Patterson and Hays provide a scene attribute dataset [110], Matthews et al. provide a materials attribute dataset [99], and Farhadi et al. provide a dataset with attributes of general objects [40].

Attribute models are commonly learned as classifiers [87, 40, 83, 110, 134, 11, 170] from images that have and ones that do not have the attribute. For relative attributes, the comparative judgments regarding attribute strengths are used similarly to relevance judgments for learning document rankings. In other words, the ranking model tries to preserve the order of training images along the given attribute dimension, as well as encourage a large margin between examples of different ranks (attribute strengths). See [107, 78, 108, 64] for more details. Usually attributes are learned on the entire image level, but Duan et al. propose techniques for learning localized attributes [33].

Most researchers treat attributes as independent of one another, but

Wang and Mori and Siddiquie et al. also model the correlations between attributes [170, 142]. This allows a more efficient learning of attributes, since labels for one attribute can impact another attribute’s model, and the learned models are also more accurate, since one avoids making invalid independence assumptions. Jayaraman et al. show how to learn more accurate attribute models by decorrelating attributes [63].

### 2.1.2 Defining Attribute Vocabularies

When working with attributes, an important question is how to define the attribute *vocabulary*, that is, the set of attribute names to be learned. Most work using attributes assumes that images are fully labeled with all their attributes, either through a top-down labeling of the object classes (e.g., all bears are “furry” [87, 107]) or by individually providing attributes for each image [40, 35]. Some researchers elicit discriminative properties from annotators [110, 97], which they use to define an attribute vocabulary. To alleviate this burden of choosing a vocabulary manually or with great crowdsourcing effort, Ferrari and Zisserman study ways to learn attribute classifiers from noisy keyword search data [45], and Rohrbach et al. propose ways to mine attributes from script data [119]. Berg et al., Ordonez et al., and Rohrbach et al. suggest to automatically discover the attributes and objects’ semantic relatedness from web images and text sources [11, 105, 120]. For animal species, field guides are a natural source of attribute names [166, 17]. Since not all words will be visually detectable, the authors of [11, 7] show how to prune the vocabulary

automatically. Additionally, Parikh and Grauman show how to interactively develop a discriminative vocabulary of nameable attributes [106].

### 2.1.3 Discovering Non-Semantic Attributes

While the term “attribute” typically connotes a *semantic* property, as in the previous subsection, some researchers also use the term to refer to discovered *non-semantic* features [96, 116, 178, 138]. The idea is to identify “splits” or clusters in the low-level image descriptor space, often subject to constraints that deter redundancy and promote discriminativeness for object recognition. However, being bottom-up, there is no guarantee the splits will correspond to a nameable property. Hence, unlike the attribute shades discovery approach I present in Chapter 6, they are non-semantic and inapplicable to descriptive attribute tasks, like image search or zero-shot learning. One can attempt to assign names to discovered “attributes” after the fact [106, 33, 178], but the patterns that are even discoverable remain biased by the chosen low-level image feature space. In contrast, in this thesis I need attributes which people can use to communicate with a vision system, so these discovery methods are orthogonal to my work.

### 2.1.4 Subjectivity in Attribute Perception

While so far we have considered attributes to be objective properties, some researchers report significant disagreement over the attribute label among annotators [40, 35, 110]. Curran et al. [24] analyze how people perceive sub-

jective properties like “cool” and “cute”, but do not propose vision techniques to account for the subjectivity. As discussed above, typically discriminative classifiers or ranking algorithms are used to predict attributes. To my knowledge, all prior work assumes monolithic attribute predictors are sufficient, and none attempts to model user-specific perception, as I propose.

This includes prior methods that represent attributes relatively [107, 141]; although they permit looser comparative labels, they still assume a single underlying relative concept and learn a single “true” ordering of images. Relative attributes represent whether an image has a property “more” or “less”, and they construct a *universal* model for, e.g., “less brown” vs. “more brown”. The point is that people may agree best on *comparisons* or *strengths*, not binary labels. However, just like categorical attributes, relative attributes assume that there is some single, common interpretation of the property shared consistently by all viewers—namely, that a single ordering of images from least to most [attribute] is possible. They do not address the issue that one person may say “image X is *browner* than Y”, while another may say the opposite. The approaches presented in Chapters 5 and 6, on the other hand, are concerned with discovering *multiple* models for varying perceptions of brown, e.g., chocolate brown vs. goldish brown.

### 2.1.5 Attributes for Object Recognition

One of the common uses of attributes is as mid-level features for object recognition. Recent work explores several ways to use visual attributes



in object recognition [87, 40, 84, 170, 17, 141, 110]. Since attributes are often shared among object categories (e.g., “made of wood”, “plastic”, “has wheels”), they are amenable to a number of interesting tasks. Lampert et al. and Parikh and Grauman perform zero-shot learning to recognize unseen objects from category descriptions [87, 107]. Akata et al. propose to perform zero-shot learning by embedding class labels in the space of attribute labels [2]. Farhadi et al. describe unfamiliar objects or novel instances [40]. Branson et al. categorize images of birds with a 20-questions game [17]. Saleh et al. detect anomalous objects with the aid of attributes [133]. Wah and Belongie detect that a previously unfamiliar object category is presented to the system [163]. Sadvnik et al. suggest how one can uniquely identify individuals through an attribute description [126]. Donahue and Grauman use attributes as an effective medium through which to explain to a visual system what identifies an image as belonging to a given category [30]. Kulkarni et al. perform on-the-fly classification via attribute-based transfer [82].

By integrating the learning process for both objects and attributes, Wang and Forsyth use weak supervision more effectively [164], and Kumar et al. and Wang et al. improve object recognition accuracy [84, 170].

### **2.1.6 Attributes for Image Search**

While attributes are most commonly used for recognition, they are also sometimes used to aid image retrieval. Attributes were only recently introduced in the computer vision community, but in the multimedia community,

*semantic concepts* have long been used as an intermediate representation for image retrieval [144, 115, 101, 167]. Concepts are tags that can be assigned to images manually or automatically, and they serve to index an image so it can be retrieved when its concept representation matches some query. Unlike attributes in computer vision, concepts simply denote the presence or absence of a certain feature of an *image*; concepts are not meant to be *properties of objects*, nor to be shared by objects, as concepts are sometimes objects themselves. Examples of concepts are “outdoors”, “face”, “people”, “landscape”, and “speech” [101].

Similarly to semantic concepts, visual attributes can be used as the feature space in which retrieval is performed [31, 167]. Attributes can also be used more directly to issue multi-attribute keyword queries to retrieval systems, either of the form “find images of *smiling Asian men*” [83, 142, 117] or of the form “find images of men *smiling more than/similarly to this one*” [134, 78]. Some recent work studies techniques for optimizing retrieval using multi-attribute queries. Siddiquie et al. model the dependencies between attributes [142], Scheirer et al. calibrate SVM decision scores per attribute [134], and Rastegari et al. selectively merge some of the attributes in a query [117]. Vaquero et al. and Reid and Nixon use attributes as an effective way to retrieve subjects in video surveillance data [156, 118].

While it is known that attributes can provide a richer representation for image retrieval than raw low-level image features, no previous work considers attributes as a handle for user feedback, as I propose.

Existing work assumes that search users mean the same as other users when they employ a certain attribute term. I show this assumption does not hold and propose how to learn *user-specific* attributes in Chapter 5.

## 2.2 Relevance Feedback in Image Search

In a major part of this thesis, I study how to best use attributes in order to perform rich relevance feedback. Relevance feedback has long been used to improve interactive image search [86, 123, 151, 23, 182, 43, 47]. The idea is to tailor the system’s ranking function to the current user. This injects subjectivity into the model, implicitly guiding the search engine to pay attention to certain low-level visual cues more than others. An early approach to relevance feedback involved allowing users to adjust the weight that different feature spaces contribute to the overall similarity between the query and database images [46, 94, 61]. A more recent family of approaches allows users to give (usually iterative) feedback on the relevance of selected exemplar images. Rui et al. [123] ask the user to provide relevance scores for a set of images, and the retrieval system learns how much each feature type should contribute to the image matching and retrieval based on this feedback. Ferecatu and Geman [43] ask the user to mark which image in a display set is closest to his target, and the system updates its relevance estimate for each image based on this information. Cox et al. [23] ask the user to provide relative similarity judgments: the displayed images which are marked by the user are more similar to the target than those left unmarked. Fogarty et al. allow users to train concept

models for concepts they would like to retrieve by providing examples [47]. Zhou et al. provide a comprehensive survey of relevance feedback methods in [182].

Like existing interactive methods, the approach I propose in Chapter 3 aims to elicit a specific user’s target visual concept. However, while prior work restricts input to the form “A is relevant, B is not” or “C is more relevant than D”, my approach allows users to comment precisely on what is missing from the current set of results. I show that this richer form of feedback can make refinement more effective.

### **2.2.1 Active Selection of Solicited Feedback**

In practice, the images displayed to the user for feedback are usually those ranked best by the system’s current relevance model. However, if a user is cooperative, it can be more valuable to present a mix of probable relevant and irrelevant examples for feedback. If feedback is binary, with the user labeling examples as relevant (positive) or irrelevant (negative), the selection can naturally be cast as an active learning problem: the best examples to show are those that will be most informative to the relevance classifier. For example, Tong et al. use the decision boundary of a “relevant” / “not relevant” SVM to determine on which examples to request feedback [152]. Li et al. employ query-by-committee to select the instances on which to request feedback [89]. The review by Zhou et al. discusses other existing approaches for optimally selecting the examples for feedback [182].

The majority of the active selection methods discussed above look for images that the retrieval system is most uncertain about, and not ones that are likely to impact the system’s uncertainty over the relevance of *all* dataset images. Prior efforts to display the exemplar set that minimizes uncertainty over the entire database were forced to resort to sampling or clustering heuristics due to the combinatorial optimization problem inherent when categorical feedback is assumed. Cox et al. use sampling to find images whose labeling is likely to terminate the search the fastest [23]. Ferecatu and Geman use heuristics to find the most balanced display scheme which is likely to have the highest entropy [43]. Sampling is a necessary procedure for other active learning work, e.g., work by Roy and McCallum [122] for document classification. In contrast, I show that eliciting *comparative* feedback on ordinal visual attributes naturally leads to an efficient sequential selection strategy, where each comparison is guaranteed to decrease the predicted relevance of half of the unexplored database images.

### **2.2.2 Personalization of Search**

In information retrieval, researchers have studied methods to provide each user with personalized search results, by learning what each user perceives as relevant in the context of their information need, as discussed by Pasi [109]. Teevan et al. and Joachims treat personalization as a form of relevance feedback which can be mined explicitly but also implicitly, by creating user profiles or mining clickthrough data [149, 64]. Whereas personalization

generally entails learning a user-specific relevance function from scratch—there is no “universal” prior on relevance—in Chapter 5 I propose a method to leverage a generic model for the attribute as a starting point, and efficiently adapt it towards the user’s preferences. As I demonstrate, doing so saves user time.

## 2.3 Active Selection for Learning Object Categories

Both relevance feedback and object category learning can be improved through intelligent selection of the data on which feedback or labels are requested. Researchers have developed techniques in which a system that is learning object category models makes active requests to the human teacher to label certain data. There is work in *active learning*, which denotes making active requests for the data that the system uses for *training*. In *active testing*, the system selects the questions that it asks during the *classification* of a single test instance. I overview each in turn next.

### 2.3.1 Active Learning for Classification

Active learning tackles the expense of the human labeling work in providing supervision to a computer vision system. It typically reduces the labeling effort by selecting the most uncertain exemplar to get labeled with its object category name(s) [112, 180, 66, 62]. Some work further shows how to actively integrate annotations of different *levels*, i.e., by alternately requesting segmented regions or asking about the contextual relationships between objects in an image [157, 158, 143].

Attributes present an annotation level which is distinct from object category labels. In the realm of natural language processing, researchers develop ways to actively ask people which words may be relevant for a document classification task [114, 32]; words could be seen as a loose analogue for attributes, though I do not consider requests about visual attribute relevance. A few methods investigate training classifiers with actively selected attribute labels. Parkash and Parikh perform active learning where the teacher provides guidance to the learning system in the form of relative attribute statements, e.g., “This is not a giraffe because its neck is not long enough” [108]. Once the teacher explains an incorrect prediction with an attribute, images with the right attribute strength (according to the explanation) are added as negative training data for the given object class, but their attribute values are not used anymore. Mensink et al. perform attribute-based classification (similarly to Lampert et al. [87]) and employ attributes in an interactive labeling scenario [100], but they do not allow the active learning to choose from a pool of object *and* attribute labels, as I propose in Chapter 7.

Active visual learning methods generally do not account for the dependencies between labels on the same image. An exception is the scene classification method of Qi et al. [112], which learns with multi-label images and requests the most informative image-label pair. However, its selection strategy considers only the local effects of a candidate label request, by measuring the uncertainty and label correlations for each individual image in isolation. In contrast, I propose a selection method that evaluates the influence of the

candidate label if propagated to all current models, which is critical to achieve the goal of exploiting shared latent attributes to reduce annotation effort.

### 2.3.2 Active Testing and Twenty Questions

While the above work tackles active *learning*, active *testing* methods deduce the object label for a single novel image by asking a person to provide information about it. Geman and Jedynek choose a series of useful “tests” (e.g., features to extract) to classify an image [49]. Branson et al. actively choose the attribute questions (“does the bird have a yellow beak?”) which they ask a human in the loop in order to classify a single image [17]. In the latter case where a *person* answers the tests, attributes are well-suited as intermediate labels that will lead to the right category label, since as Branson et al. argue, for fine-grained recognition tasks like bird species identification, it is easier for a person to label an attribute than to classify the image [17].

In Chapter 4, I describe work on active selection for image feedback, which shares the spirit of rapidly reducing uncertainty through a sequence of useful questions. However, my aim is distinct. My method selects queries to efficiently find a target in a database of images, as opposed to classify a single image. Moreover, my approach solicits visual *comparisons*—key to eliminating irrelevant content in search—whereas prior work solicits traditional image labels.



## 2.4 Learning Visual Categories with Domain Adaptation

Active selection helps learn models more efficiently with less data, and transfer learning shares the spirit of minimizing the need for annotated data for every category. In Chapter 5, I show how to adapt a generic attribute model to learn a user-specific one. Somewhat analogously, transfer learning work in object recognition leverages previously learned object categories when training a new category for which few labeled images are available (e.g., [146, 5, 41, 113]). Also related are domain adaptation methods (e.g., [130, 55, 54]), which account for the feature distribution mismatch between a source domain (in which objects are learned) and a target domain (in which the objects must be recognized). For example, as Saenko et al. argue, this allows a classifier trained on web images to work well on images taken by a robot [130].

Conceptually my adaptation goal is closer in spirit to speaker-dependent speech recognition. Speaker adaptation methods have long been used in the speech community to adapt parameters of a speaker-independent model; for example, Gauvain et al. account for an individual’s idiosyncrasies (voice, accent, etc.) [48]. Also related is collaborative spam filtering, where a personalized spam classifier can make use of a global non-personal one, e.g., work by Attenberg et al. [4]. I explore existing adaptation formulations for SVMs by Yang et al. and Geng et al. [177, 50]. Using them to learn user-specific attribute models is novel, and I introduce methods to infer user-specific training labels.

## 2.5 Aggregating Crowd Labels

Researchers frequently need to obtain labels on possibly large amounts of new data. Learning from multiple noisy labelers is increasingly important for training data-hungry vision systems, particularly given the inexpensive annotation which researchers can request on Mechanical Turk from untrained labelers. Typically, an image labeling task is “crowdsourced” by submitting it to workers on Mechanical Turk, then aggregating their labels through a majority vote.

Crowd input has been aggregated in novel ways for image clustering [53], image similarity [148], and object labeling [171]. In [171], modeling annotators’ competence and bias makes it possible to discover their “schools of thought”, and subsequently undo their biases to produce more reliable ground truth. Tian and Zhu also discover schools of thought that exist among workers [150], and Gomes et al. present a clustering method that acknowledges that workers can have their own notion of categories [53]. Kajino et al. first learn “personal classifiers” for each worker, and then integrate these to achieve one common model [67]. Ertekin et al. modify majority voting by accounting for the reliability of each worker [36]. Yan et al. integrate the skills of different users into an active learning formulation, so they choose both the instances to be labeled and the workers to label them [176].

While the above work models each worker’s school of thought, most authors still aim for consensus and recover a single model that captures the opinion of the crowd [171, 36, 53, 150, 176, 67]. In contrast, the adaptive at-

tribute models I propose in Chapter 5 recover an individual user’s subjective attribute model from their annotations, by properly adapting a generic model over all previously seen users. Furthermore, in my work the task is not recognition but search, which is inherently more person-dependent since relevance is defined only for a specific user. Note that I focus on individual users who are trying to perform a search, as opposed to “crowds” on MTurk.

Also in the realm of crowdsourcing, matrix factorization is often used to complete user-label matrices and solve collaborative filtering problems (e.g., the Netflix challenge) by exploiting commonalities among users [132, 175]. Rather than impute missing labels, I propose to use the latent factors themselves to represent the interplay between language, human perception, and image examples. I show how to use the recovered schools of thought to build content-based attribute models.

## 2.6 Polysemy of Words in Images

A polysemous word has multiple “senses” or meanings, which can be found in a dictionary. Similarly, in Chapter 6, I aim to discover the (likely unlisted in a dictionary) “senses” of an attribute term. Some existing work bridging text and visual analysis aims to cluster web images according to distinct senses [8, 92, 128, 12]. However, the focus is on nouns/object categories, not descriptive properties. Typically the visual differences (or surrounding text context) are fairly stark (e.g., a river *bank* or financial *bank*). In contrast, the attribute shades I study in Chapter 6 are often subtle differences in inter-

pretation. Furthermore, unlike a truly polysemous word, for which one can enumerate the multiple dictionary definitions, attribute shades are often more difficult to definitively express in language. I show how to automatically infer them from trends in crowd labels.

## Chapter 3

# WhittleSearch: Image Search with Relative Attribute Feedback

In this chapter, I will demonstrate how attributes can provide a rich form of relevance feedback for interactive image search.<sup>1</sup> By being more fine-grained than categories and also extending across category boundary lines, attributes enable a very precise description of the user’s information need.

Using the relevance feedback paradigm, the approach I propose allows a user to iteratively refine the search using feedback on attributes, as discussed in Chapter 1. The user initializes the search with some keywords—either the name of the general class of interest (“shoes”) or some multi-attribute query (“black high-heeled shoes”)—and my system’s job is to help refine from there. If no such initialization is possible, the search simply begins with a random set of images for feedback. The top-ranked images are then displayed to the user, and the feedback-refinement loop begins.

---

<sup>1</sup>This work was published in the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2012 with the title “WhittleSearch: Image Search with Relative Attribute Feedback” and authors Adriana Kovashka, Devi Parikh, and Kristen Grauman. I wrote the majority of the code and conducted most experiments and data collection, while Devi Parikh wrote some code, collected some data, and conducted some experiments in Section 3.5.6, and all authors contributed to developing the algorithm, devising the experiments, and writing the paper.

Offline, the system firsts learn a set of ranking functions, each of which predicts the relative strength of a nameable attribute in an image (e.g., the degree of “shininess”, “furriness”, etc.). At query time, the system presents an initial set of reference images or a single image paired with an attribute, and the user provides relative attribute feedback. Using the resulting constraints in the multi-dimensional attribute space, the system updates its relevance function, re-ranks the pool of images, and displays to the user the image(s) which are most relevant. (Later, in Chapter 4, I extend this idea to also display to the user the image(s) most likely to reduce the system’s uncertainty.) This procedure iterates using the accumulated constraints until the top ranked images are acceptably close to the user’s target.

Throughout, let  $\mathcal{D} = \{I_1, \dots, I_N\}$  refer to the pool of  $N$  database images that are ranked by the system using its current scoring function  $S_t : I \rightarrow \mathbb{R}$ , where  $t$  denotes the iteration of refinement. The scoring function is trained using all accumulated feedback from iterations  $1, \dots, t - 1$ , and it supplies an ordering (possibly partial) on the images in  $\mathcal{D}$ .  $S_t(I_i)$  captures the likelihood that image  $I_i$  is relevant to the user’s information need, given the feedback received before iteration  $t$ .

At each iteration  $t$ , the top  $K < N$  ranked images  $\mathcal{T}_t = \{I_{t1}, \dots, I_{tK}\} \subseteq \mathcal{D}$  are displayed to the user for further feedback, where  $S_t(I_{t1}) \geq S_t(I_{t2}) \geq \dots \geq S_t(I_{tK})$ . A user then gives feedback of his choosing on any or all of the  $K$  refined results in  $\mathcal{T}_t$ . We refer to  $\mathcal{T}_t$  interchangeably as the *reference set* or *top-ranked set*.

In the following, I first describe a traditional binary relevance feedback model (Section 3.1), since it will serve as a strong baseline to which to compare my approach. Then I introduce the proposed new mode of relative attribute feedback (Section 3.3) after describing how relative attribute models are learned (Section 3.2). Finally, I extend the idea to accommodate both forms of input in a hybrid approach (Section 3.4).

### 3.1 Background: Binary Relevance Feedback

In a binary relevance feedback model, the user identifies a set of relevant images  $\mathcal{R}$  and a set of irrelevant images  $\bar{\mathcal{R}}$  among the current reference set  $\mathcal{T}_t$ . In this case, the scoring function  $S_t^b$  is a classifier (or some other statistical model), and the binary feedback essentially supplies additional positive and negative training examples to enhance that classifier. That is, the scoring function  $S_{t+1}^b$  is trained with the data that trained  $S_t^b$  *plus* the images in  $\mathcal{R}$  labeled as positive instances and the images in  $\bar{\mathcal{R}}$  labeled as negative instances.

As a baseline, we use a binary feedback approach that is intended to represent traditional approaches such as [23, 43, 123, 151, 152]. While a variety of classifiers have been explored in previous systems, we employ a support vector machine (SVM) classifier for the binary feedback model due to its strong performance in practice. Thus, the scoring function for binary feedback is  $S^b(I_j) = \mathbf{w}_b \mathbf{x}_j^T + b$ , where  $\mathbf{w}_b$ ,  $b$  are the SVM parameters and  $\mathbf{x}_j$  denotes the visual features extracted from image  $I_j$ , to be defined below.

By definition, the classification approach to relevance feedback requires

instances with both positive and negative feedback. In practice, it may be valuable to gather more or less of either type, depending on the data. See [168, 69, 34] for an examination of the different contribution of positive and negative feedback.

### 3.2 Learning to Predict Relative Attributes

Suppose we have a vocabulary of  $M$  attributes  $A_1, \dots, A_M$ , which may be generic or domain-specific for the image search problem of interest. For example, a domain-specific vocabulary for shoe shopping could contain attributes such as “shininess”, “heel height”, “colorfulness”, etc., whereas for scene descriptions it could contain attributes like “openness”, “naturalness”, “depth”. While we assume this vocabulary is given, recent work suggests it may also be discoverable automatically (see Section 2.1.2).

To leverage the proposed relative attribute feedback, my method requires attribute strengths on all images<sup>2</sup> and a means to aggregate cumulative constraints on individual attributes, as I describe in the following.

Typically semantic visual attributes are learned as categories: a given image either exhibits the concept or it does not, and so a classification approach to predict attribute presence is sufficient [115, 101, 179, 87, 40, 83, 170, 31]. In contrast, to express feedback in the form sketched above, we require *relative*

---

<sup>2</sup>It would be too expensive to manually annotate all images with their attribute strength, so we will learn to extrapolate a small set of annotations to a prediction function over all database images.



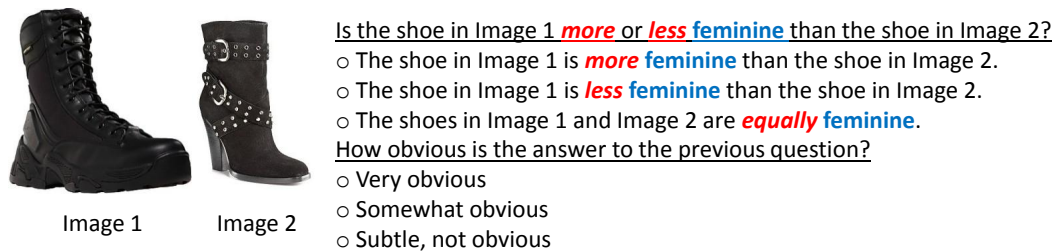


Figure 3.1: Interface for image-level relative attribute annotations.

attribute models [107] that can predict the degree to which an attribute is present. Therefore, we first learn a *ranking function* for each attribute in the given vocabulary. One might informally treat classifier outputs as “strengths”, yet doing so is inconsistent with a training procedure that actually targets hard categorical labels. Results in [107] confirm that simply treating a binary classifier output value as the strength of presence is inferior in practice compared to training ranking functions.

For each attribute  $A_m$ , we obtain supervision on a set of image pairs  $(i, j)$  in the training set  $\mathcal{J}$ . We ask human annotators to judge whether that attribute has stronger presence in image  $i$  or  $j$ , or if it is equally strong in both. Such judgments can be subtle, so on each pair we collect five redundant responses from multiple annotators on Amazon Mechanical Turk, in order to elicit the most common perception of the attribute and reduce the impact of noisy responses. See Figure 3.1. To ensure annotation quality and distill reliable relative constraints for training, we use only those for which most labelers agree on one of the three possible responses (“more”, “less”, or “equally”). This yields a set of ordered image pairs  $O_m = \{(i, j)\}$  and a set of un-ordered

pairs  $E_m = \{(i, j)\}$  such that  $(i, j) \in O_m \implies i \succ j$ , i.e., image  $i$  has stronger presence of attribute  $A_m$  than  $j$ , and  $(i, j) \in E_m \implies i \sim j$ , i.e.,  $i$  and  $j$  have equivalent strengths of  $A_m$ .

I stress the design for constraint collection: rather than ask annotators to give an *absolute* score reflecting how much the attribute  $m$  is present, we instead ask them to make *comparative* judgments on two exemplars at a time. This is both more natural for an individual annotator, and also permits seamless integration of the supervision from many annotators, each of whom may have a different internal “calibration” for the attribute strengths.

Next, to learn an attribute’s ranking function, we employ the large-margin formulation of Joachims [64], which was originally shown for ranking web pages based on clickthrough data, and recently used for relative attribute learning [107]. Suppose each image  $I_i$  is represented in  $\mathbb{R}^d$  by a feature vector  $\mathbf{x}_i$  (we use color and GIST). We aim to learn  $M$  ranking functions, one per attribute:

$$a_m(\mathbf{x}) = \mathbf{w}_m^T \mathbf{x}_i, \tag{3.1}$$

for  $m = 1, \dots, M$ , such that the maximum number of the following constraints is satisfied:

$$\forall (i, j) \in O_m : \mathbf{w}_m^T \mathbf{x}_i > \mathbf{w}_m^T \mathbf{x}_j. \tag{3.2}$$

Joachims’ algorithm approximates this NP hard problem by introducing (1) a regularization term that prefers a wide margin between the ranks

assigned to the closest pair of training instances, and (2) slack variables  $\xi_{ij}$  on the constraints, yielding the following objective [64]:

$$\begin{aligned} \text{minimize} \quad & \left( \frac{1}{2} \|\mathbf{w}_m^T\|_2^2 + C \sum \xi_{ij} \right) & (3.3) \\ \text{s.t.} \quad & \mathbf{w}_m^T \mathbf{x}_i \geq \mathbf{w}_m^T \mathbf{x}_j + 1 - \xi_{ij}; \quad \forall (i, j) \in O_m \\ & \xi_{ij} \geq 0, \end{aligned}$$

where  $C$  is a constant penalty. The objective is reminiscent of standard SVM training (and is solvable using similar decomposition algorithms), except the linear constraints enforce relative orderings rather than labels. The method is kernelizable. We use Joachims’ SVMRank code [65].<sup>3</sup>

Having trained  $M$  such functions, we are then equipped to predict the extent to which each attribute is present in any novel image, by applying the learned functions  $a_1, \dots, a_M$  to its image descriptor  $\mathbf{x}$ . Note that this training is a one-time process done before any search query or feedback is issued, and the data  $\mathcal{J}$  used for training attribute rankers is not to be confused with our database pool  $\mathcal{D}$ .

These predicted attribute values  $a_m(I_i)$  are what we can observe for image  $I_i$ . They are a function of (but distinct from) the “true” latent attribute strengths  $A_m(I_i)$ . I will refer to both below. Using standard features and kernels, I find that 75% of held-out ground truth comparisons are preserved

---

<sup>3</sup>Note that one can also use the equality constraints in  $E_m$  for training these ranking functions, as in [107]. In my approach, I use these constraints to compute parameters for scoring relevance, in Section 4.2.

by attribute predictors trained with  $\sim 200$  pairs. Thus, they are quite reliable; more elaborate features [83] or learning algorithms [90] would likely improve them even further, but this is outside the scope of this thesis.

Whereas Parikh and Grauman [107] propose generating supervision for relative attributes from top-down category comparisons (“person X is (always) more smiley than person Y”), my approach extends the learning process to incorporate *image-level* relative comparisons (“image A exhibits more smiling than image B”). While training from category-level comparisons is clearly more expedient, I find that image-level supervision is important in order to reliably capture those attributes that do not closely follow category boundaries. The “smiling” attribute is a good example of this contrast, since a given person (the category) need not be smiling to an equal degree in each of his/her photos. In fact, my user studies on MTurk show that category-level relationships violate 23% of the image-level relationships specified by human subjects for the “smiling” attribute. In Section 3.5.6, I detail related studies analyzing the benefits of instance-level comparisons.

### 3.3 Relative Attribute Feedback

With the ranking functions learned above, we can now map any image from  $\mathcal{D}$  into an  $M$ -dimensional space, where each dimension corresponds to the relative rank prediction for one attribute. It is in this feature space I propose to handle query refinement from a user’s feedback.

A user of the system has a mental model of the target visual content he

seeks. To refine the current search results, he surveys the  $K$  top-ranked images in  $\mathcal{T}_t$ , and uses some of them as reference images with which to better express his envisioned optimal result. These constraints are of the form “What I want is more/less/similarly  $m$  than image  $I_{t_f}$ ”, where  $m$  is an attribute name, and  $I_{t_f}$  is an image in  $\mathcal{T}_t$  (the subscript  $t_f$  denotes it is a *reference* image at iteration  $t$ ). These relative constraints are given for some combination of image(s) and attribute(s) of the user’s choosing.

The conjunction of all such user feedback statements gives us a set of constraints for updating the scoring function. For all statements of the form “I want images exhibiting more of attribute  $m$  than reference image  $I_{t_f}$ ”, our updated attribute-based scoring function  $S_{t+1}^a$  should satisfy:

$$\begin{aligned} S_{t+1}^a(I_i) &> S_{t+1}^a(I_j), \quad \forall I_i, I_j \in \mathcal{D} \\ \text{s.t. } a_m(I_i) &> a_m(I_{t_f}), \quad a_m(I_j) \leq a_m(I_{t_f}), \end{aligned} \tag{3.4}$$

where as before  $\mathbf{x}_i$  denotes the image descriptor for image  $I_i$  used to predict its relative attributes, and images  $I_i$  and  $I_j$  are otherwise equivalently ranked by the scoring function  $S_t^a$  at the previous iteration. This simply reflects that images having more of the desired property  $m$  than the displayed reference image are better than those that do not. I stress that the relative attribute values on all images are *predicted* using the learned function  $a_m$  (as opposed to having ground truth on the attribute strengths in each image).

Similarly, for all statements of the form “I want images exhibiting less of attribute  $m$  than  $I_{t_f}$ ”, our updated scoring function should satisfy:

$$S_{t+1}^a(I_i) > S_{t+1}^a(I_j), \quad \forall I_i, I_j \in \mathcal{D} \tag{3.5}$$

$$s.t. \quad a_m(I_i) < a_m(I_{t_f}), \quad a_m(I_j) \geq a_m(I_{t_f})$$

For all statements of the form, “I want images that are similar in terms of attribute  $m$  to  $I_{t_f}$ ”, the constraints are:

$$S_{t+1}^a(I_i) > S_{t+1}^a(I_j), \quad \forall I_i, I_j \in \mathcal{D} \quad (3.6)$$

$$s.t. \quad (a_m(I_{t_f}) - \epsilon) \leq a_m(I_i) \leq (a_m(I_{t_f}) + \epsilon),$$

$$a_m(I_j) < a_m(I_{t_f}) - \epsilon \quad \text{or} \quad a_m(I_j) > a_m(I_{t_f}) + \epsilon,$$

where  $\epsilon$  is a constant specifying the distance in relative attribute space at which instances are considered dissimilar. Note that these similarity constraints differ from binary feedback, in that they single out an individual attribute. The implementation in this chapter focuses on the two relative forms of feedback (“more”, “less”).

Each of the above carves out a relevant region of the  $M$ -dimensional attribute feature space, *whittling away* images not meeting the user’s requirements. We combine all such constraints to adapt the scoring function from  $S_t^a$  to  $S_{t+1}^a$ . Let  $\mathcal{F} = \{(I_{t_f}, m, r)\}$  denote the set of all accumulated comparative constraints thus far. Each item in  $\mathcal{F}$  consists of a reference image  $I_{t_f}$  for attribute  $m$ , and a user response  $r \in \{\text{“more”}, \text{“less”}, \text{“equally”}\}$ . The number of such feedback constraints is  $|\mathcal{F}|$ , and we take the intersection of all  $|\mathcal{F}|$  feedback constraints thus far to identify the set of top ranked images, for which  $S_{t+1}^a(I_i) = |\mathcal{F}|$ . Those satisfying all but one constraint receive score  $|\mathcal{F}| - 1$ , and so on, until images satisfying no constraints receive the score 0. See Figure



Figure 3.2: A toy example illustrating the intersection of relative constraints with  $M = 2$  attributes. The images are plotted on the axes for both attributes. The space of images that satisfy each constraint are marked in a different color. The region satisfying all constraints is marked with a black dashed line. In this case, there is only one image in it (outlined in black). Best viewed in color.

3.2. The final output at iteration  $T$  of our search system will be a sorting of the database images in  $\mathcal{D}$  according to their likelihood of being relevant.

Formally, and to maintain consistency with Chapter 4, we can describe the relevance function as follows. Let  $G_{k,i} \in \{0, 1\}$  be a binary random variable representing whether image  $I_i$  satisfies the  $k$ -th feedback constraint. For example, if the user’s  $k$ -th comparison on attribute  $m$  yields response  $r = \text{“more”}$ , then  $G_{k,i} = 1$  if the database image  $I_i$  has attribute  $m$  more than the corresponding reference image  $I_{t_f}$ . The estimate of relevance is thus proportional to the probability that any  $|\mathcal{F}|$  feedback comparisons are satisfied:

$$S_T^a(I_i) = \sum_{k=1}^{|\mathcal{F}|} P(G_{k,i} = 1 | I_i, \mathcal{F}_k). \quad (3.7)$$

Using Iverson bracket notation, we set the probability that an individual constraint is satisfied given that the user’s response was  $r$  for reference  $I_{t_f}$  to:

$$P(G_{k,i} = 1 | I_i, \mathcal{F}_k) = \begin{cases} [a_m(I_i) > a_m(I_{t_f})] & \text{if } r = \text{“more”} \\ [a_m(I_i) < a_m(I_{t_f})] & \text{if } r = \text{“less”}. \end{cases} \quad (3.8)$$

One could also learn a ranking function for  $S_{t+1}^a$  using these constraints within the large-margin objective above; however, for the sake of determining the *ordering* on the data—as is needed to refine the top ranked results—its behavior would be equivalent. Thus we take this purely set-logic approach, as it is less costly.

I stress that the proposed form of relative attribute feedback refines the search in ways that a straightforward multi-attribute [85, 142, 134] query cannot. That is, if a user were to simply state the attribute labels of interest (“show me black shoes that are shiny and high-heeled”), one can easily retrieve the images whose attribute predictions meet those criteria. However, since the user’s description is in absolute terms, it *cannot change based on the retrieved images*. In contrast, with access to relative attributes as a mode of communication, for every new set of reference images returned by the system, the user can further refine his description.

Similarly to multi-attribute queries, faceted browsing or search—where the retrieval system organizes documents or products according to several



properties (facets) and allows the user to query with different combinations of the facets [153, 72, 81, 10, 155]—is also a form of keyword search with fixed values for the attribute properties. However, while this form of search may be appropriate for content where properties can be objectively measured and quantized, it does not suffice for search over items where a user’s preferences and goals may be very specific and possibly subjective. Further, it is not easy to quantize attributes in order to show multiple-valued facets, e.g., to determine what lies within a range of 0.2 to 0.4 of “pointiness”, as that varies across datasets and contexts.

Once a cycle of feedback and refinement is completed, the method repeats the loop, accepting any additional feedback from the user on the newly top-ranked images. In practice, the system can either iterate until the user’s target image is found, or else until his “budget” of interaction effort is expended.

### **3.4 Hybrid Feedback Approach**

So far, we have considered relative attribute feedback in isolation and discussed its advantages over traditional binary relevance feedback. However, binary relevance feedback and relative attribute feedback can have complementary strengths: when reference images are nearly on target (or completely wrong in all aspects), the user may be best served by providing a simple binary relevance label. Meanwhile, when a reference image is lacking only in certain describable properties, he may be better served by the relative attribute feed-

back. Thus, it is natural to combine the two modalities, allowing a mix of feedback types at any iteration.

To this end, one can consider a learned hybrid scoring function. The basic idea is to learn a ranking function  $S_{t+1}^h$  that unifies both forms of constraints. Recall that  $\mathcal{R}$  and  $\bar{\mathcal{R}}$  denote the sets of reference images for which the user has given positive and negative binary feedback, respectively. Let  $\mathcal{V}_k \subset \mathcal{D}$  denote the subset of images satisfying  $k$  of the relative attribute feedback constraints, for  $k = 0, \dots, F$ . We define a set of ordered image pairs

$$O_s = \{\{\mathcal{R} \times \bar{\mathcal{R}}\} \cup \{\mathcal{V}_F \times \mathcal{V}_{F-1}\} \cup \dots \cup \{\mathcal{V}_1 \times \mathcal{V}_0\}\}, \quad (3.9)$$

where  $\times$  denotes the Cartesian product. This set  $O_s$  reflects all the desired ranking preferences—that relevant images be ranked higher than irrelevant ones, and that images satisfying more relative attribute preferences be ranked higher than those satisfying fewer.

Note that the subscript  $s$  in  $O_s$  distinguishes the set from those indexed by  $m$  above, which were used to train relative attribute ranking functions in Section 3.2.

Using training constraints  $O_s$  we can learn a function that predicts *relative image relevance* for the current user with the large-margin objective in Equation 3.3. The result is a parameter vector  $\mathbf{w}_s$  that serves as the hybrid scoring function  $S_{t+1}^h$ . Since there are many more pairs in  $O_s$  that come from relative attribute feedback than from binary relevance feedback, we set the

penalty on the binary feedback pairs to be inversely proportional to the fraction of such pairs in the set  $O_s$ .

To recap the approach section, we now have three forms of scoring functions to be used for refining search results: traditional binary feedback ( $S^b$ ), relative attribute feedback ( $S^a$ ), and a hybrid that unifies the two ( $S^h$ ).

## 3.5 Experimental Validation

I analyze how the proposed relative attribute feedback can enhance image search compared to classic binary feedback, and study what factors influence their behavior.

### 3.5.1 Experimental Design

**Datasets** I use three datasets: the **Shoes** from the Attribute Discovery Dataset [11], the Public Figures dataset of human faces [84] (**PubFig**), and the Outdoor Scene Recognition dataset of natural scenes [103] (**OSR**). These datasets validate my approach in diverse domains of interest: finding products, people, and scenes. The Shoes data contains 14,658 shoe images from `like.com`. I augment the data with relative attributes (see Table 3.1) which cover many useful properties of shoes. I collect labels for the shoe attributes by computing a majority vote across labels from 5 workers per image. For PubFig I use the subset from [107], which contains 772 images from 8 people and 11 attributes. OSR consists of 2,688 images from 8 categories and 6 attributes [107].

Shoes	OSR	Pubfig
pointy at the front	natural	masculine-looking
open	open	white
bright in color	perspective	young
covered with ornaments	large-objects	smiling
shiny	diagonal-plane	chubby
high at the heel	close-depth	visible forehead
long on the leg		bushy eyebrows
formal		narrow eyes
sporty		pointy nose
feminine		big lips
		round face

Table 3.1: A list of the attribute names for the Shoes, OSR, and PubFig datasets.

Figure 3.3 shows some example images from each dataset, and Table 3.1 lists the attribute names per dataset. For OSR and PubFig, I use whichever attributes the datasets included. For Shoes, I define my own attribute vocabulary such that it is compact but provides good coverage of the properties of shoes one might want to describe. As we will see below, I obtain strong results on all three datasets. This shows that my approach is not very sensitive to the choice of attribute vocabulary, as long as the attributes used can be both learned with reasonable accuracy by the machine, and understood well by the user.

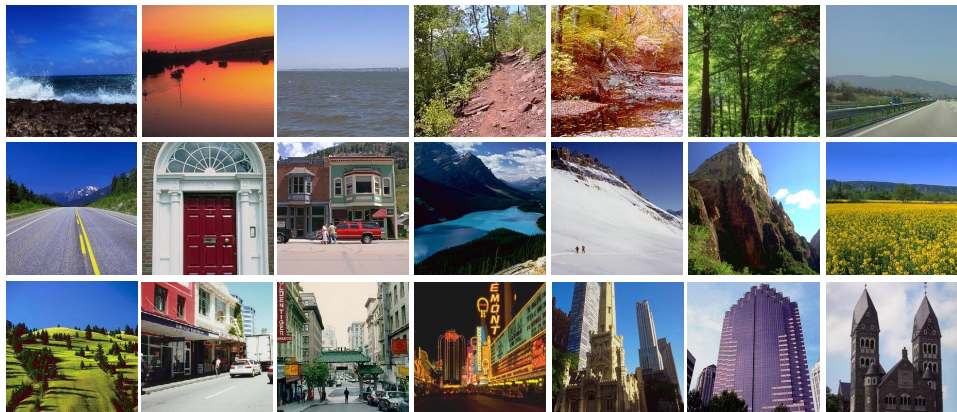
For image features  $\mathbf{x}$ , I use GIST [103] and LAB color histograms for Shoes and PubFig, and GIST alone for OSR, since the scenes do not seem well characterized by color.

I have released the features for Shoes, as well as the MTurk annotations

## Shoes



## OSR



## PubFig



Figure 3.3: Example images from the Shoes, OSR, and PubFig datasets.

for all three datasets, on <http://vision.cs.utexas.edu/whittlesearch/>. The features for OSR and PubFig are provided on <https://filebox.ece.vt.edu/~parikh/relative.html>.

**Methodology** For each query we select a random *target image* and score how well the search results match that target after feedback. This target stands in for a user’s mental model; it allows us to prompt multiple subjects for feedback on a well-defined visual concept, and to precisely judge how accurate results are. This part of my methodology is key to ensure consistent data collection and formal evaluation.

We use two metrics: (1) the ultimate *percentile rank* assigned to the user’s target image, which captures the fraction of images that are ranked *below* the target image, and (2) the *correlation* between the full ranking computed by  $S_t$  and a ground truth ranking that reflects the perceived relevance of all images in  $\mathcal{D}$ . Higher ranks are better, since that means the target image appears among the top-ranked search results presented to the user. Similarly, higher correlations are better. The two metrics give complementary information: while rank reveals how the exact target image ranks, NDCG reveals how many images very similar to the target are found among the top-ranked results. My method often produces a partial ordering where multiple images satisfy the same number of constraints; thus, we assign all  $n$  images that satisfy all constraints a *raw* rank of 1, then all images in the next equivalence class a raw rank of  $n + 1$ , and so on.

The correlation metric captures not only where the target itself ranks, but also how similar to the target the other top-ranked images are. We form the ground truth relevance ranking by sorting all images in  $\mathcal{D}$  by their distance to the given target. To ensure this distance reflects *perceived* relevance, we learn a metric based on human judgments. Specifically, we show 750 triplets of images  $(i, j, k)$  from each dataset to seven MTurk human subjects, and ask whether images  $i$  and  $j$  are more similar, or images  $i$  and  $k$ . Using their responses, we learn a linear combination of the image and attribute feature spaces that respects these constraints via [64]. Our ground truth rankings thus mimic human perception of image similarity. To score correlation, we use Normalized Discounted Cumulative Gain at top  $K$  (NDCG@K) [70], which scores how well the predicted ranking and the ground truth ranking agree, while emphasizing items ranked higher. We use  $K = 50$ , based on the typical number of images visible on a single page of image search results.

**Feedback generation** We use MTurk to gather feedback for my method and the binary feedback baseline. We pair each target image with 16 reference images. For our method we ask, “Is the target image more or less  $\langle$ attribute name $\rangle$  than the reference image?” (for each  $\langle$ attribute name $\rangle$ ), while for the baseline we ask, “Is the target image similar to or dissimilar from the reference image?” We also request a confidence level for each answer; see Figure 3.1. We get each pair labeled by five workers and use majority voting to reduce noise. When sampling from these constraints to impose feedback, we sort the

constraints by their average confidence level across all workers, and we take the most confident ones. In this way, our feedback generation simulates a live feedback session, in which a user will most likely comment on those properties of the target and reference images which are most evident.

Since the MTurk annotations are costly, for studies on the impact of iterative feedback, impact of the amount of feedback, and impact of the type of reference images, we generate feedback automatically. This works as follows. For relative constraints, we randomly sample constraints based on the predicted relative attribute values, checking how the target image relates to the reference images. In other words, the simulated user randomly chooses an attribute and one of the  $n$  top-ranked images at that round, and compares his target image to the chosen reference image along the given attribute dimension. For example, if the target’s predicted “shininess” is 0.5 and the reference image’s “shininess” is 0.6, then a valid constraint is that the target is “less shiny” than that reference image. For binary feedback, we analogously sample positive/negative reference examples based on their image feature distance to the true target. In particular, we sort the  $n$  currently top-ranked in terms of their Euclidean distance in raw feature space to the target image. We then generate constraints that say the top quartile of these images are “similar to” the target image, while the bottom quartile are “dissimilar from” the target. When scoring rank, we add Gaussian noise to the predicted attributes (for my method) and the SVM outputs (for the baseline), to coarsely mimic people’s uncertainty in constraint generation.



The automatically generated feedback is a good proxy for human feedback since the relative predictions are explicitly trained to represent annotator judgments. It allows me to test performance on a larger scale. Of course, like any simulated study, the simulated experiments I conduct have some limitations. For example, if noise has some other distribution than Gaussian, or users behave in some manner which is in dramatic contrast to the Gaussian noise model, the results might differ. However, I confirm the validity of my simulated experiments with the user-generated feedback experiments in Section 3.5.5.

As mentioned in Section 3.2, I train each attribute model with about 200 image pairs. The performance of attribute-based feedback may vary depending on the quality of the models. For example, it is possible that a smaller number of human feedback statements may be required to accomplish the same task if the attribute models are trained with more data and are therefore more reliable.

### 3.5.2 Impact of Iterative Feedback

First I examine how the rank of the target image improves as the methods iterate. Both methods start with the same random set of 16 reference images. At each round of feedback, both methods obtain eight automatically generated feedback constraints, each time re-scoring the data to revise the top reference images (using  $S_t^a$  and  $S_t^b$  for my method and the binary baseline,

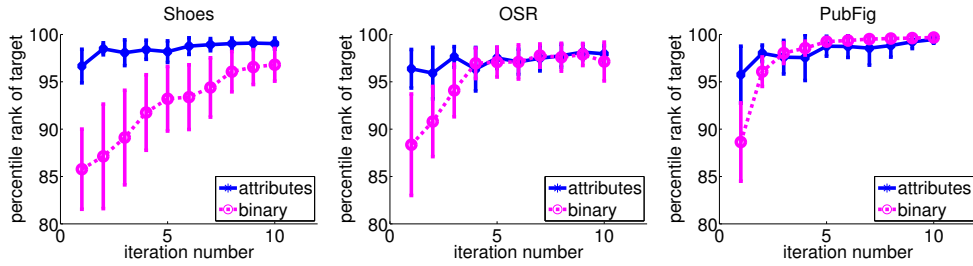


Figure 3.4: Impact of iterative feedback: Iteration experiments on the three datasets. My method often converges on the target image more rapidly.

respectively).<sup>4</sup>

Figure 3.4 shows the results, for 50 such queries. My method outperforms the binary feedback baseline for all datasets, more rapidly converging on a top rank for the target image. On PubFig my method’s advantage is slight, however. I suspect this is due to the strong category-based nature of the PubFig data, which makes it more amenable to binary feedback; adding positive labels on exemplars of the same person as the target image is quite effective. In contrast, on scenes and shoes where images have more fluid category boundaries, my approach’s advantage is much stronger. The searches tend to stabilize after 2-10 rounds of feedback. The run-times for my method and the baseline are similar.

---

<sup>4</sup>To ensure new feedback accumulates per iteration, we do not allow either method to reuse a reference image.

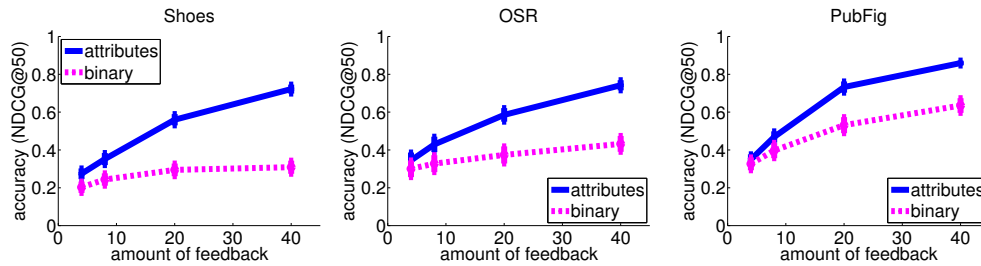


Figure 3.5: Impact of the amount of feedback: Ranking accuracy as a function of amount of feedback. While more feedback enhances both methods, the proposed attribute feedback yields faster gains per unit of feedback.

### 3.5.3 Impact of Amount of Feedback

Next I analyze the impact of the amount of feedback, using automatically generated constraints. Figure 3.5 shows the rank correlation results for 100 queries. These curves show the quality of all top-ranked results as a function of the amount of feedback given in a single iteration. Recall that a round of feedback consists of a relative attribute constraint or a binary label on one image, for my method or the baseline, respectively. For all datasets, both methods clearly improve with more feedback. However, the precision enabled by attribute feedback yields a greater “bang for the buck”—higher accuracy for fewer feedback constraints. The result is intuitive, since with my method users can better express *what about* the reference image is (ir)relevant to them, whereas with binary feedback they cannot.

A multi-attribute query baseline that ranks images by how many binary attributes they share with the target image achieves NDCG scores 40% weaker on average than my method when using 40 feedback constraints. This result

Dataset-Method	Near	Far	Near+Far	Mid
Shoes-Attribute	.39	.29	<b>.40</b>	.38
Shoes-Binary	.12	.05	<b>.27</b>	.06
PubFig-Attributes	<b>.60</b>	.41	.58	.52
PubFig-Binary	.39	.21	<b>.64</b>	.15
OSR-Attributes	<b>.53</b>	.27	.52	.40
OSR-Binary	.18	.18	<b>.32</b>	.11

Table 3.2: Impact of the reference images: Ranking accuracy (NDCG@50 scores) as we vary the type of reference images available for feedback.

supports my claim that binary attribute search lacks the expressiveness of iterative relative attribute feedback.

### 3.5.4 Impact of Reference Images

The results thus far assume that the initial reference images are randomly selected, which is appropriate when the search cannot be initialized with keyword search. We are interested in understanding the impact of the *types* of reference images available for feedback. Thus, we next control the pool of reference images to consist of one of four types: “near”, meaning images close to the target image, “far”, meaning images far from the target, “near+far”, meaning a 50-50 mix of both, and “mid”, meaning neither near nor far from the target. Nearness is judged in the GIST/color feature space.

Table 3.2 shows the resulting accuracies, for all types and all datasets using 100 queries and automatic feedback. Both methods generally do well with “near+far” reference images, which makes sense. For attributes, we expect useful feedback to entail statements about images that are similar to the

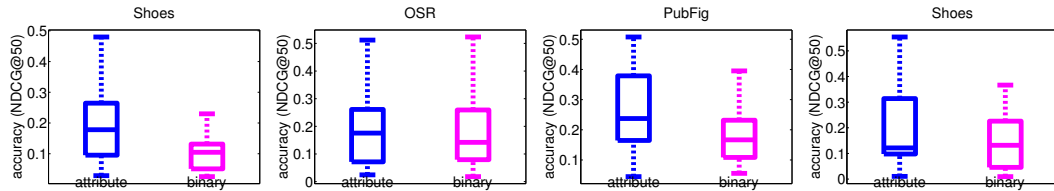


Figure 3.6: Ranking accuracy with user-generated feedback with randomly chosen (first three plots) and keyword-initialized reference images (fourth plot).

target overall, but lack some attribute. Meanwhile, for binary feedback, we expect useful feedback to contain a mix of good positives and negatives to train the classifier. We further see that attribute feedback also does fairly well with only “near” reference images; intuitively, it may be difficult to meaningfully constrain precise attribute differences on an image much too dissimilar from the target.

### 3.5.5 Ranking Accuracy with User-Given Feedback

Having analyzed in detail the key performance aspects with automatically generated feedback, now I report results using user-generated feedback. Figure 3.6 (first three plots) shows the ranking correlation for both methods on 16 queries per dataset after one round of eight feedback statements. Attribute feedback largely outperforms binary feedback, and does similarly well on OSR. One possible reason for the scenes being less amenable to attribute feedback is that people seem to have more confusion interpreting the attribute meanings (e.g., “amount of perspective” on a scene is less intuitive than “shininess” on shoes). In Chapters 5 and 6, I propose methods that will help account for these ambiguities and differences in user perception.

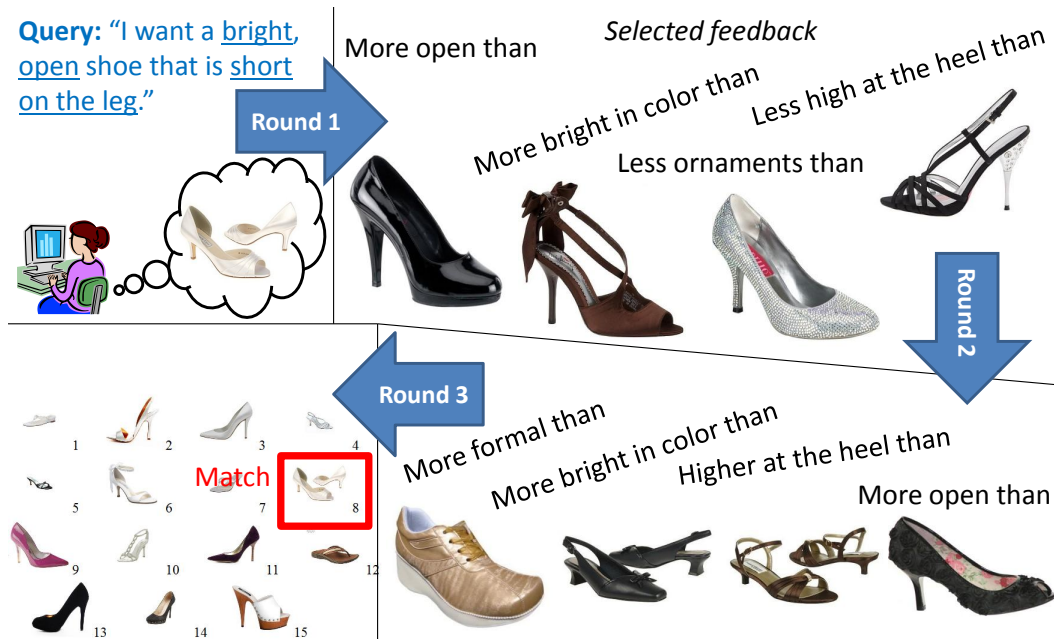


Figure 3.7: Example search result with iterative relative attribute feedback.

Next, I consider initialization with keyword search. The Shoes dataset provides a good testbed, since an online shopper is likely to kick off his search with descriptive keywords. Figure 3.6 (fourth plot) shows the ranking accuracy results for 16 queries when we restrict the reference images to those matching a keyword query composed of three attribute terms. Both methods get four feedback statements (I expect less total feedback to be sufficient for this setting, since the keywords already narrow the reference images to good exemplars). My method maintains its clear advantage over the binary baseline. This result shows (1) there is indeed room for refinement even after keyword search, and (2) the precision of attribute statements is beneficial.

Figure 3.7 shows a real example search using relative attribute feedback

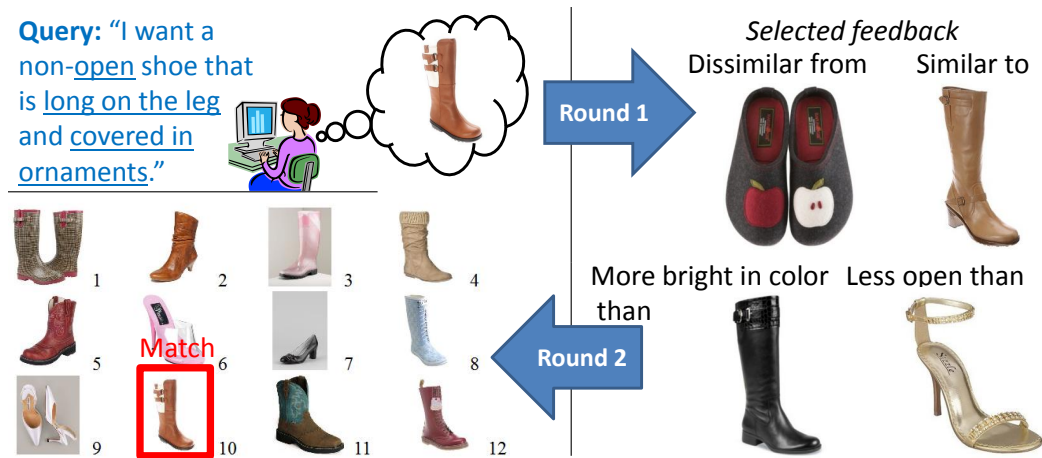


Figure 3.8: Example search result with hybrid feedback.

done on MTurk. Note how the user’s mental concept is quickly met by the returned images. In particular, the user can precisely pinpoint the shoe heel height, by making a “less” statement in Round 1 and a “more” statement in Round 2.

Figure 3.8 shows a real example using a hybrid of both binary and attribute feedback. This suggests how a user can specify a mix of both forms of input, which are often complementary.

### 3.5.6 Consistency of Relative Supervision Types

Finally, I examine the impact of how human judgments about relative attributes are collected.

**Class-level vs. instance-level** For all results above, we train the relative attribute rankers using image-level judgments. How well could we do if simply

	Class	Instance
Shoes	26.10%	<b>22.89%</b>
Scenes	38.92%	<b>33.41%</b>
Faces	<b>28.38%</b>	30.16%

Table 3.3: Errors for class- vs. instance-level attribute training.

training with class-based supervision, i.e., “coasts are more open than forests”? To find out, we use the relative ordering of classes given in [107] for PubFig and OSR, and define them for Shoes. We train ranking functions for each attribute using both modes of supervision. Table 3.3 shows the percentage of  $\sim 200$  test image pair orderings that are violated by either approach. Intuitively, instance-level supervision outperforms class-level supervision for Shoes and OSR, where categories are more fluid. Further, the 20 MTurkers’ inter-subject disagreement on instance-level responses was only 6%, versus 13% on category-level responses. Both results support the proposed instance-level design for relative attribute training.

**Absolute vs. relative** Finally, I analyze the consistency in people’s responses when asked to make *absolute* judgments about the strength of an attribute in a single image (on a scale of 1 to 3) as opposed to *relative* judgments for pairs of images (“more than”, “less than”, or “equal”). In a similar study as above, for absolute supervision, the majority vote over half the subjects disagreed with the majority vote over the other half 22% of the time. For relative responses, this disagreement was somewhat lower, at 17%. This indicates that the labels we obtain by requesting comparisons for relative at-



tributes are more reliable than the traditional approach of requesting absolute judgments.

### **3.6 Conclusions**

I proposed an effective new form of feedback for image search using relative attributes. In contrast to traditional binary relevance feedback which restricts the user’s input to labeling images as “relevant” or “not relevant”, my approach allows the user to precisely indicate how the results compare with his mental model. In-depth experiments with three diverse datasets show relative attribute feedback’s clear promise, and suggest interesting new directions for integrating multiple forms of feedback for image search.

Next, I study how to select the reference images used for feedback so the provided feedback is as informative to the retrieval system as possible.

## Chapter 4

# Active WhittleSearch: Attribute Pivots for Guiding Relevance Feedback in Image Search

In the previous chapter, I described how a user’s search can be made efficient through relative attribute relevance feedback. Traditionally in relevance feedback, the user is shown a page of results, and has the freedom to choose which images to mark as relevant/irrelevant. Similarly, the WhittleSearch approach I described in Chapter 3 employs such a free-form feedback interaction.

However, in principle, the choice of the image-attribute pairs on which feedback is provided can be left up to the user *or* the system. Depending on the application context, each of these options can be beneficial, as I will discuss in Section 4.4. In this chapter, I examine how the retrieval system can actively choose the image-attribute pairs on which it seeks feedback in the form of an attribute comparison.<sup>1</sup>

A user initiates a search with a multi-attribute query (e.g., “black high-

---

<sup>1</sup>This work was published in the Proceedings of the International Conference on Computer Vision (ICCV) 2013 with the title “Attribute Pivots for Guiding Relevance Feedback in Image Search” and authors Adriana Kovashka and Kristen Grauman. I wrote the code and conducted the experiments and data collection, while all authors contributed to developing the algorithm, devising the experiments, and writing the paper.

heels”) or a sample image (e.g., a snapshot of a pair of heels she saw). The goal of the approach presented below is to then refine the results. It interacts with the user through multiple-choice questions of the form: “Is the image you are looking for *more*, *less*, (or *equally*)  $A$  than image  $I$ ?”, where  $A$  is a semantic attribute and  $I$  is an exemplar from the database being searched. Our goal is to generate the series of such questions that will most efficiently narrow down the relevant images in the database, so that the user finds his target in few iterations. To this end, at each iteration we will actively select a comparison for the user to provide, that is, the  $(A, I)$  pair which yields the expected maximal information gain. Rather than exhaustively search all database images as potential exemplars, however, we consider only a small number of *pivot* exemplars—the internal nodes of binary search trees constructed for each attribute. The output of the system is the list of database images, sorted by their predicted relevance.

We again use a learning to rank approach to learn relative attributes, as in Section 3.2. I next explain how we construct attribute binary search trees (Section 4.1), and present my model of image relevance that accounts for the user’s attribute-based feedback (Section 4.2). The latter generalizes the one presented in Chapter 3 to use probabilities as opposed to strict decisions about attribute comparisons. Finally, I introduce my active selection approach to determine which comparison should be requested next (Section 4.3).

Similarly to Chapter 4, let  $\mathcal{D} = \{I_1, \dots, I_N\}$  denote the  $N$  images in the database, each of which has a corresponding image descriptor  $\mathbf{x}_1, \dots, \mathbf{x}_N$

(GIST and color, in our case). We have an attribute vocabulary consisting of  $M$  properties  $A_1, \dots, A_m, \dots, A_M$ . For example, for a shoe shopping database, those properties might be “pointiness”, “shininess”, “heel height”, etc. We use  $A_m(I_i)$  to denote the true strength of an attribute  $m$  in image  $I_i$ —that is, as would be perceived by a human viewer—and  $a_m(I_i)$  to denote the predicted attribute strength in image  $I_i$ —which is the only attribute strength to which our system has access.

## 4.1 Attribute Binary Search Trees

For each attribute  $m = 1, \dots, M$ , we construct a binary search tree. The tree recursively partitions all the database images into two balanced sets, where the key at a given node is the median relative attribute value occurring within the set of images passed to that node. To build the  $m$ -th attribute tree, we start at the root with all database images, sort them by their attribute values  $a_m(I_1), \dots, a_m(I_N)$ , and identify the median value. Let  $I_p$  denote the “pivot” image—the one that has the median attribute strength. Those images exhibiting the attribute less than  $I_p$ , i.e., all  $I_i$  such that  $a_m(I_i) \leq a_m(I_p)$ , are passed to the left child, while those exhibiting the attribute more, i.e.,  $a_m(I_i) > a_m(I_p)$ , are passed to the right child. Then the splitting repeats recursively, each time storing the next pivot image and its relative attribute value at the appropriate node.

Note that both the relative attribute ranker training and the search tree construction are offline procedures; they are performed once, before handling

any user queries.

Already, one could imagine a search procedure that walks a user through one such attribute tree, at each successively deeper level requesting a comparison to the pivot, and then eliminating the appropriate portion of the database depending on whether the user says “more” or “less”. However, there are two problems with such a simple approach. First, we cannot assume that the attribute predictions are identical to the attribute strengths a user will perceive; thus, a hard pruning of a full sub-tree is error-prone. Second, this approach fails to account for the variable information gain that could be achieved depending on *which attribute* is explored at any given round of feedback. Therefore, I propose a probabilistic representation of whether images satisfy the comparison constraints (Section 4.2). Further, I use the pivots to limit the pool of candidate images that are evaluated for their expected information gain (Section 4.3).

## 4.2 Predicting the Relevance of an Image

Now I explain how we predict the relevance of a database image, given the user’s comparative feedback. Let  $\mathcal{F} = \{(I_{p_m}, r)_k\}_{k=1}^T$  denote the set of comparative constraints accumulated in the  $T$  rounds of feedback so far. The  $k$ -th item in  $\mathcal{F}$  consists of a pivot image  $I_{p_m}$  for attribute  $m$ , and a user response  $r \in \{\text{“more”}, \text{“less”}, \text{“equally”}\}$ . The final output of our search system will be a sorting of the database images  $I_i \in \mathcal{D}$  according to their probability of relevance, given the image content and all user feedback.

Let  $G_{k,i} \in \{0, 1\}$  be a binary random variable representing whether image  $I_i$  satisfies the  $k$ -th feedback constraint. For example, if the user's  $k$ -th comparison yields response  $r = \text{"more"}$ , then  $G_{k,i} = 1$  if the database image  $I_i$  has attribute  $m$  more than the corresponding pivot image  $I_{p_m}$ . Let  $y_i \in \{1, 0\}$  denote the binary label for image  $I_i$ , which reflects whether it is relevant to the user (matches his target), or not. The probability of relevance is thus the probability that all  $T$  feedback comparisons in  $\mathcal{F}$  are satisfied:

$$P(y_i = 1 | I_i, \mathcal{F}) = \prod_{k=1}^T P(G_{k,i} = 1 | I_i, \mathcal{F}_k). \quad (4.1)$$

For numerical stability, we use a sum of log probabilities rather than a product:

$$P(y_i = 1 | I_i, \mathcal{F}) = \sum_{k=1}^T \log P(G_{k,i} = 1 | I_i, \mathcal{F}_k). \quad (4.2)$$

Recall Equation 3.7 in Chapter 3, where  $S_T(I_i)$  denotes the relevance of an image given all feedback received from the user.  $P(y_i = 1 | I_i, \mathcal{F})$  serves an analogous function here, the only difference being that we now have a soft (as opposed to 1/0) score of whether an image satisfies a constraint.

The probability that the  $k$ -th individual constraint is satisfied given that the user's response was  $r$  for pivot  $I_{p_m}$  is:

$$P(G_{k,i} = 1 | I_i, \mathcal{F}_k) = \begin{cases} P(A_m(I_i) > A_m(I_p)) & \text{if } r = \text{"more"} \\ P(A_m(I_i) < A_m(I_p)) & \text{if } r = \text{"less"} \\ P(A_m(I_i) = A_m(I_p)) & \text{if } r = \text{"equally"} \end{cases} \quad (4.3)$$

To estimate these probabilities, we map the attribute predictions  $a_m(\cdot)$  to probabilistic outputs, by adapting Platt’s method [111] to the paired classification problem implicit in the large-margin ranking objective. Specifically, this yields:

$$P(A_m(I_i) > A_m(I_p)) = \frac{1}{1 + \exp(\alpha_m(a_m(I_i) - a_m(I_p)) + \beta_m)} \quad (4.4)$$

$$P(A_m(I_i) = A_m(I_p)) = \frac{1}{1 + \exp(\gamma_m|a_m(I_i) - a_m(I_p)| + \delta_m)}, \quad (4.5)$$

where the sigmoid parameters are learned using the sets  $O_m$  (in which the first image in a pair has attribute  $m$  than the other) and  $E_m$  (in which two images have a similar strength of attribute  $m$ ) from Section 3.2. In particular, to learn  $\alpha_m$  and  $\beta_m$ , we use pairs with “more” judgments from  $O_m$  as positive paired-instances, and “less” judgments as negative instances. For  $\gamma_m$  and  $\delta_m$ , we use “equally” pairs from  $E_m$  as positive labels, and both “more” and “less” responses from  $O_m$  as negative instances. Note that  $P(A_m(I_i) < A_m(I_p)) = 1 - P(A_m(I_i) > A_m(I_p))$ . When estimating the likelihood of each possible user response (Sec. 4.3.2), we normalize these values so the three probabilities (“more”/“less”/“equally”) sum to 1.

My probabilistic model of relevance accounts for the fact that predicted attributes can deviate from true perceived attribute strengths. In Chapter 5 I will further develop a representation that accounts for differences in the user perception of attributes.

### 4.3 Actively Selecting an Informative Comparison

The proposed binary trees serve to guide the active exemplar selection and reduce its computational overhead, rather than completely eliminate images from consideration. Our system maintains a set of  $M$  current pivot images (one per attribute tree) at each iteration, denoted  $\mathcal{P} = \{I_{p_1}, \dots, I_{p_M}\}$ . The pivots are initially the root pivot images from each tree. During active selection, our goal is to identify the pivot in this set that, once compared by the user to his target, will most reduce the entropy of the relevance predictions on all database images. Note that selecting a pivot corresponds to selecting both an image as well as an attribute along which we want it to be compared;  $I_{p_m}$  refers to the pivot for attribute  $m$ .

#### 4.3.1 Entropy Reduction Objective

Given the feedback history  $\mathcal{F}$ , we want to predict the information gain across all  $N$  database images for each pivot in  $\mathcal{P}$ . We will request a comparison for the pivot that most reduces the total relevance entropy over all images—or equivalently, the pivot that minimizes the expected entropy when used to augment the current set of feedback constraints.

The entropy based on the feedback thus far is:

$$H(\mathcal{F}) = - \sum_{i=1}^N \sum_{\ell} P(y_i = \ell | I_i, \mathcal{F}) \log P(y_i = \ell | I_i, \mathcal{F}), \quad (4.6)$$

where  $\ell \in \{0, 1\}$ . Let  $R$  be a random variable denoting the user’s response,



$R \in \{\text{“more”}, \text{“less”}, \text{“equally”}\}$ . We select the next pivot for comparison as:

$$I_p^* = \arg \min_{I_{pm} \in \mathcal{P}} \sum_r P(R = r | I_{pm}, \mathcal{F}) H(\mathcal{F} \cup (I_{pm}, r)). \quad (4.7)$$

That is, the best pivot to inquire the user about is the one that is most likely to reduce relevance entropy. Here  $\mathcal{F} \cup (I_{pm}, r)$  refers to the expanded feedback set in which the pivot is paired with the feedback response  $r$ .

### 4.3.2 User Response Likelihood

Optimizing Equation 4.7 requires estimating the likelihood of each of the three possible user responses to a question we have not issued yet. I develop three possible strategies to estimate it. In each case, we use cues from the available feedback history to form a “proxy” for the user, essentially borrowing the probability that a new constraint is satisfied from previously seen feedback.

For the first strategy, which I call ALL RELEVANT, we use all relevant database images as the proxy. The assumption is that the images that are relevant to the user thus far are (on the whole) more likely to satisfy the user’s next feedback than those that are irrelevant. This is reminiscent of active classifier training, where posteriors estimated with the current classifier are used as weights in the expected entropy reduction of acquiring a new label. Ideally we would average the  $P(G_{c,i} = 1 | I_i, \mathcal{F}_k)$  values among only the relevant images  $I_i$ , where  $c$  indexes the candidate new feedback for a (yet unknown) user response  $R$ . Of course, we can only *predict* relevance, so we compute the

weighted probability of each possible response  $R$ :

$$P_{all}(R = r|I_{p_m}, \mathcal{F}) = \frac{1}{N} \sum_{i=1}^N P(y_i = 1|I_i, \mathcal{F})P(G_{c,i} = 1|I_i, \mathcal{F}_c), \quad (4.8)$$

where the *all* subscript stands for ALL RELEVANT.

The second strategy, which I call MOST RELEVANT, is similar, but uses only our current best guess for the target image as the proxy:

$$P_{most}(R = r|I_{p_m}, \mathcal{F}) = P(G_{c,b} = 1|I_b, \mathcal{F}_c), \quad (4.9)$$

where  $I_b$  is the database image that maximizes  $P(y_i = 1|I_i, \mathcal{F})$ , for  $i = 1, \dots, N$ .

The third strategy, which I call SIMILAR QUESTION, examines all previously answered feedback requests, and copies the answer from the question that is most similar to the new one. I define question similarity in terms of the Euclidean distance between the pivot images’ descriptors plus the similarity of the two attributes involved in either question. I quantify the latter by the Kendall’s  $\tau$  correlation [71] between the ranks they assign to a set of validation images. For example, this reflects that “feminine” and “heel height” are more aligned than “feminine” and “grayness”. Let  $r_k^*$  denote the response to the most similar question  $k$  found in the history  $\mathcal{F}$  for the new pivot  $I_{p_m}$  under consideration. Then we have:

$$P_{question}(R = r|I_{p_m}, \mathcal{F}) = \begin{cases} 1 & \text{if } r = r_k^* \\ 0 & \text{otherwise} . \end{cases} \quad (4.10)$$

I evaluate all three likelihood strategies in the results.

### 4.3.3 Recap of Interaction Loop

Figure 4.1 recaps the active WhittleSearch algorithm. At each iteration, we present the user with the pivot selected with Equation 4.7 and request the specified attribute comparison. In order for the user to monitor the search progress and stop if an image similar to his target has been found, we also show him the current top-ranked images. If further feedback is given, we first update  $\mathcal{F}$  with the user’s new image-attribute-response constraint. Then we either replace the pivot in  $\mathcal{P}$  for that attribute with its appropriate child pivot (i.e., the left or right child in the binary search tree if the response is “less” or “more”, respectively) or terminate the exploration of this tree (if the response is “equally”). Note that this means that the set of pivots consists of pointers into the binary trees at *varying* levels. See Figure 4.1. This is because our active selection criterion considers which attribute will most benefit from more refined feedback at any point in time.

Finally, the approach iterates until the user is satisfied with the top-ranked results, or until all of the attribute trees have bottomed out to an “equally” response from the user (in which case, the method can gain no further knowledge about the target given the available attribute vocabulary).

The cost of my selection method per round of feedback is  $O(MN)$ , where  $M$  is the size of the attribute vocabulary,  $N$  is the database size, and  $M \ll N$ . For each of  $O(M)$  pivots which can be used to complement the feedback set, we need to evaluate expected entropy for all  $N$  images. In contrast, a traditional information gain approach would scan all database items paired

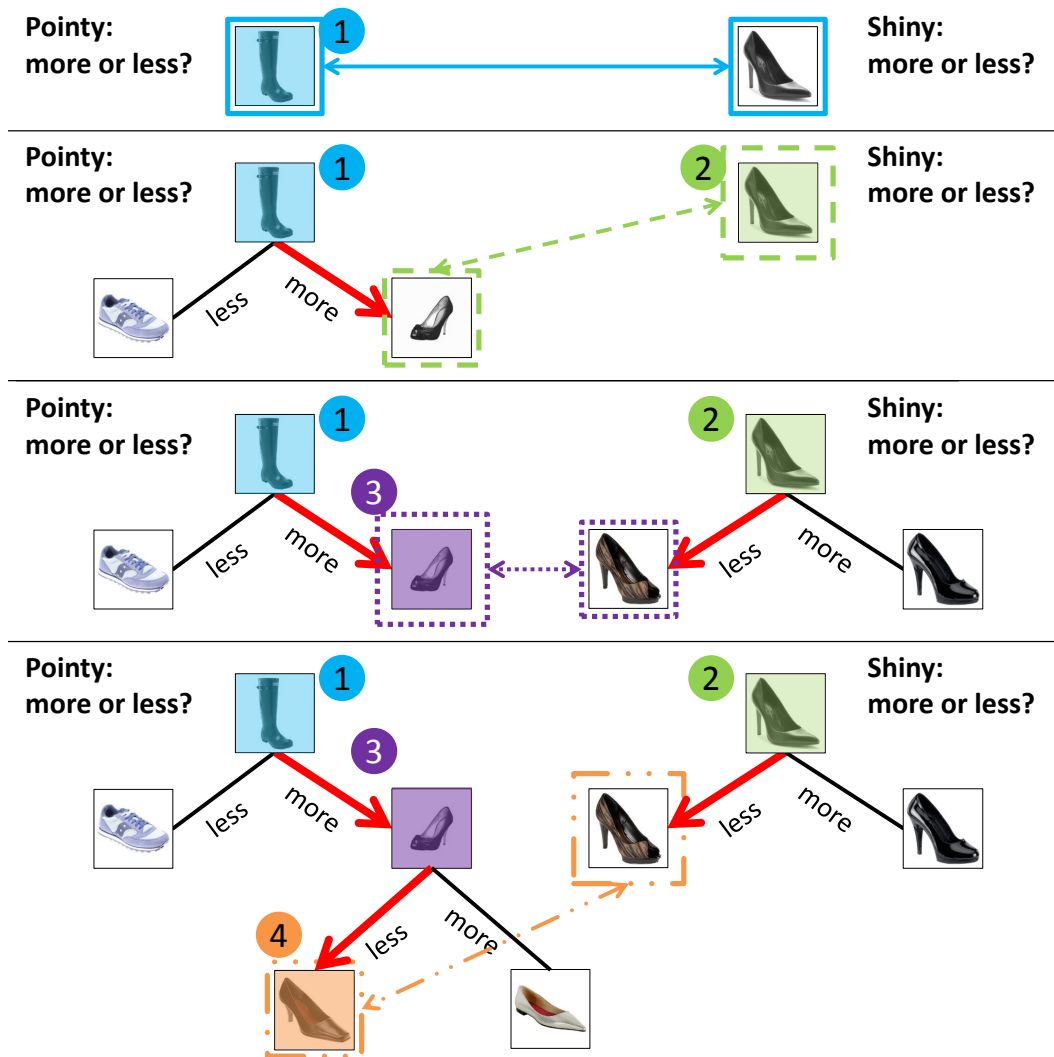


Figure 4.1: We request feedback on images that elicit the most information, using binary search trees to focus the active selection. In this sketch,  $M = 2$  attribute trees are shown. Images with the same color outline are the pairs considered at each round, and the number in this color marks the image chosen at this round. Red arrows denote the user's responses. Here, first the user is asked to compare his target to the boot pivot (1) in terms of pointiness; then he is asked to compare it to (2) in terms of shininess, followed by (3) in terms of pointiness, and so on. Best viewed in color.

with all attributes, requiring  $O(MN^2)$  time.

#### 4.3.4 Discussion

The basic idea of expected error reduction was first proposed in [122] for active learning in text classification, and variations have been explored in vision tasks (e.g., [17, 79, 100]). My formulation is novel in that the method surveys only the attribute pivots, exploiting the special structure of rankable visual properties for substantial computational savings. In contrast, existing work resorts to sampling heuristics [23], approximations [43], or simply small data pools to make the problem tractable.

Furthermore, as I will show in the results, the pivots also enhance selection *accuracy*, by essentially isolating those images likely to impact relevance predictions. Intuitively, if a user has ruled out a subtree (“The target image is bluer than the reference image with blueness  $X$ .”), it is likely redundant (low information gain) to ask how the target compares to more data on that path (“Is the target image bluer than this other reference image with blueness  $X - Y$ ?”), i.e., to ask the user to comment on something even less blue than the previous exemplar.

The attribute trees help us (softly) eliminate half of the search space in each round. Even if the images’ order along a given attribute is not *exactly* the same as the order that the user envisions, as long as these orders are roughly aligned, we would penalize the relevance scores of the correct images in each round. On the other hand, an exhaustive active approach that considers

entropy reduction resulting from feedback on each possible database image in turn can be misled by outliers that seem to have high expected information gain, with no way to regularize its selection.

My approach is a *probabilistic* variant of a *relative* 20-questions game, where the system greedily chooses the most useful question to ask at each round. Therefore, while it does not perform a hard pruning of tree branches, and it alternates between multiple trees, it benefits from the theoretical properties of standard binary search trees, which are known to have optimal performance as they reduce the search space in half with each subsequent split.

While my approach is myopic or greedy (it makes the choice that is optimal at each round but not necessary optimal overall), one could consider an extension which attempts to choose the optimal set of questions to ask as a batch.

#### **4.4 Conceptual Comparison of Free-form and Active WhittleSearch**

Next, I compare and contrast the two versions of my method: the first one introduced in Chapter 3 (which I refer to as *free-form WhittleSearch*), which initially picks reference images randomly and then presents those which are ranked highest by the system, seeking feedback from the user; and the second one introduced in this chapter (which I call *active WhittleSearch*), which asks the user for a visual comparison of the envisioned target image and an actively selected reference image along a given attribute.

Both the original free-form WhittleSearch and its active version have advantages over one another, which can be revealed under different scenarios. Active WhittleSearch makes a choice which is optimal with respect to the knowledge that the image search system possesses. This can be likened to a situation where we rely on a student's own understanding of what she knows, in order to improve her knowledge. However, unlike the free-form version of WhittleSearch, the set of images which are shown to the user for feedback are often disjoint from those that are ranked highest by the system. Therefore, the user must separately examine the images for feedback and the image results. The free-form WhittleSearch gives the user several options about the reference images and attributes on which to comment. Therefore, the performance of the system depends both on the choices that the user makes, as well as the correctness of the response which the user gives on the chosen pairing of image and attribute. In this case, we rely on the human "teacher" to know what additional information to give to the "learner" system. Free-form WhittleSearch requires more time for the completion of one feedback statement compared to active WhittleSearch, since it requires the user to examine a set of options and choose among them.

In cases when the user does not wish to spend a long time considering what image and attribute to comment on, I expect that active WhittleSearch will serve the user search goal better. For example, the user might choose to comment on those comparisons which are most obvious, but this might not be very informative to the system. However, if the user is careful and experienced

enough with the system interaction to pick informative comparisons, free-form WhittleSearch might perform better. For example, the user might see a unique attribute which is important for discriminating between relevant and irrelevant images, which the system has not asked about yet. This will be particularly important if there is large discrepancy between the human perception of an attribute and the system ranking for this attribute, in which case the entropy reduction estimates might be inaccurate. Another factor which affects how well the two versions of WhittleSearch perform is the number of feedback statements that the system has received so far. As I show in my results below (Section 4.5), the entropy-based selection criterion is most crucial early on in the iterative cycle, so I expect the advantage of active WhittleSearch over free-form WhittleSearch to be stronger in the first few iterations.

The level of specificity of the user’s mental model might affect the comparative performance of WhittleSearch’s two versions as well. If the user is simply browsing, the free-form WhittleSearch might be preferable as it gives the user more freedom to explore the current results and refine or terminate the search, depending on the precise qualities of the desired target. For example, a user shopping for a product with only a vague preconception of what is desired may be best suited by the free-form browsing approach. However, if the user has a very specific target in mind, active WhittleSearch might be more helpful as the user of binary search trees helps narrow down the search to the exact range of the attribute value distribution that matches the “signature” of the target image. The feasibility of browsing can be affected by the size of the



search interface—for example, it might be harder to browse reference images or results on a small mobile phone screen, which speaks in favor of eliminating user choice for the feedback statements, and pinpointing the exact object that the user has in mind (with active WhittleSearch).

## 4.5 Experimental Validation

In this section, I compare my active WhittleSearch method with pivots to several strong baselines. I demonstrate my method’s efficiency: it allows the user to retrieve high-quality results with fewer iterations than existing methods, and it also requires far less computational time than a traditional active selection method.

### 4.5.1 Experimental Design

I validate with the three public datasets also used in Chapter 3: **Shoes** [11], with the attributes from [78] (14,658 images and 10 attributes); outdoor scenes in **OSR** (2,688 images and 6 attributes); and celebrity faces in **PubFig** [83] (772 images and 11 attributes). I concatenate GIST and color features for Shoes and PubFig, and GIST alone for OSR. To train the relative attributes  $a_m(\cdot)$  and fit the sigmoid parameters in Section 4.2, I use the human judgment data collected in Chapter 3 (with about 200 image pairs per attribute) and provided online.

In order to quantify accuracy precisely, we tell the user which image to search for. That is, for a given search session, the user is instructed to

give feedback by comparing the target we specify to the various methods' selected exemplars. As in Chapter 3, we report the *percentile rank* each method assigns to the target at each iteration, defined as the fraction of database images ranked lower than the target. Higher percentile ranks are better; the ideal method would rank the target at the top of the search results page after very few iterations of feedback. Additionally, we measure the *NDCG@40 correlation* between the method's full ranking and the ground truth ranking. Higher correlations are better. To define the ground truth ranking, we sort all database images according to their perceptual distance (a learned metric on attributes and low-level features) from the target, as in Chapter 3.

In order for a user to participate in our studies, we require her to take a qualification test on the meaning of the attribute term. In this test, we first explain the attribute terms, by giving examples of pairs of images that are in a certain relationship for each attribute—e.g., the first image contains the attribute more than the second, or the two images contain it equally. Then for each attribute, we ask one or two questions on image pairs whose relationship is fairly clear, and is unlikely to be affected by slight variations in the user's perception of the attribute. This test helps eliminate some noise in the MTurk data collection process, weeding out users who do not understand the task or who are intentionally making errors.

### 4.5.2 Baselines

I compare my method ACTIVE ATTRIBUTE PIVOTS against the following six methods:

- ATTRIBUTE PIVOTS is a simplified version of my method that uses the proposed attribute trees to select candidate images, but cycles among the attributes in a round-robin fashion.
- ACTIVE ATTRIBUTE EXHAUSTIVE uses entropy to select questions like my method, but it evaluates all possible  $M \times N$  candidate questions.
- TOP selects the image that has the current highest probability of relevance and pairs it with a random attribute. This method represents traditional interactive methods that assume an “impatient” user for whom feedback exemplars and search results must be one and the same. It is similar in spirit to the WhittleSearch approach proposed in Chapter 3 in that top-ranked images are shown to the user, but the user is only shown a single image so there is no element of choice.
- PASSIVE selects a random image paired with a random attribute for its question.
- ACTIVE BINARY FEEDBACK does not use statements about the relative attribute strength of images, but rather asks the user whether the exemplar is similar to the target. This popular method uses a binary SVM to rank images, and treats similar images as positives and dissimilar images

as negatives. It actively chooses the image whose decision value is closest to 0, as in [152].

- **PASSIVE BINARY FEEDBACK** works as above, but randomly selects the images for feedback.

Note that relative feedback methods use the same relevance prediction function and only differ in the feedback they gather.

### 4.5.3 Results with Feedback by Simulated Users

To thoroughly test the methods, we first conduct experiments where we simulate the user’s responses.<sup>2</sup> For relative attribute feedback, we have to answer the question “Is the target image  $I_t$  more  $m$  than, less  $m$  than, or equally  $m$  as the pivot image  $I_{p_m}$ ?”, where  $m$  is an attribute. To the vector of relative attribute values for each attribute, we add Gaussian noise with  $\mu = 0$  and  $\sigma = 0.1s$ , where  $s$  is the standard deviation of values for that attribute. This results in predicted attribute values  $a'_m$  for each attribute  $m$ . We examine the predicted relative attribute values  $a'_m(I_t)$  and  $a'_m(I_{p_m})$ . If their difference is within a learned threshold, we generate a response of “equally”. This threshold is learned from MTurk annotation data. In particular, it is the average of the distances along  $a_m$  for training image pairs which have been

---

<sup>2</sup>The protocol is related to standard validation for active learning, where the algorithm receives the labels for those examples it queries, even if a person is not answering “live” in the loop. The predicted attribute values are an extrapolation of the ground-truth labels we have obtained from users. Note, gathering all possible comparisons in advance would cost \$2B if paying Turkers 1 cent each!

marked equal in terms of  $m$  by human judges, i.e., pairs that belong to the set  $E_m$ , as defined in Section 3.2. (We eliminate outlier training image pairs whose distances are more than one standard deviation of the values for the corresponding attribute.) If the threshold for equality is exceeded, we give a response of “more” if  $a'_m(I_t) > a'_m(I_{p_m})$  and “less” if  $a'_m(I_t) < a'_m(I_{p_m})$ . The addition of Gaussian noise is to account for the discrepancy between the user-perceived attributes  $A_m$  and our predictions  $a_m$ . By extrapolating a sparse set of real user judgments through a learned ranking function, we can perform large-scale comparisons and isolate the impact of my idea from the impact of the attribute rankers’ precision. We initialize all attribute search methods with the same feedback constraint.

For binary feedback, we have to answer the question “Is the target image  $I_t$  similar to or dissimilar from the exemplar image  $I_i$ ?” We respond with “similar” if the distance between  $I_t$  and  $I_i$  in terms of the learned perceptual distance between images is within one standard deviation of the distance between  $I_t$  and all other images in the database. Otherwise we respond with “dissimilar”. We add Gaussian noise with the same parameters as above to the SVM decision values. We initialize the binary feedback methods by peeking at the distances between the target image and a pool of 40 images, and selecting the closest image (Euclidean distance in feature space) as a positive and the furthest as a negative. This simulates a user starting the search with feedback on a page of random images. If anything, it is generous to the baseline, since our method gets only one “bit” of feedback at the onset, while the

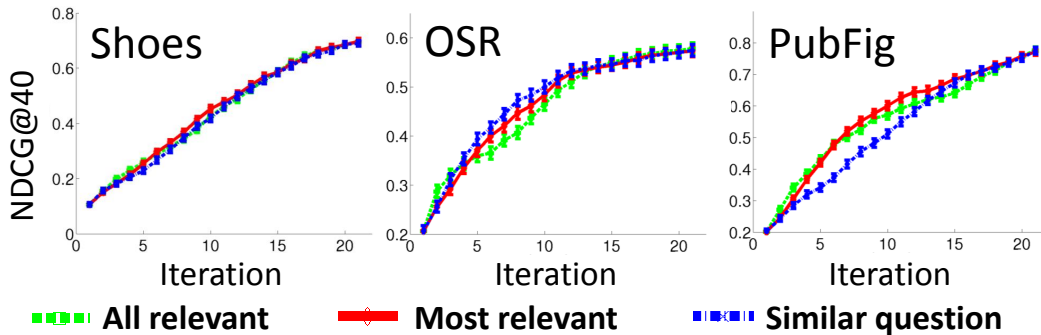


Figure 4.2: Comparison of the proposed models for the likelihood of a user’s response (higher curves are better). Best viewed in color.

binary feedback baselines get two.

I show all results over 200 randomly chosen queries (target images).

**Comparison of likelihood models** Figure 4.2 compares the three proposed methods of predicting the user response (Section 4.3.2). MOST RELEVANT consistently outperforms the other two methods on all but the OSR. This suggests that our best guess at the target tends to be a sufficient proxy, having a fairly similar attribute signature. ALL RELEVANT is slightly weaker, indicating that isolating the most relevant instance gives a “cleaner” likelihood than attempting to refine it with our uncertainty about each relevant instance. SIMILAR QUESTION performs the best for a fraction of the iterations on OSR, but does poorly on PubFig. This is likely because we cannot estimate attribute similarity reliably due to the distinct face attributes (e.g., face “chubbiness” has no strongly correlated attributes, whereas scene “openness” does). In all remaining results, we use the MOST RELEVANT method.

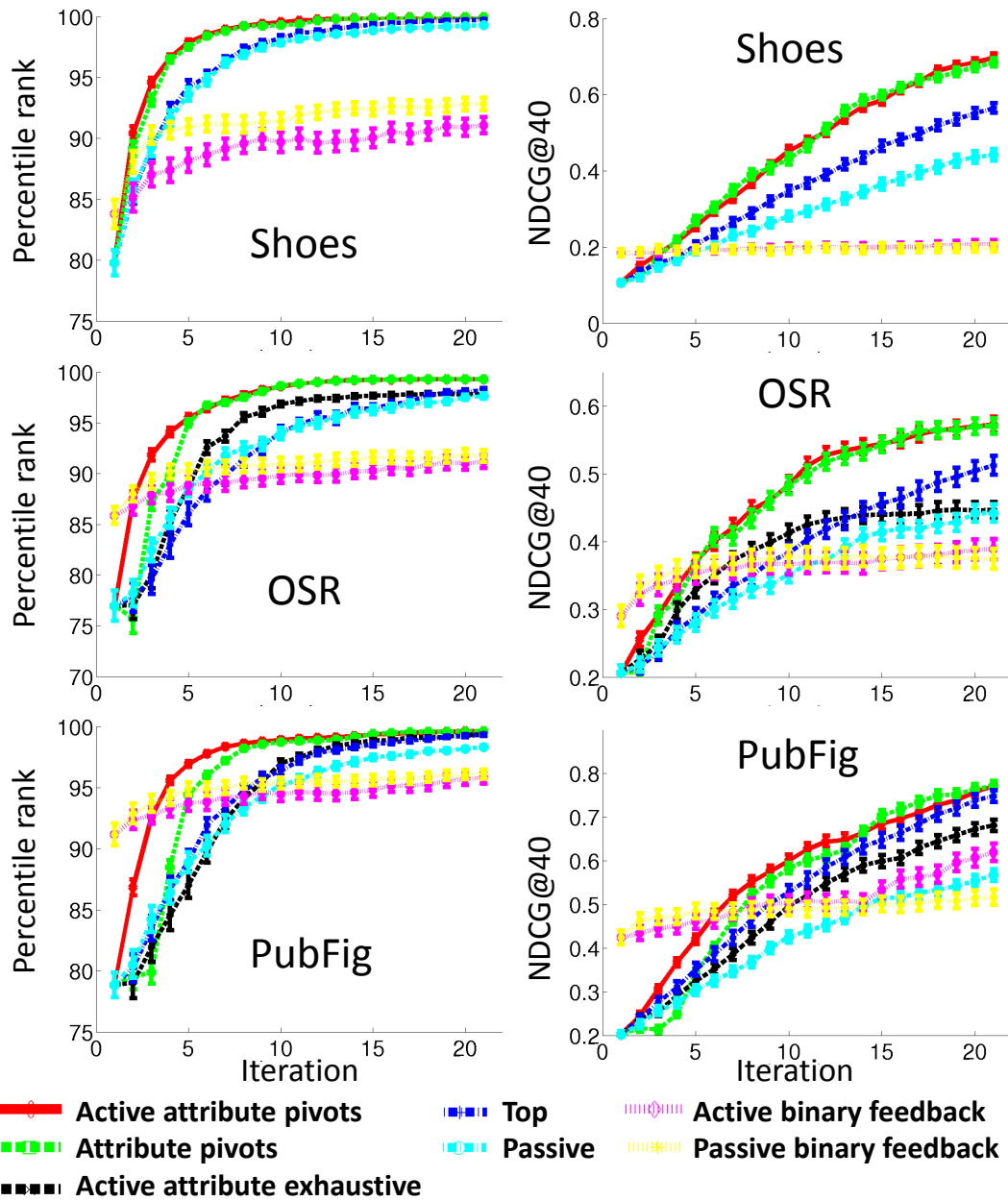


Figure 4.3: Comparison to existing interactive search methods (higher and steeper curves early on are better). Best viewed in color.

Method/Dataset	Shoes	OSR	PubFig
Active attribute pivots (Ours)	0.05	0.01	0.01
Active attribute exhaustive	656.27	28.20	3.42

Table 4.1: Selection time for 1 iteration of my method vs. the exhaustive active baseline, in seconds.

**Comparison to existing methods** Figure 4.3 compares all methods defined in Section 4.5.2 on all three datasets. Overall, my method finds the target image most efficiently. Not only does it outperform traditional passive selection (PASSIVE), but it also substantially improves over the TOP approach. This shows that relative attribute feedback alone does not offer the most efficient search; rather, my idea to actively elicit comparisons is essential. We also see that my full active approach outperforms the round-robin variant of my method (ATTRIBUTE PIVOTS), with an average percentile rank 7.6% better after only 3 iterations. This shows actively interleaving the trees allows us to focus on attributes that better distinguish the relevant images.

My method also outperforms ACTIVE ATTRIBUTE EXHAUSTIVE.<sup>3</sup> This shows that the attribute trees serve as a form of regularization, helping my method focus on those comparisons that *a priori* may be most informative<sup>4</sup>. Furthermore, my method is orders of magnitude faster (see Table 4.1).

---

<sup>3</sup>The exhaustive baseline was too expensive to run on all 14K Shoes. On a 1000-image subset, it does similarly as on other datasets.

<sup>4</sup>Note that my approach also outperforms the exhaustive active approach when we consider how fast these methods reduce the overall entropy of the system. However, if both methods were allowed to “peek” at the true user responses as opposed to estimating them, the exhaustive approach becomes an upper bound for the accuracy achievable by my approach.



The results confirm the striking advantage of attribute feedback compared to binary relevance feedback. Binary feedback has an advantage only in the first few iterations, likely because we generously initialize it with 2 feedback statements. We find that both feedback modes require similar user time: 6.4 s for relative, and 5.5 s for binary, and so the trends remain if we plot rank as a function of user time. Interestingly, we find that `PASSIVE BINARY FEEDBACK` is actually stronger than its active counterpart for this data. This is likely because images near the decision boundary were often negative, whereas the passive approach samples more diverse instances.

In practical terms, we are interested in how many iterations it takes to get the target in the top 40 most relevant images, since that is how many images fit on a typical search page (e.g., on Google). On average my method uses 12, 10, and 4 iterations to place the target in the top 40 for Shoes, OSR, and PubFig, vs. 21, 21, and 9 iterations for TOP. Thus, my method saves a user up to 70 seconds per query.

#### 4.5.4 Results with Live Users

Next, we test my method “live” in real time with Mechanical Turk workers. Note, this experiment is only possible because my method can make decisions in real time, unlike the exhaustive active method. We compare its performance against my `ATTRIBUTE PIVOTS` and the strongest baseline, TOP. We issue 50 queries for Shoes-1k (a random 1000-image subset of Shoes), OSR, and PubFig-Unique (one image for each of 200 individuals from the original

PubFig dataset [83], using the six most predictable attributes). For fairest comparison, we eliminate any queries where one or more methods did not receive 5 complete feedback iterations, leaving 34, 42, and 47 total queries for Shoes-1k, OSR, and PubFig-Unique, respectively. We stop updating the probabilities of relevance for a method once this method places the target image in the top 40 images. All methods share one simulated feedback statement at iteration 0, which we do not plot.

In order to get richer feedback from users, we allow users to express their confidence in their responses. Specifically, we allow them to say “a lot more” and “a lot less” in addition to “more”, “less”, and “equally”, as a way to express their confidence of an answer. We then give twice the weight to constraints for which the user says “a lot more (less)” when computing the relevance probabilities. We show users images from the bottom and top of our attribute rankers, in order to guide their answers and ameliorate the effect of the discrepancy between machine and user understanding of an attribute. For the live experiments, we restrict the set of exemplar images for Shoes-1k and OSR to 100 images that are diverse in terms of their attribute values. The target image is chosen as one of those 100. We do this in order to ensure that the questions posed to live users are not too difficult and the answers are not too subtle. However, we still always rank all the images according to their relevance for all methods.

For the live tests with Faces, we restrict the dataset to unique individuals because the dataset has strong category boundaries (i.e., the different

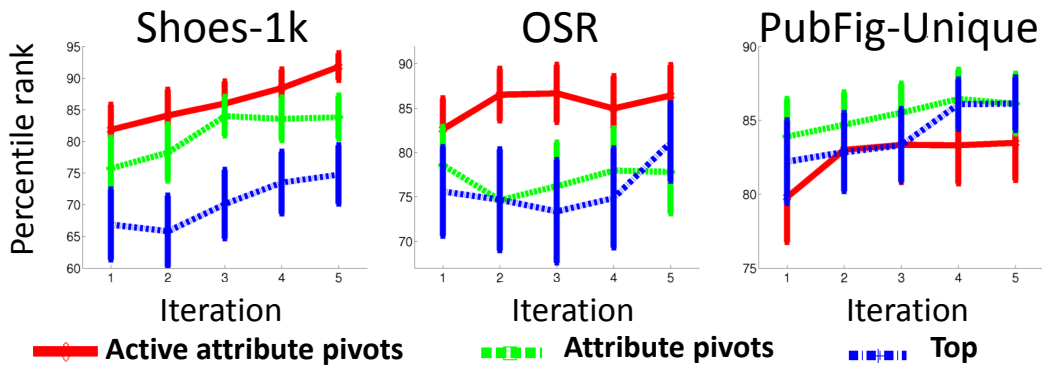


Figure 4.4: Results with live users: My method makes quick and reliable choices, allowing the MTurk users to more efficiently find the target.

celebrities) and low intra-class variation, so it does not make sense to compare one image of a person to another image of the same person.

Figure 4.4 shows the results. Consistent with the results above, we see that typically my method ranks the target image better than the baselines. We achieve a 100-200 *raw*<sup>5</sup> rank improvement on two datasets, and a negligible 0-10 *raw* rank loss on PubFig. This is a very encouraging result, given the noise inherent in MTurk responses (in spite of our best efforts at qualification tests) and the difficulty of predicting all attributes reliably. Our informativeness predictions on PubFig-Unique are imprecise since the facial attributes are difficult for both the system and people to compare reliably (e.g., it is hard to say who among two white people is whiter). This difficulty seems to hurt all methods, judging by their flatter curves. Since the rank metric does not give any credit for finding an image very close to the target, we also asked

<sup>5</sup>measured by number of as opposed to fraction of images



Figure 4.5: Example of a live search: Using the user’s feedback on the left, we retrieve the images on the right at the top of the results list.

a separate set of workers to judge whether any of the top 10 ranked images were “very similar” to the target. For Shoes-1k, my full method takes only 1.9 iterations on average to find one that is very similar, whereas my ATTRIBUTE PIVOTS require 2.4 and TOP requires 3.15.

Figure 4.5 shows an example search done by an MTurker. Notice how my method generates useful comparison questions across the different attributes, quickly converging on top-ranked images that look like the target.

### 4.5.5 Experimental Comparison of Free-form and Active WhittleSearch

The above results demonstrate the advantages of relative attribute feedback, as well as the benefit of actively selecting the images shown for such relative attribute feedback. In Section 4.4, I also discussed the advantages of the free-form WhittleSearch approach proposed in Chapter 3, and its active selection counterpart where choice is relegated to the image ranking system.

Next, we compare the two versions of WhittleSearch experimentally, using the Shoes dataset. We conduct experiments where users provide one feedback statement at each of five iterations, whether that is passively chosen based on an initial set of eight references images or those that are ranked highest at the previous iteration (for free-form WhittleSearch), or actively chosen (for active WhittleSearch). Each of twenty queries is submitted to five workers, and each worker completes the task for the same query for each method. Unlike in all other results, we take into account the time that each feedback statement requires, by timing the user responses at each iteration. We manually remove outliers in terms of time, and queries for which the users provided obviously incorrect responses, for any method.

In Figure 4.6 (a), I demonstrate that active WhittleSearch does indeed reduce the overall entropy of the system better than free-form WhittleSearch, albeit by a small margin. I plot how entropy decreases as the system receives more feedback, and the end point of both plots is an average that corresponds to the final ranks plotted in Figure 4.6 (b). The entropy estimates in the first

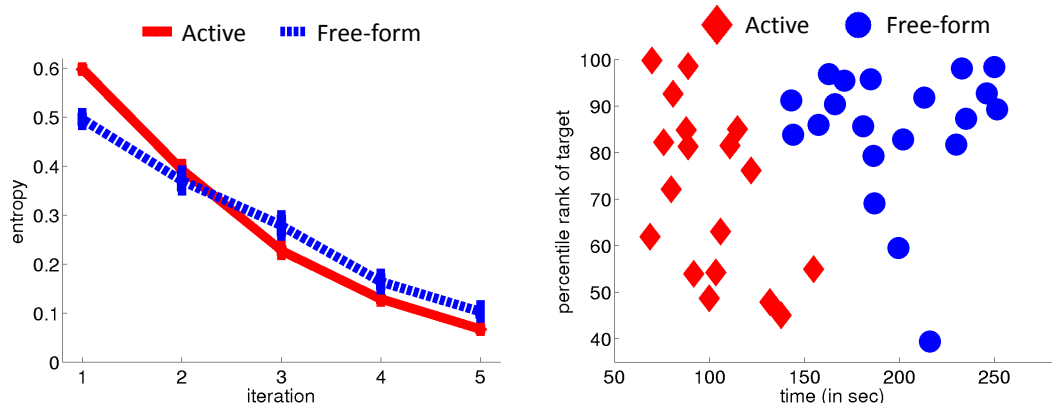


Figure 4.6: Comparison of free-form WhittleSearch and active WhittleSearch. (a) System entropy for free-form and active WhittleSearch (lower is better). (b) Percentile rank of target vs time required for feedback (higher rank and lower time is better).

few iterations are inaccurate due the system having received too little feedback to estimate relevance accurately, which likely explains why active WhittleSearch is initially weaker at reducing entropy, but after enough iterations, it starts to reduce entropy faster than the free-form version of WhittleSearch.

In Figure 4.6 (b), I plot the median final percentile rank of the target image per query, and the median total time it took to provide all feedback statements for that method. The time for feedback captures the time which users spend to examine the reference images and attribute vocabulary and consider the possible combinations thereof they can use for a feedback statement, and then actually submit the selected feedback. If no options are given and the system simply presents the user with a single question, then the time for feedback simply involves deciding on the answer to that question (i.e., “more”,

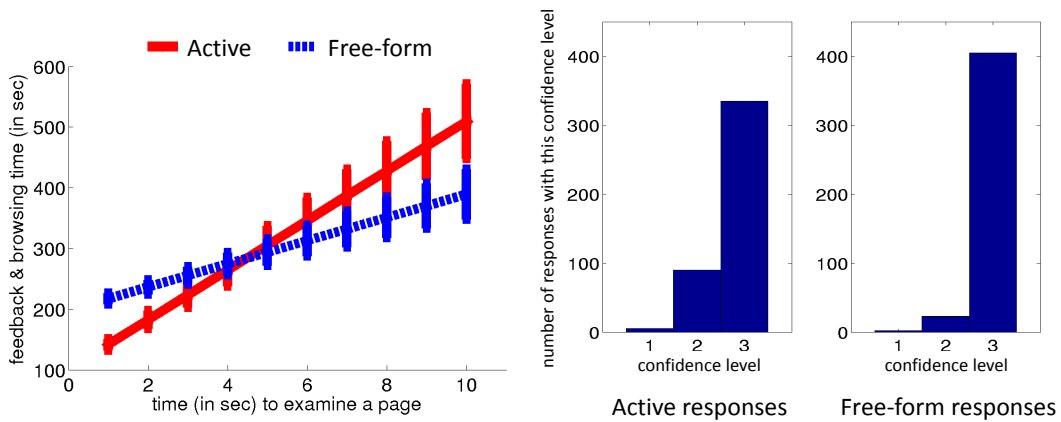


Figure 4.7: Comparison of free-form WhittleSearch and active WhittleSearch. (a) Total time, with rank converted to time (see text). (b) Confidence of user responses for free-form and active WhittleSearch.

“less”, or “equally”). Since free-form WhittleSearch gives the user more freedom and the user needs to examine options and select among them, that version requires significantly more time for feedback than the active version.

I also devise a unified metric which measures both how long it takes to provide a specific form of feedback, and how effectively this feedback enables the system to retrieve results, captured by the rank of the target image. This metric, which I plot in Figure 4.7 (a), sums the time for providing the feedback, and the time required to examine the results. The latter term, namely the time to examine the results pages, corresponds to the rank of the target image converted to the time required to find it at that rank, using a varying number of seconds that are required to examine a page of 40 images. In other words, if the target image is shown at rank 70, it will be on page two of the search results, and if it takes 4 seconds to examine a page, the total time to examine

the results will be 8 seconds. I allow the time to examine a page to vary because examining a page of results can take a short time, if the target image has very prominent and easy to spot distinctive features or if all of the results are obviously very different than the target image, or longer time, if some of the results are similar to the target and the user needs to look more carefully to determine if there is an actual match. I find that perusing a page of 40 image results takes 5.7 seconds on average, hence the choice of range I use on the x-axis of Figure 4.7 (a).

In Figure 4.6 (b), we see that active WhittleSearch is cheaper in terms of user time, but achieves slightly worse ranks for the target image. Because free-form WhittleSearch achieves better ranks than active WhittleSearch on average but is much slower than active, the free-form version outperforms the active one when the cost of examining a page of results starts to dominate the cost of providing feedback, as seen in Figure 4.7 (a).

To examine possible reasons for the performance of the two methods, in Figure 4.7 (b) I show a histogram of the confidences that users reported for their responses. I plot the average certainty that the user provided over the five iterations, with 3 being most certain and 1 being uncertain. We see that user responses on free-form WhittleSearch are much more certain than those for its active counterpart, likely because users often comment on the most obvious relationships of target and reference images when they are given a choice. This explains the inferior performance in terms of rank of active WhittleSearch. However, we observe that when all five MTurkers agree on all



of the active WhittleSearch responses, which occurred for one query (shown to five users) in my experiments, for four of the five users active WhittleSearch performed better than free-form WhittleSearch. This is encouraging because it indicates that if we can pick feedback requests that are informative *and* likely to be answered with confidence, my active approach will produce even more accurate search results. In Chapter 8, I discuss extending active WhittleSearch to account for the user’s ease of answering a question in the active selection formulation.

Example live searches using the two versions of WhittleSearch for the same query are shown in Figure 4.8. Observe the discriminative questions selected by the active system—not only in terms of attributes like “bright in color” and “long on the leg”, but also in terms of the images involved in the comparison along those attribute dimensions. For example, the user of WhittleSearch chose to comment on the relevant “long on the leg” property, but there are a lot more images that are less “long on the leg” than a boot (right), compared to those that are less “long on the leg” than a pump (left).

## 4.6 Conclusions

Today’s visual search systems place the burden on the user to initiate useful feedback by labeling images as relevant. They often prioritize showing the user pleasing results over striving to obtain useful feedback. In contrast, my system *actively* guides the search based on visual *comparisons*, helping a user navigate the image database via relative semantic properties. Compared to



**Target:**


<b>Active Whittle Search:</b>	<b>Free-form Whittle Search:</b>
<i>Less</i> <u>long on the leg</u> than	<i>More</i> <u>bright in color</u> than
<i>Equally</i> <u>open</u> as	<i>Less</i> <u>formal</u> than
<i>Less</i> <u>pointy</u> than	<i>Less</i> <u>long on the leg</u> than
<i>Less</i> <u>shiny</u> than	<i>Less</i> <u>ornamented</u> than
<i>More</i> <u>bright in color</u> than	<i>Equally</i> <u>bright in color</u> as
<b>Rank of target: 14</b> <b>NDCG@50: 0.295</b>	<b>Rank of target: 76</b> <b>NDCG@50: 0.152</b>
<b>Final ranking (top 10):</b>	<b>Final ranking (top 10):</b>
	

Figure 4.8: Qualitative comparison of free-form and active WhittleSearch: A case when active WhittleSearch is most useful. See text for an explanation.

existing active and passive methods, my pivot-based formulation is both more efficient (by orders of magnitude) and more accurate in practice. Results with both simulated and live users confirm that we can rapidly pinpoint the visual target using a series of well-chosen comparative queries.

So far, I have assumed that each attribute permits a universal interpretation. In the next chapter, I study ways to personalize results by building user-specific attribute models using explicitly requested labels or via implicit labels mined from a user's prior search sessions.

## Chapter 5

# Attribute Adaptation for Personalized Image Search

In the previous chapters, I described the power of relative attribute statements as a form of relevance feedback for search. I showed that this feedback enables the user to quickly find his search target, and I also described how the retrieval system can actively select the feedback to request.

Now I shift my attention to the user side of the search interaction. How does the user utilize the attribute vocabulary, and what does he mean when he uses each attribute term? Existing work does not account for differences in the user perception of attributes, and I show that this leads to suboptimal performance.

As discussed in Chapter 1, in existing work, the underlying assumption is that an image has a single “true” label per attribute that objective viewers could agree upon. However, multiple objective viewers are bound to have slightly different internal models of a visual property. Indeed, researchers collecting attribute-labeled datasets report significant disagreement among human annotators [40, 35, 110]. The differences may stem from several factors: the words for attributes are imprecise, their meanings often depend on context,

and they often stretch to refer to quite distinct object categories. For all such reasons, people inevitably craft their own definitions for visual attributes.

The variability in user perceptions of a given attribute has important implications for any application where a person uses attributes to communicate with a vision system. For example, in image search, a user requests images containing certain attributes [85, 142, 134, 78]; in recognition, a user teaches a system about objects by describing their properties [87, 40, 17, 107, 108]. Failing to account for user-specific notions of attributes will lead to discrepancies between the user’s precise intent and the message received by the system. Yet, even when training labels are solicited from multiple annotators, existing methods learn only a single “mainstream” view of each attribute, forcing a consensus through majority voting.

The goal of this part of my thesis is to account for the differences in the way that individual users might perceive a given attribute.<sup>1</sup> Furthermore, I want to learn user-specific models of attributes efficiently. For this purpose, I pose the problem as adaptation from a generic attribute model to a user-specific model. I use adapted SVM formulations that treat the generic model as a form of regularization. In this fashion, the system can learn the user’s perception with fewer labels than if it used a given user’s data alone.

---

<sup>1</sup>This work was published in the Proceedings of the International Conference on Computer Vision (ICCV) 2013 with the title “Attribute Adaptation for Personalized Image Search” and authors Adriana Kovashka and Kristen Grauman. I wrote the code and conducted the experiments and data collection, while all authors contributed to developing the algorithm, devising the experiments, and writing the paper.

I first train a *generic* model of an attribute using a large margin learning algorithm and data labeled with majority vote from multiple annotators. This is the “source” model, in transfer learning terms. Then, for a given user, the system adapts the parameters of the generic model to account for any user-specific labeled data, while not straying too far from the prior generic model. I refer to the resulting prediction function as an *adapted attribute* or *user-specific attribute*. This is the “target” model, in transfer learning terms.

In the following, I first overview the adaptation learning algorithms I use (Section 5.1). Then, I describe how my system uses the adapted attributes to perform personalized content-based image search (Section 5.2). Finally, I explain how it gathers explicit and implicit user-specific labeled data (Section 5.3).

## 5.1 Learning Adapted Attributes

Whereas thus far we have focused on relative attributes, for adaptation we consider two variants of attributes: binary attributes, which entail learning a classifier, and relative attributes, which entail learning a ranking function. For both, we perform adaptation with a large-margin formulation and a regularizer preferring user-specific parameters that do not deviate greatly from the generic parameters.

Adaptation requires that the source and target tasks be related, such that it is meaningful to constrain the target parameters to be close to the source’s. Whereas in some transfer problems this requires a “leap of faith”

and/or hand crafting (e.g., to specify that bicycle classifiers should transfer well to motorcycles), in our setting the assumption naturally holds. An attribute is semantically meaningful to all annotators, just with (usually slight) perceptual variations among them. Thus, we are assured that the generic model is a valid prior for each novel user we aim to adapt to.

We learn each attribute of interest separately (i.e., one classifier for “white”, another for “pointy”). Similarly, an adapted function is user-specific, with one distinct function for each user. In the following, we do not notate individual attributes or users to avoid subscript clutter.

Let  $D'$  denote the set of images labeled by majority vote that are used to learn the generic model. Let  $\mathbf{x}_i$  denote a feature describing the  $i$ -th image (texture, color), and  $y_i$  be its label. We assume the labeled examples originate from a pool of possibly many annotators who collectively represent the common denominator in attribute perception. We train a generic attribute  $f'(\mathbf{x}_i)$  from  $D'$ . Let  $D$  denote the set of user-labeled images, which is typically disjoint from  $D'$ . Both adaptive learning objectives below will take a  $D$  and  $f'$  as input, and produce an adapted attribute  $f$  as output.

### 5.1.1 Adapting Binary Attribute Classifiers

Binary attributes predict whether or not an attribute is present in an image. In this case, the generic data  $D'_b = \{\mathbf{x}'_i, y'_i\}_{i=1}^{N'}$  consists of  $N'$  labeled images, with  $y_i \in \{-1, +1\}$ . The subscript  $b$  denotes *binary*. Let  $f'_b$  denote the generic binary attribute classifier trained with  $D'_b$ . For a linear support

vector machine (SVM), we have  $f'_b(\mathbf{x}) = \mathbf{x}^T \mathbf{w}'_b$ . To adapt the parameters  $\mathbf{w}'_b$  to account for user-specific data  $D_b = \{\mathbf{x}_i, y_i\}_{i=1}^N$ , we use the Adaptive SVM [177] objective function:

$$\begin{aligned} \min_{\mathbf{w}_b} \quad & \frac{1}{2} \|\mathbf{w}_b - \mathbf{w}'_b\|^2 + C \sum_{i=1}^N \xi_i, \\ \text{subject to} \quad & y_i \mathbf{x}_i^T \mathbf{w}_b \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i \end{aligned} \quad (5.1)$$

where  $\mathbf{w}_b$  denotes the desired user-specific hyperplane, and  $C$  is a constant controlling the tradeoff between misclassification on the user-specific training examples and the regularizer. Note that the objective expands the usual large-margin regularizer  $\|\mathbf{w}_b\|^2$  to additionally prefer that  $\mathbf{w}_b$  be similar to  $\mathbf{w}'_b$ .<sup>2</sup> In this way, the generic attribute serves as a prior for the user-specific attribute, such that even with small amounts of user-labeled data we can learn an accurate predictor.

The optimal  $\mathbf{w}_b$  is found by solving a quadratic program to maximize the Lagrange dual objective function. This yields the Adaptive SVM decision function:

$$f_b(\mathbf{x}) = f'_b(\mathbf{x}) + \sum_{i=1}^N \alpha_i y_i \mathbf{x}^T \mathbf{x}_i, \quad (5.2)$$

where  $\boldsymbol{\alpha}$  denotes the Lagrange multipliers that define  $\mathbf{w}_b$ . Hence, the adapted attribute prediction is a combination of the generic model's prediction and similarities between the novel input  $\mathbf{x}$  and (selected) user-specific instances  $\mathbf{x}_i$ .

---

<sup>2</sup>See [5] for a variant that separates the transfer and margin regularizers.



### 5.1.2 Adapting Relative Attribute Rankers

As discussed above, rather than make a hard decision about attribute presence, relative attributes predict the strength of an attribute in an image [107]. In this case, labels are provided in terms of ordered pairs of examples:  $D'_r = \{(\mathbf{x}'_i, \mathbf{x}'_j)\}$ , where the subscript  $r$  denotes *relative*. Each pair denotes that image  $i$  exhibits the attribute more strongly than image  $j$ —for example, that  $i$  is “pointier” than  $j$ . Therefore, collecting  $D'_r$  requires asking multiple annotators to vote on which of the two images exhibit the attribute more. Implicitly, this corresponds to  $y'_i > y'_j$ , though during training the absolute strengths are irrelevant—only the comparative values matter. Following [107], we use a Rank SVM [64] approach to train each generic relative attribute. The Rank SVM seeks a hyperplane  $\mathbf{w}'_r$  that, when used to project all training data, (1) maintains their specified orderings, and (2) keeps a wide margin between the nearest projected points. For a linear ranker, we have  $f'_r(\mathbf{x}) = \mathbf{x}^T \mathbf{w}'_r$ .

To adapt the parameters  $\mathbf{w}'_r$  to account for user-specific ordered pairs  $D_r = \{(\mathbf{x}_{i_1}, \mathbf{x}_{i_2})\}_{i=1}^N$ , we use a Ranking Adaptation SVM [50]. It modifies the Rank SVM objective to add a regularizer that, similar to above, prefers that the resulting function stay close to the generic one. Specifically, to learn the adapted ranker, we optimize:

$$\begin{aligned} \min_{\mathbf{w}_r} \quad & \frac{1 - \delta}{2} \|\mathbf{w}_r\|^2 + \frac{\delta}{2} \|\mathbf{w}_r - \mathbf{w}'_r\|^2 + C \sum_{i=1}^N \xi_i & (5.3) \\ \text{subject to} \quad & \mathbf{w}_r^T \mathbf{x}_{i_1} - \mathbf{w}_r^T \mathbf{x}_{i_2} \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i, \end{aligned}$$

where  $\mathbf{w}_r$  denotes the user-specific hyperplane, and  $\delta \in [0, 1]$  is a constant balancing the two regularizers. The constraints reflect that the resulting  $\mathbf{w}_r$  ought to rank each  $\mathbf{x}_{i_1}$  higher than its corresponding  $\mathbf{x}_{i_2}$ , with a large margin. Again the solution requires solving a quadratic program [50], and the resulting adapted relative attribute predictor is:

$$f_r(\mathbf{x}) = \delta f'_r(\mathbf{x}) + \sum_{i=1}^N \beta_i \mathbf{x}^T (\mathbf{x}_{i_1} - \mathbf{x}_{i_2}), \quad (5.4)$$

where  $\beta$  denotes the Lagrange multipliers defining  $\mathbf{w}_r$ . Though shown here as linear functions, non-linear decision boundaries and rankers are also possible via kernelization.

### 5.1.3 Suitability for Adapted Attributes

Having defined the two adaptation methods, we can now reflect on their strengths for our problem. The adaptive formulations integrate the generic model and user-specific data during learning. This is preferable to independently training generic and user-specific models then combining their outputs, which is prone to overfit to the few available user-labeled examples. Intuitively, when optimizing Equation 5.1 or 5.3, a larger weight on a user-specific support vector  $\mathbf{x}_i$  is more likely when the generic model  $f'$  mispredicts  $\mathbf{x}_i$ , i.e., when  $f'_b(\mathbf{x}_i) \neq y_i$  or  $f'_r(\mathbf{x}_i) \not\asymp f'_r(\mathbf{x}_j)$ . Thus, user-specific instances that deviate from the generic model will have more impact on  $f$ . For example, suppose a user mostly agrees with the generic notion of “formal” shoes, but, unlike the average annotator, is also inclined to call loafers “formal”. Then the adapted classifier will likely exploit some user-labeled loafer image(s) with nonzero  $\alpha_i$

in Equation 5.2 when predicting whether a shoe would be perceived as formal by that user.

The adaptation strategy promotes efficiency in two ways. First, the human labeling cost is low, since the effort of the extensive label collection required to train the generic models is distributed among many users. Meanwhile, each user only needs to provide a small amount of labeled data. In experiments, we see substantial gains with as few as 12 user-labeled examples (Figure 5.5). Second, training time is substantially lower than training each user model from scratch by pooling the generic and user-specific data. We train the generic model once, offline, with a large pool of annotations. Then, the user-specific function is trained with a small amount of new data and the (already fixed) parameters  $\mathbf{w}'$ . This amortizes the “big” generic SVM’s training cost—superquadratic in the number of training examples—across all future user-specific functions we learn. The efficiency is especially valuable for personalized search, where we continually adapt a user’s attributes as his search history accumulates more user-specific data.

Finally, a more subtle advantage of my model choice is its modularity. The adaptation objectives do not require access to the generic training data. This is convenient, since in practice the data could be proprietary or simply unwieldy to pass around, yet one still would like to avoid learning personal attributes from scratch.

## 5.2 Personalized Image Search with Adapted Attributes

I next describe how we use the adapted attributes to personalize image search results. Compared to using generic attributes, the personalized results should more closely align with the user’s perception, leading to more precise retrieval of relevant images.

For binary attributes, we use the user-specific classifiers to retrieve images that match a multi-attribute query. Similar to [85], the user states “I want images with attributes  $X$ ,  $Y$ , and not  $Z$ ”. For relative attributes, we use the adapted rankers to retrieve images that agree with comparative relevance feedback. Similar to the interaction described in Chapter 3, the user states “I want images that show more of attribute  $X$  than image  $A$  and less of attribute  $Y$  than image  $B$ ”, etc. Then, in both cases, the system sorts the database images according to how confidently the adapted attribute predictions agree with the attribute constraints mentioned in the query or feedback. We use the magnitude of classifier/ranker outputs as confidences.

I stress that one can directly incorporate our adapted attributes into any existing attribute-search method [85, 142, 134].

## 5.3 Obtaining User-Specific Labeled Data

In order to learn an adapted attribute, we need to populate  $D$  with data annotated by the specific user. I present two forms of data collection: explicit and implicit.

### 5.3.1 Explicit Collection

Most directly, we ask the user to label a small set of images with the presence/absence of attributes (in the binary case) or pairs of images with comparative labels of the form “Image  $A$  is more/less/equally [*attribute name*] than Image  $B$ ” (in the relative case). We track worker IDs on MTurk to keep each user’s data separate. We convey the generic attribute meanings via qualification tests where we show examples of images that have and ones that do not have the attribute (in the binary attribute case), and pairs of images where the first one has the attribute less than or similarly to the second one (in the relative attribute case). We do so to start all users on roughly the same page as far as the meaning intended by the attribute name. While in some cases this could slightly obscure some user perceptual differences (and hence diminish the impact of my method), it is worthwhile to have sufficient definition of the generic attribute term. Requiring users to complete some very simple qualification tests helps ensure that the differences in their responses stem from perceptual and not linguistic reasons. In order to allow potential annotators to provide labels, we ask them to answer one, two, or four questions (depending on the dataset) of the same format as the examples correctly. The correct answers on the questions asked were determined by hand (for Shoes) or using examples where all labelers agreed (for SUN). See Figure 5.1 for an example qualification test for relative attribute annotations.

When collecting labels explicitly, the main consideration is how to select the images that the user should annotate. Intuitively, we want to focus on

\*\*\*IMPORTANT\*\*\*: Please read the following instructions carefully.

1) Please answer the following questions. You will first need to learn about the meaning of the attributes.

2) When deciding on how to answer these questions, please only base your answers on the information contained in the images.

3) Please try to give a "more" or "less" response, and reserve the "similarly" response only for cases when it is impossible (or very hard) to make a comparative judgment about the two images because they are too similar in terms of the given attribute.

Thank you!

---

The shoe in Image 1 is **less pointy at the front** than the shoe in Image 2.



Image 1



Image 2

The shoe in Image 1 is **similarly pointy at the front** as the shoe in Image 2.



Image 1



Image 2

---

QUESTION:

Is the shoe in Image 1 **more**, **less**, or **similarly pointy at the front** than/as the shoe in Image 2?



Image 1



Image 2

ANSWER A:

The shoe in Image 1 is **more pointy**.

ANSWER B:

The shoe in Image 1 is **less pointy**.

ANSWER C:

The shoe in Image 1 is **similarly pointy**.

Figure 5.1: MTurk qualification test for Shoes Relative. Users were required to get 2 of 2 questions right in order to provide user-specific labels.

examples for which his perception is likely to deviate from the generic model. Thus, we take an active learning approach. For binary attributes, we consider two forms. The first uses a margin criterion [152], requesting labels for those  $N$  images closest to the generic classifier’s hyperplane. For the second, we devise a variant of the query-by-committee criterion [137], requesting user-specific labels for the  $N$  images where the generic labels were most in disagreement.

While we find the margin criterion useful for binary attributes, for relative attributes it is less so. This is likely because it is hard to meaningfully choose which of two images has the attribute “more” when they are very close. Therefore, for relative attributes we adopt a simple diversity-based active selection scheme. We sort the candidate image pairs by their Euclidean distance in feature space, and request user comparisons on an even mix of the most similar, most dissimilar, and those surrounding the median.

My preliminary experiments indicated that actively obtained user-specific labels result in better models than passively obtained labels, as expected. Thus, we use them in all results.

### 5.3.2 Implicit Collection

Explicit labels offer the purest cues, but they also place some burden on the user. Therefore, I propose ways to infer “implicit” user-specific labels by mining the user’s relative attribute search history. I define two forms based on *transitivity* and *contradictions*.

In the transitivity case, we infer constraints on-the-fly when the user

gives feedback of the form “I want images *flatter* than Image  $A$  and *less flat* than Image  $C$ .” Let  $B$  denote the user’s mental target image—the item he envisions finding in the database. From his feedback, we now know that  $f_r(B) > f_r(A)$  and  $f_r(B) < f_r(C)$  in terms of “flatness”. By transitivity, we can infer a new user-specific label pair for  $D_r$  that requires that  $f_r(A) < f_r(C)$ .

In the contradictions case, we exploit seeming contradictions in a user’s relevance feedback. If he issues statements that appear contradictory according to the current model, he implicitly reveals a discrepancy between his perception and the system’s models. For example, if he says, “I want images *whiter* than  $A$  and *less white* than  $B$ ”, but the current “whiteness” model says  $f_r(A) \sim f_r(B)$ , then in principle the set of images satisfying the user’s model is empty.

Contradictions on the *same* attribute, while informative, are bound to be infrequent. Thus, we generalize this idea to the case where contradictions may occur *across* attributes. We discover which pairs of attributes are strongly correlated or anti-correlated<sup>3</sup>. Now treating strongly (anti-)correlated attributes as the same (opposite) attribute, we detect contradictions as described above, for images  $A$  and  $B$  that have the same approximate attribute rank. Consider Figure 5.2, where “feminine” and “sporty” are strongly anti-correlated. If the user requests images both “more feminine” than  $A$  and “more sporty” than  $B$ , where  $A$  and  $B$  are similarly feminine and similarly sporty, he seems to indicate that no images satisfy both constraints (green regions share

---

<sup>3</sup>We say two attributes are strongly correlated if they share at least a third of the images in their top or bottom quartiles.



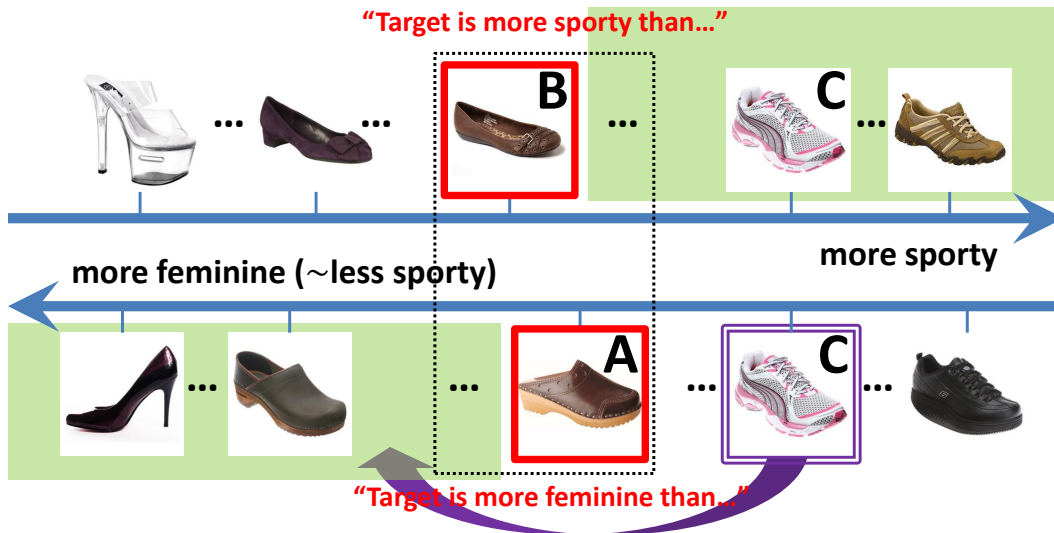


Figure 5.2: Example illustrating my idea for extracting implicit user-specific labels from a user’s search history. See text for details.

no images). This suggests his perception on one or both attributes differs from the current model  $f'_r$ . For example, perhaps he finds a pink sneaker  $C$  (which is high on “sportiness”) more “feminine” than clog  $A$ .

For each constraint in a contradictory pair, we select an image  $C$  that violates it by a small margin, and create an implicit user-specific pair using  $A$  and  $C$  in the reverse order of how the current generic attribute ranks them. In Figure 5.2, we create a pair “ $C$  is more feminine than  $A$ ”. By swapping the order, we correct the attribute model, and the theoretical set of images satisfying the user’s mental target is no longer empty (image  $C$  is now in both green regions). Thus, we have a better chance to align with the user’s perception.

Naturally, getting such labels “for free” carries some risk. We are not

guaranteed that a user would agree with all implicit labels, if asked. My approach can be viewed as a twist on self-training, a semi-supervised learning method in which one trains a classifier with labeled data, then uses it to classify unlabeled examples, and augments the labeled training set with the most confident predictions. As I demonstrate in the results, labels inferred from the user’s search history prove to be quite valuable.

## 5.4 Experimental Validation

To validate my idea, I experiment with 75 unique users on two large datasets. I evaluate adapted attributes in terms of both their generalization accuracy (Section 5.4.3) and their utility for personalized image search (Section 5.4.4).

### 5.4.1 Experimental Design

I use two datasets: **Shoes** [11], which contains 14,658 online shopping images describable by 10 attributes shown in Table 3.1, as in previous chapters, and **SUN Attributes** [110], which contains 14,340 scenes. I consider 12 attributes from SUN (see Table 5.1) that appear frequently and are likely to be relevant for image search applications. Shoes has both binary and relative attributes; SUN has only binary attributes.

Since SUN comes with annotators’ label votes, we use the query-by-committee criterion to solicit user-specific labels. Since Shoes does not, we use the margin criterion (see Section 5.3). These datasets represent the largest and

Shoes	SUN
pointy at the front	sailing/ boating
open	vacationing/ touring
bright in color	hiking
covered with ornaments	camping
shiny	socializing
high at the heel	shopping
long on the leg	vegetation
formal	clouds
sporty	natural light
feminine	cold
	open area
	far-away horizon

Table 5.1: The Shoes and subset of SUN attributes used in my experiments.

most challenging attribute-labeled collections available today, and they allow us to observe the impact of adaptation for both a narrow class of objects (Shoes) as well as a wide domain of scenes (SUN). Figure 5.3 shows some images from the SUN dataset. Please refer to Figure 3.3 in Chapter 3 for images of the Shoes dataset.

To form descriptors  $\mathbf{x}$  for Shoes, we use the GIST and color histograms as before. For SUN, we concatenate features provided by [110]: GIST, color, and base HOG and self-similarity. We cross-validate  $\delta$  and  $C$  for all models, per attribute and user.

#### 5.4.2 Baselines

I compare my USER-ADAPTIVE approach to the following methods:

- **GENERIC**: learns a model from the generic majority vote data  $D'$  only.

# SUN

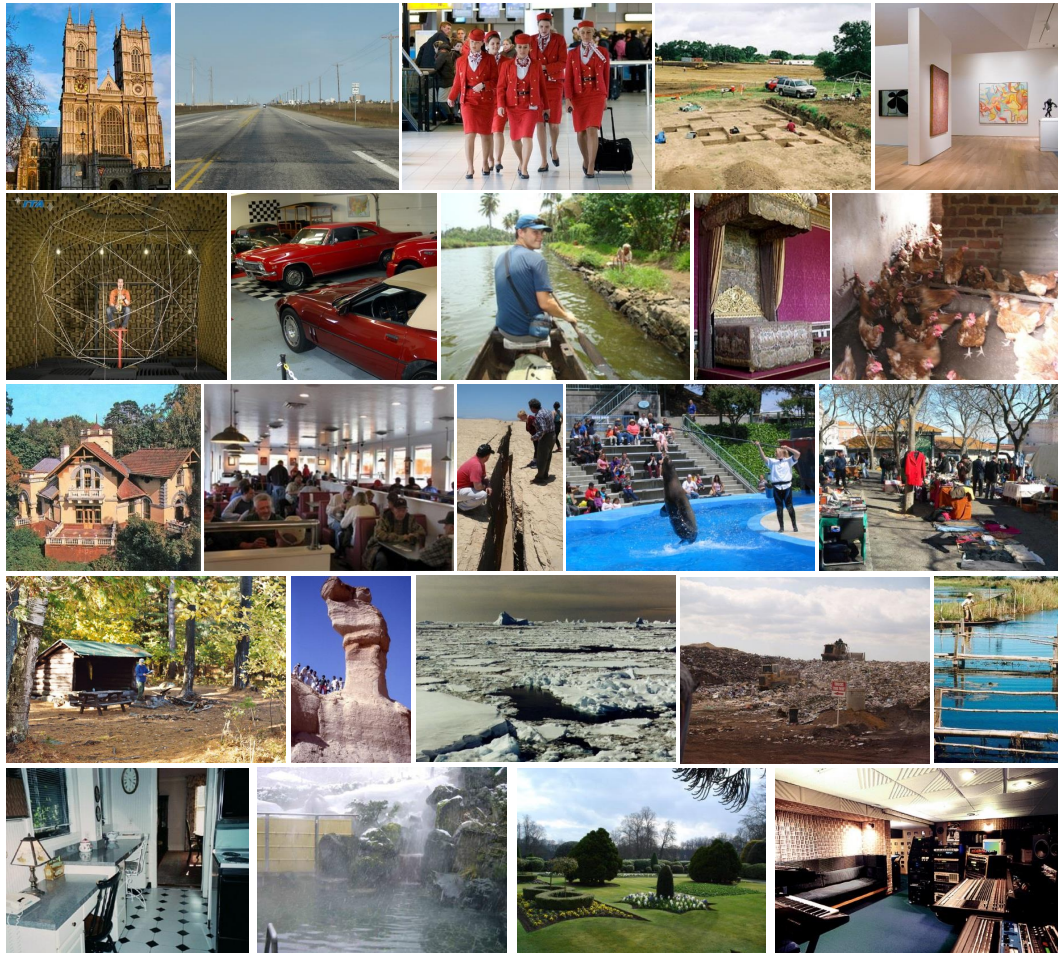


Figure 5.3: Example images from the SUN dataset.

This is how attributes are learned in prior work (e.g., [87, 40, 17, 107, 85, 134]).

- **GENERIC+**: is just like above, but uses more generic data. For every additional user-specific label my method gets, it gets an additional generic label from some other user. This baseline lets us compare the effect of adapting to user-specific data versus simply adding more generic data.
- **USER-EXCLUSIVE**: learns a user-specific model from scratch, without the generic model. It always uses the exact same user-specific data as my method. This baseline lets us see how much my method benefits from regularization with the generic model.

Aside from these distinctions, all methods use the exact same features and learning algorithms.

### 5.4.3 Adapted Attribute Accuracy

First we evaluate generalization accuracy: will adapted attributes better agree with a user’s perception in novel images? To form a generic model for each dataset, we use 100-200 images (or pairs, in the case of Shoes Relative) labeled by majority vote. We collect user-specific labels on 60 images/pairs, from each of 10 (Shoes) or 5 (SUN) workers on MTurk. We reserve 10 random user-labeled images per user as a test set in each run, and show average accuracy and standard error across 300 random splits.

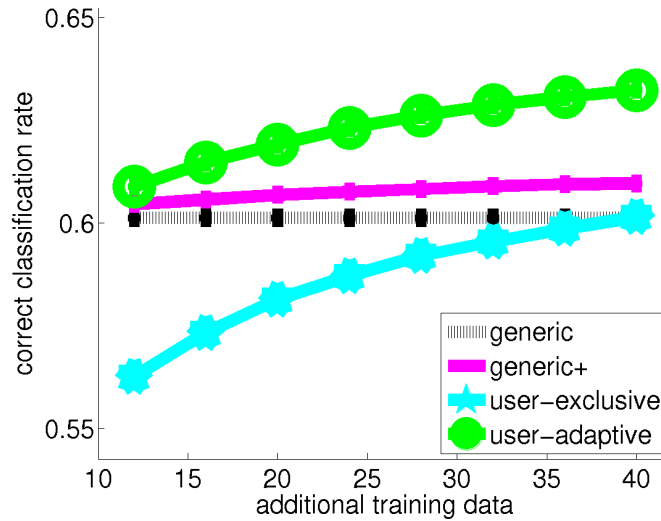


Figure 5.4: Adapted attribute prediction accuracy: Average performance and standard error over all datasets, all attributes, and all users.

Figure 5.4 shows an average over all datasets, attributes, and users, and Figure 5.5 shows representative results for individual attributes and individual users. We plot test accuracy as a function of the amount of additional training data beyond the generic pool  $D'$ . GENERIC remains flat, as it gets no additional data. For binary attributes, chance is 50%; for relative it is 33%, since there are three possible responses (“more”, “less”, “equally”).

My adapted attributes typically outperform all other methods. Their advantage over the generic model supports my main claim: we need to account for users’ individual perception when learning attributes. Further, the advantage over the user-exclusive model shows my approach successfully leverages “universal” perception as a prior; learning from scratch is inferior, par-

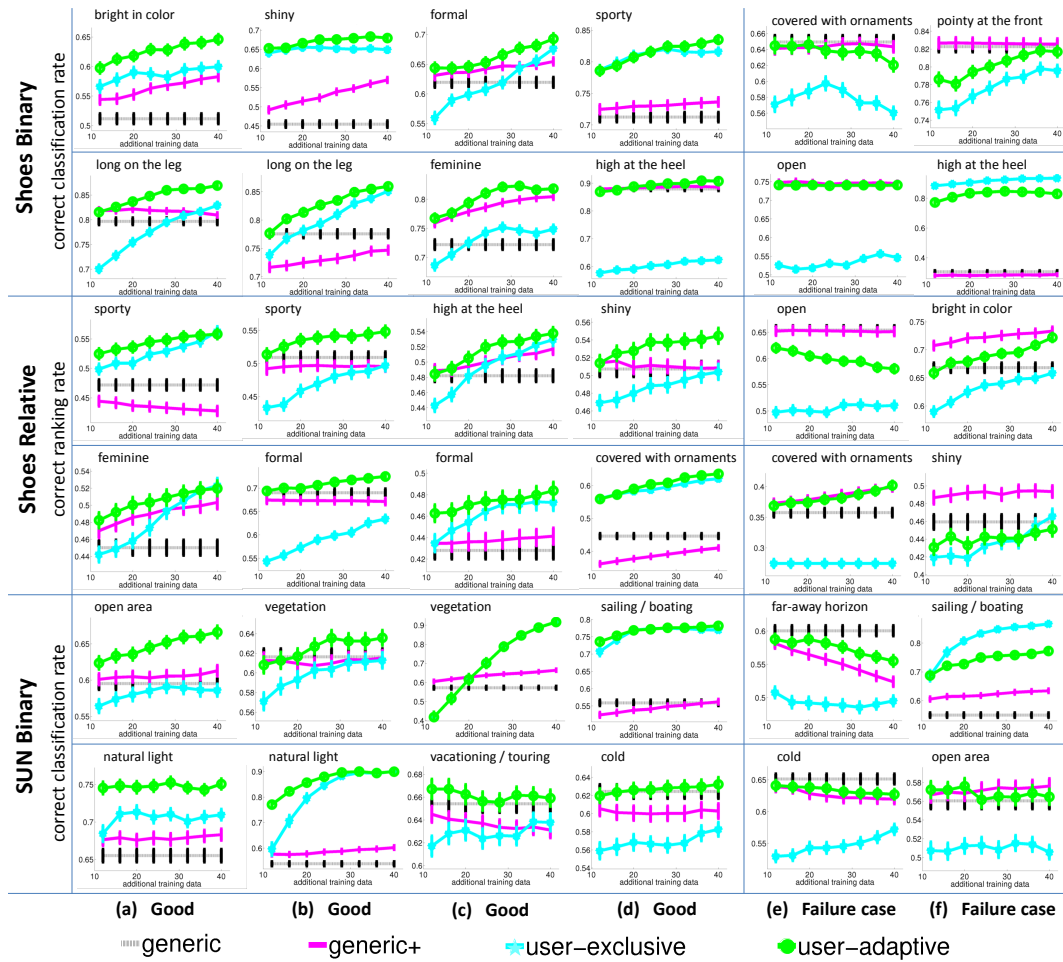


Figure 5.5: Adapted attribute prediction accuracy: Individual per-attribute per-user plots, as more training data is added.

ticularly if very few user-specific labels are available (see leftmost points on plots). With more user-specific labels, the non-adaptive approach can sometimes catch up (see “feminine” in Figure 5.5 column (a)), but at the expense of a much higher burden on each user. Finally, the GENERIC+ baseline confirms that my method’s advantage is not simply a matter of having more data available. GENERIC+ usually gives generic a bump, but much less than USER-ADAPTIVE. For example, on “bright in color”, my method improves accuracy by up to 26%, whereas GENERIC+ only gains 14%.

We do see some failure cases though, as shown in columns (e) and (f). The failures are by definition rather hard to analyze. That’s because by focusing on user-specific perception, we lose any ability to filter noisy label responses (e.g., with voting). So, when a user-adapted model misclassifies, we cannot rule out the possibility that the worker himself was *inconsistent with his personal perception* of the attribute in that test case. Nonetheless, we do see a trend in the failure cases—weaker USER-EXCLUSIVE classifiers. As a result, my model can start to underperform GENERIC, pulled down by (what are possibly inconsistent) user responses, as seen by a number of cases where the user-exclusive baseline remains close to chance on binary attributes. Another reason for failure (with respect to the user-exclusive model) were user responses which were the opposite of generic responses (see “high at the heel” for Shoes Binary in column (f)), where the generic prior can cause negative transfer for my method.

Note that the success of adaptation depends not just on the attribute



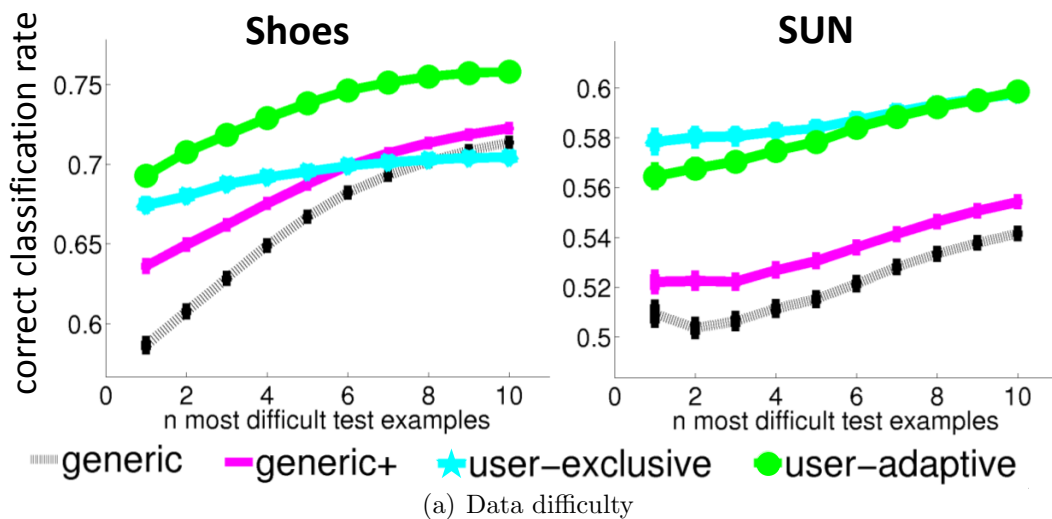
being learned, but also on individual users. For example, for “open area” (SUN Binary), my method performs well for one user (column (a)) and poorly for another (column (f)). This motivates one of the extensions of this approach which I describe as future work in Chapter 8, namely detecting internal consistency in a user’s responses, which should be taken into account when deciding when/how to adapt. However, for some attributes like binary “high at the heel” (column (d)), we see a similar impact for most users.

Figure 5.6 shows example attribute spectra for three generic and adapted attribute predictions, sorted from least to most. They illustrate how my method captures user-specific nuances in attribute meaning. In the top set, it learns that this user perceives flat fancy shoes to be “feminine”, whereas the generic impression is that high-heeled shoes are more feminine. In the middle set, it learns that for this user, shoes that are darker in color are more “formal”, whereas the generic model says shoes similar but brighter in color are formal. In the bottom set, it learns that this user finds landscapes with mountains more “vacation-like” than other settings.

Figure 5.7 analyzes my method’s impact as a function of task difficulty, using all 40 training labels. First, we consider test case difficulty (a), as measured by the distance to the binary attribute generic hyperplane; closer instances are more difficult. We sort the 10 test examples per split by difficulty, and average over all attributes and users. We then plot accuracy as we add increasingly easy examples to the test set. We see that user-adapted attributes are often strongest when test cases are hardest. This is intuitive, since the



Figure 5.6: Example learned generic (top row per example) and user-specific (bottom row per example) attribute spectra.



(a) Data difficulty

	Most divergent	All
GENERIC	58.66 (0.35)	71.38 (0.11)
USER-EXCLUSIVE	71.86 (0.33)	70.54 (0.11)
USER-ADAPTIVE	69.91 (0.29)	75.78 (0.10)

(b) User difficulty

Figure 5.7: Adapted attribute accuracy as a function of task difficulty. Best viewed in color.

intent of my method is to capture what may be subtle, fine-grained perceived differences. For SUN attributes, the user-exclusive model outperforms mine by a small margin for the most difficult examples, likely because binary judgments are hard to make for some of these attributes, making the generic prior less valuable.

As a second form of analysis of task difficulty, in Figure 5.7(b) we consider *user* difficulty on the Shoes Binary dataset, as measured by how often a worker disagrees with the majority. Numbers in parentheses are standard error over all binary shoe attributes and random splits. The margin between

our adaptive method and the generic method is significantly increased for divergent workers (left column) compared to all workers (right column), as the generic model is insufficient when the user has a unique perception. In contrast, my method faithfully captures users' notions.

#### 5.4.4 Personalized Search with Adapted Attributes

Next I show that correctly capturing attribute perception is important for accurate search. The same user perception underlies both classification (“is the image red?”) and search (“get all red images”). Search is a key application where adapted attributes can alleviate inconsistencies between what the user says, and what the (traditionally majority-vote-trained) machine understands. For all search results, we use the attributes that seem most in need of adaptation, based on my previous results (5 for Shoes, 4 for SUN). A user might query an image database by stating “Show me shoes that are pointy”, thus performing a keyword search using the term “pointy”. Then they might refine the search by stating “These are good, but I want them to be higher at the heel”, thus providing relevance feedback using relative attributes.

**Multi-attribute keyword search** First we evaluate multi-attribute keyword queries. We ask 10 MTurkers to label 40 images for each of the attributes. We train all models, then apply them to a test set of 20 held-out images per user. We issue all combinations of three-attribute queries. Accuracy is the percentage of test images where the binary predictions on all three

	GENERIC	GENERIC+	USER-EXCLUSIVE	USER-ADAPTIVE
Shoes-B	31.5 (0.13)	36.3 (0.14)	40.3 (0.15)	<b>43.6</b> (0.13)
SUN	34.3 (0.19)	47.3 (0.15)	51.9 (0.24)	<b>64.5</b> (0.16)

Table 5.2: Personalized image search accuracy: Multi-attribute keyword search.

	GENERIC	GENERIC+	USER-EXCLUSIVE	USER-ADAPTIVE
Shoes-R	70.96 (0.12)	72.70 (0.10)	72.75 (0.14)	<b>74.70</b> (0.12)

Table 5.3: Personalized image search accuracy: Relative attribute search feedback.

query attributes agree with that user’s ground truth.

Table 5.2 shows the results, averaged over all users and queries. We see that the generalization power of the adapted attributes translates into the search setting. My method is substantially better at finding the images relevant to the user. This result demonstrates how my idea can benefit a number of prior binary attribute search systems [85, 142, 134].

**Relevance feedback with relative attributes** Next we evaluate adapted attributes for relevance feedback. We ask 10 users for whom we have trained user-specific relative attribute models to examine 10 target query images, and tell us whether they exhibit a specified attribute more/less/equally than 20 random reference images. This yields a total of 20 feedback statements per query per user.

Table 5.3 shows the results. Since in this scenario the user describes a single image which is known to us, we gauge accuracy in terms of the percentile

	GENERIC	EXPLICIT	+CONTRADICTIONS	+TRANSITIVITY
Shoes-R	70.96 (0.1)	72.58 (0.1)	<b>74.15</b> (0.1)	<b>74.34</b> (0.1)

Table 5.4: The benefit of inferring implicit user-specific labels.

rank for this target image, i.e., the proportion of database images that the system ranks lower than the correct target image that the user was trying to find (higher is better). Again, the personalized search results are best, even notably stronger than the personalized user-exclusive model. To give a concrete sense of significance, my method ranks the target image seven pages higher than the closest baseline, assuming a webpage fits 40 images per page. This result shows how my idea can improve prior systems for relative attribute search.

**Learning with inferred labels** Finally, I validate my ideas to infer user-specific labels. They apply only to the relative attribute search scenario, so we test on Shoes Relative. We replace half of the explicit user-specific labels used for adaptation with all the labels we infer using transitivity or contradictions. Table 5.4 shows that either inference method boosts search accuracy. The user’s target image is on average ranked six pages higher using those “free” inferred labels compared to just explicit labels. Note we are getting comparable accuracy to my result in Table 5.3, but now with half the user-specific labels.

## 5.5 Conclusions

Existing work assumes that users agree on the attribute values of images, despite evidence that this is not always the case. My main contribution in this chapter is the idea of adapting attributes to account for user-specific perception. The approach I propose accommodates both binary and relative properties, and makes it possible to leverage existing labeled datasets as a prior to regularize new user-specific models. My results on two compelling datasets indicate that (1) people do indeed have varying shades of attribute meaning, (2) transferring generic models makes learning those shades more cost-effective than learning from scratch, and (3) accounting for the differences in user perception is essential in image search applications.

In the next chapter, I show how to exploit commonalities between users in terms of their attribute perceptions, in order to learn models which lie in between the monolithic generic models and the user-specific models with respect to their degree of personalization.

## Chapter 6

# Discovering Shades of Attribute Meaning with the Crowd

In the previous chapter, I described how to personalize an attribute model for a particular user. This is needed because, unlike object categories, attributes are inherently subjective, so users will have different internal models of the attributes. In this chapter, I present an extension of this work which discovers groupings among users with respect to how they use a given attribute term. Learning these groupings allows the discovery of the “shades of meaning” that the attribute term contains.

In addition to enabling more robust personalized models to be learned, discovering attribute shades of meaning also presents an interesting disambiguation problem that lies at the intersection of visual perception and natural language. Word sense disambiguation is a classic disambiguation problem in language alone. It involves, for instance, automatically determining whether the word “bank” refers to a financial bank or a river bank. I discuss polysemy of words in images in Section 2.6. However, in this thesis I study the distinct problem of determining which shade of an adjective a user employs when judging whether a visual property is present or not in a particular image.



I find evidence for this last claim in work on linguistic relativity [37], which examines how language affects perception and how cultural differences influence how people describe objects, shape properties of animals, colors, etc. Colors are the quintessential example: e.g., Russian has two words for what would be shades of “blue” in English, while other languages do not strongly distinguish “blue” and “green”. In other words, if asked whether an object in some image is “blue” or not, people of different countries might be grouped around different answers. For many attributes, such ambiguities in language *use* cannot be resolved by adjusting the attribute definitions, since people *use the same definition differently*.

One additional reason why we cannot avoid ambiguity in attribute terms is that it is prohibitively expensive to define a vocabulary which has the potential to describe any possible object in a given dataset, in such a way that every term in the vocabulary is completely defined with no room for interpretation. On the other hand, if we can discover the shades of a small set of attribute terms, we can expand a seed vocabulary that does not have complete coverage.

Therefore, the next goal of my thesis is to automatically discover the shades of an attribute. As discussed earlier, a “shade” is a visual interpretation of an attribute name that one or more people apply when judging whether that attribute is present in an image. Given a semantic attribute name, I want to discover its multiple visual interpretations and train a discriminative model for each one. Rather than attempt to manually enumerate the possible

shades, I propose to learn them indirectly from the crowd. First I ask many annotators to label various images, reporting whether the attribute is present or not. Using their responses, the method I develop estimates latent factors that represent the annotators in terms of the kinds of visual cues that they associate with the attribute. Then, clustering in the low-dimensional latent space, the method identifies the “schools of thought” (about how to interpret this attribute) underlying the discrete set of labels the annotators provided. Finally, it uses the positive exemplars in each school to train a predictive model, which can then detect when the particular attribute shade is present in novel images.

The resulting models are both semantic and visually precise. By discovering the shades from the crowd’s latent factors, my approach isolates the features corresponding to the perceived shades. This makes the method less susceptible to the more “obvious” splits in the feature space that an image clustering approach (including today’s sophisticated discovery methods [106, 96, 33, 116, 138, 178]) may find, which need not directly support the semantic attribute of interest.

I demonstrate two key applications for shade discovery: attribute prediction (in Section 6.6.2) and attribute-based search (in Section 6.6.3). These results show the utility of my shade discovery approach for interactive search: for a user to reliably find “formal” shoes, the system must correctly estimate “formal” in the database images. If the wrong attribute shade is predicted, the wrong image is retrieved. In general, detecting shades is key whenever

linguistic attributes are required, which includes applications beyond image search as well (e.g., zero-shot recognition).

I first explain the crowdsourced label collection in Section 6.1. Then I describe how to recover the latent factors responsible for those labels (Section 6.2) and use them to discover attribute shades (Section 6.3). Finally, I exploit the discovered shades to improve attribute prediction by accounting for the users’ varying interpretations (Section 6.4).

## 6.1 Collecting Crowd Labels per Attribute

I use two datasets: **Shoes** [11, 78] and **SUN Attributes** [110], as in Chapter 5. I use 2559 and 2086 total images from Shoes and SUN, respectively. While attribute labels are available for both, my method needs to record which annotator labeled which image. Therefore, I run my own crowdsourced label collection.

To focus the study on plausibly “shaded” words, we select 12 attributes that can be defined concisely in language, yet may vary in their visual instantiations<sup>1</sup>. This helps ensure that variance in the annotators’ labels stems from the attribute’s visual sub-meanings, as opposed to external factors like the annotator’s personal taste. The 12 attributes are: “pointy”, “open”, “ornate”, “comfortable”, “formal”, “fashionable”, “brown” (for Shoes); and “clut-

---

<sup>1</sup>If some “less shaded” attribute is considered, fewer (or one) shades may be discovered via automatic model selection, discussed in Section 6.6.1. In this case, shades should do no harm. Note that the incurred annotation cost per attribute is about \$50, so it is not trivial to collect data on all attributes.

Attribute	Dictionary definition
Pointy	having a comparatively sharp point, or having numerous pointed parts
Open	having interspersed gaps, spaces, or intervals
Ornate	made in an intricate shape or decorated with complex patterns
Comfortable	providing physical comfort, ease and relaxation
Formal	designed for wear or use at elaborate ceremonial or social events
Brown	the color of, for example, chocolate and coffee
Fashionable	conforming to the current fashion; stylish; trendy; modern
To clutter	to make disorderly or hard to use by filling or covering with objects
To soothe	to bring comfort, composure, or relief
Open (area)	affording unobstructed passage or view
Modern	characteristic or expressive of recent times or the present; contemporary
Rustic	of, relating to, or typical of country life or country people

Table 6.1: The 12 attribute definitions shown to annotators.

tered”, “soothing”, “open area”, “modern”, “rustic” (for SUN). We sample  $N = 250$  to 1000 images per attribute. To get representative images spanning the dataset, we cluster all images using  $K$ -means, then sample ones near the cluster centers. For “brown”, we sample images with high scores output by a “brown” classifier.

We build a Mechanical Turk interface to gather the labels. Workers are shown definitions of the attributes from a web dictionary (see Table 6.1), but no example images. Thus, they all receive the same linguistic definition, but they are not prompted with any particular *visual* definition. Then, given an image, the worker must provide a binary label, i.e., she must state whether

the image does or does not possess a specified attribute. Additionally, for a random set of 5 images, the worker must explain his label in free-form text, and state which image most has the attribute, and why. These questions both slow the worker down, helping quality control, and also provide valuable ground truth data for evaluation, as I will explain in Section 6.6.4.

My latent factor model (defined next) can accommodate imbalanced and sparse labels. This is good, because in realistic scenarios, labels may not originate from concentrated one-time labeling efforts, but rather as a side product of another task—such as click data in image search. In such a case, the images that one user labels will not entirely overlap with those that another user labels. Furthermore, each user will label few examples. To mimic this scenario, we gather labels in a sparse fashion. Each worker labels 50 randomly chosen images, per attribute. To help ensure self-consistency in the labels, we exclude workers who fail to consistently answer 3 repeated questions sprinkled among the 50. This yields annotations from 195 workers per attribute on average.

While multiple workers may label the same image, I stress their labels are *not* aggregated to create a majority vote “ground truth”. The main premise of this chapter is that attribute names can be visually imprecise and so admit multiple interpretations. The same attribute word can have different meanings to different people, even if they all know the same linguistic definition of the word. (Contrast this with object category names, which are relatively precise.) Thus, rather than discard label discrepancies as noise, we

use them to discover shades.

## 6.2 Recovering Latent Factors for Attribute Labels

Now we use the label data to discover latent factors, which are needed to recover the shades of meaning. Note that we learn factors for each attribute independently, so all variables below are attribute-specific. From the above data collection, we retain each worker’s ID, the indices of images he labeled, and how he labeled them. Let  $M$  denote the number of unique annotators, and let  $N$  denote the number of images seen by at least one annotator. Let  $\mathbf{L}$  be the  $M \times N$  label matrix, where  $L_{ij} \in \{0, 1, ?\}$  is a binary attribute label for image  $j$  by annotator  $i$ . A ? denotes an unlabeled example. The matrix is only partially observed, as on average only 20% of the possible image-worker pairs are labeled.

We suppose there is a small number  $D$  of unobserved factors that influence the annotators’ labels. This reflects that their decisions are driven by some mid-level visual cues. For example, when deciding whether a shoe looks “ornate”, the latent factors might include presence of buckles, amount of patterned textures, material type, color, and heel height; when deciding whether a scene looks “modern”, they might include color, object composition, and materials.

Assuming a linear factor model, the label matrix  $\mathbf{L}$  can be factored as the product of an  $M \times D$  annotator latent factor matrix  $\mathbf{A}^T$  and a  $D \times N$  image latent factor matrix  $\mathbf{I}$ :  $\mathbf{L} = \mathbf{A}^T \mathbf{I}$ . A number of existing methods can

be used to factor this partially observed matrix, by finding the best rank- $D$  approximation under some loss function [131, 132, 175]. We use a probabilistic matrix factorization algorithm (PMF) [131, 132], due to its efficiency for large, sparse matrices. Briefly, it works as follows. PMF takes a probabilistic approach to recover the two low-rank matrices. The likelihood distribution for the observed labels is

$$p(\mathbf{L}|\mathbf{A}, \mathbf{I}, \sigma^2) = \prod_{i=1}^M \prod_{j=1}^N [\mathcal{N}(L_{ij}|A_i^T I_j, \sigma^2)]^{\ell_{ij}}, \quad (6.1)$$

where  $A_i$  and  $I_j$  denote columns of  $\mathbf{A}$  and  $\mathbf{I}$ , respectively, and  $\ell_{ij} = 1$  if we received a label on image  $j$  by annotator  $i$ , and  $\ell_{ij} = 0$  otherwise.  $\mathcal{N}(x|\mu, \sigma^2)$  denotes a Gaussian distribution with mean  $\mu$  and standard deviation  $\sigma^2$ . The priors over the latent factors are spherical Gaussians,  $p(\mathbf{A}|\sigma_A^2) = \prod_{i=1}^M \mathcal{N}(A_i|0, \sigma_A^2 \mathbb{I})$  and  $p(\mathbf{I}|\sigma_I^2) = \prod_{j=1}^N \mathcal{N}(I_j|0, \sigma_I^2 \mathbb{I})$ .

We seek the latent features that maximize the log-posterior:

$$\mathbf{A}^*, \mathbf{I}^* = \arg \max_{\mathbf{A}, \mathbf{I}} \ln p(\mathbf{A}, \mathbf{I}|\mathbf{L}, \sigma^2, \sigma_A^2, \sigma_I^2). \quad (6.2)$$

Obtaining the MAP factors amounts to minimizing an SSD objective function with quadratic regularization terms using gradient descent [131]; this yields a probabilistic extension of what would be standard SVD in the case of fully observed labels. Upgrading to a full Bayesian treatment [132], we put priors on the hyperparameters  $\sigma^2$ ,  $\sigma_A^2$ ,  $\sigma_I^2$  and obtain a predictive distribution for the latent factors, using MCMC to sample the latent feature matrices in parallel. This reduces overfitting and saves parameter tuning. See [132] for details.

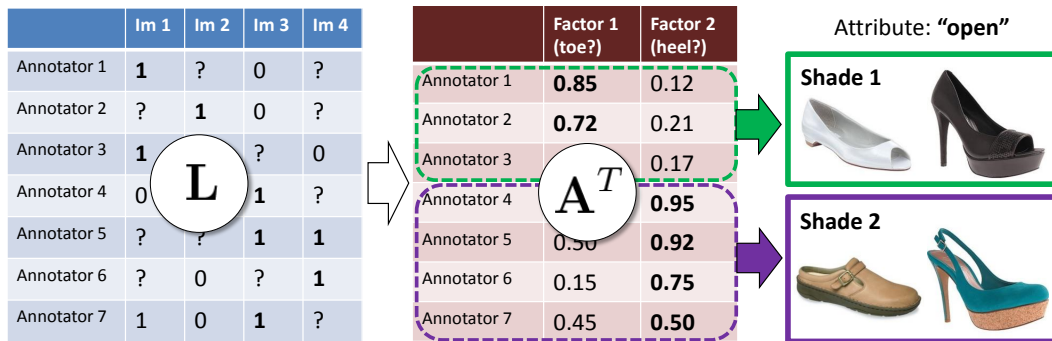


Figure 6.1: Given a partially observed attribute-specific label matrix (left), we recover its latent factors and their influence on each annotator (middle). We discover shades by clustering in this space (right).

### 6.3 Discovering Shades of Meaning

In collaborative filtering, the goal of such a factorization is to impute missing labels (e.g., to predict how a user will rate an unseen movie,  $L_{ij} \approx \langle A_i, I_j \rangle$ ). While missing labels could similarly be estimated for our data, our goal is different. We aim to discover attribute shades of interpretation and generate predictive visual models for them.

To this end, we first represent each annotator in terms of his association with each discovered factor. The “latent feature vector” for annotator  $i$  is  $A_i \in \mathbb{R}^D$ , the  $i$ -th column of  $\mathbf{A}$ . It represents how much each of the  $D$  factors influences that annotator when he decides if the named attribute is present. Likewise, the latent feature for image  $j$  is  $I_j \in \mathbb{R}^D$ , the  $j$ -th column of  $\mathbf{I}$ , and represents how much each of the  $D$  factors is visible in the image.

Figure 6.1 illustrates with a cartoon example. As seen on the left, annotators did not label all images for the attribute “open”. Some tended to



label images 1 and 2 as having the attribute, whereas others tended to label 3 and 4 as positive. After factoring the label matrix, suppose we discover  $D = 2$  latent factors. Though nameless, they align with semantic visual cues; suppose here they are “toe is open” and “heel is open”. Each annotator’s feature  $A_i$  encodes how important those two factors were for his label decision. In this hypothetical example, we see the first three annotators labeled images 1 and 2 as open due to factor 1, whereas the others focused on factor 2 in other images.

We pose shade discovery as a grouping problem in the space of these latent features.<sup>2</sup> While other clustering algorithms could be used, we apply  $K$ -means to the columns of  $\mathbf{A}$  to obtain clusters  $\{\mathcal{S}_1, \dots, \mathcal{S}_K\}$ .<sup>3</sup> Each cluster is a shade. Annotators in the same cluster display similar labeling behavior, meaning they interpret similar combinations of mid-level visual cues as salient for the attribute at hand. For example, in Figure 6.1, the two dominant shades reflect which part of the shoe the annotator focused on to judge openness—toe or heel. (Of course, for real data, there will be  $D > 2$  factors, and shades will combine many such factors.)

Recall that shade discovery is done on a per-attribute basis. Depending on the visual precision of the word, some attributes may have only one shade; others may have many. To automatically select  $K$  based on the structure

---

<sup>2</sup>While we can cluster either annotators or images to identify shades, we choose annotators in order to facilitate ground truth evaluation in Section 6.6.

<sup>3</sup>Preliminary tests with Bayesian non-parametric clustering showed inferior results. An alternative would be to impute missing labels and group with EM, but clustering in the compact latent space is preferable when labels are very sparse.

of the data, we use the silhouette coefficient [121]. It quantifies how tightly grouped all the latent features in a cluster are, averaged across all clusters, in terms of the average distance of an instance to all other data in the same cluster versus its distance to a neighboring cluster.

## 6.4 Using Shades to Predict Perceived Attributes

A key valuable application of shades is to improve attribute prediction accuracy, generalizing what the system discovered to novel images. Prior work uses one of two extremes for attribute prediction—either (1) a *consensus* classifier: a single model trained with majority vote labeled examples (e.g., [85, 87, 40, 156, 110]), or (2) a *user-specific* classifier trained by adapting that majority vote model to satisfy an individual user’s training labels, as in Chapter 5.

I now propose an approach in between these two extremes. With shades, we can account for the fact that people perceive an attribute differently, yet avoid specializing predictions down to the level of each individual user. The idea is to tailor an attribute classifier according to the user’s “school of thought”, i.e., the shade to which he subscribes.

To this end, we train shade-specific classifiers that adapt the consensus model. Each shade  $\mathcal{S}_k$  is represented by the total pool of images that its annotators labeled as positive. Several annotators in the cluster may have labeled the same image, and their labels need not agree. Thus, we perform majority vote (over just the annotators in  $\mathcal{S}_k$ ) to decide whether an image is

positive or negative for the shade. This majority vote can be seen as a form of quality control, where we assume consistency within the group. For both my shade models and the consensus model, we discard labels where fewer than 90% of users agree. We use the images to train a discriminative classifier, using the adaptive SVM objective of Yang et al. [177] to regularize its parameters to be similar to those of the consensus model, as in Chapter 5. In other words, we are now personalizing to schools of users, as opposed to individual users. Then we apply the adapted shade model for the cluster to which a user belongs to predict the presence/absence of the attribute in novel images. Thus, the predictions are automatically tailored to that user’s perception of the property.

To recap, shades offer an important midpoint on the spectrum discussed above. Compared to the standard consensus approach, we account for distinct perceived shades. Compared to user-adaptive models, the advantages are twofold. First, each model typically leverages more training data than a single user provides. This lets us effectively “borrow” labeled instances from the user’s neighbors in the crowd. Second, we leverage the robustness of the intra-shade majority vote. This helps reduce noise in an individual user’s labeling. The results in Section 6.6.2 reveal the impact of these advantages in practice.

Note, a user must provide at least some attribute labels to benefit from the shade models, since we need to know which shade to apply. For users who contributed to the label matrix  $\mathbf{L}$  this is straightforward. For users adding labels later, we could either re-factor  $\mathbf{L}$ , or use a folding-in heuristic [60] (not

attempted in my experiments).

## 6.5 Discussion

The key thing to note about the shade classifiers is how their positive labeled exemplars came about. Images within a shade can be visually diverse from the point of view of typical global image descriptors, since annotators attuned to that shade’s latent factors could have focused on arbitrarily small parts of the images, or arbitrary subsets of feature modalities (color, shape, texture). For example, one shade for “open” might focus on shoe toes, while another focuses on shoe heels. Similarly, one shade for “formal” capturing the notion that dark-colored shoes are formal would rely on color alone, while another capturing the notion that shoes with excessively high heels are not formal would rely on shape alone. An approach that attempts to discover shades based on image clustering—or non-semantic attribute discovery [106, 96, 33, 116, 138, 178]—will be hard pressed to group images according to these perceived, possibly subtle, cues. My insight is to leverage patterns among the crowd labels to partition the images *semantically*. Then, even though the training images may be visually diverse, standard discriminative learning methods let us isolate the informative features. Essentially, we avoid biasing the shades to a particular low-level descriptor space, since their training images are determined independent of the descriptors.

One might wonder: why not just manually enumerate the attribute shades with words? My approach has multiple advantages over that strategy,

beyond being automatic. For polysemous *nouns*, the visual definitions are enumerable—check the dictionary. In contrast, it can be difficult to put an attribute’s distinct visual instantiations in words. Furthermore, the words annotators typically provide to explain their labels are concrete instances of the shade, which need not comprehensively define the shade. For example, in my data collection, when asked to explain why an image is “ornamented”, an annotator might comment on the “buckle” or “bow”; yet the latent shade of “ornamented” underlying many users’ labels is more abstract. It encompasses combinations of such concrete mid-level cues. In short, I find that people are good at naming examples, but less good at characterizing an entire shade in words. My method fills that gap, using structure in the labels to identify shades.

## 6.6 Experimental Validation

I first demonstrate shades’ key utility for improving attribute prediction (Section 6.6.2) and attribute-based search (Section 6.6.3). I then quantitatively analyze the purity of the discovered shades (Section 6.6.4). I offer comparisons to existing techniques, including both standard consensus attributes as well as state-of-the-art methods for attribute discovery [116] and personalized attributes ([76] and Chapter 5). I analyze the shades qualitatively (Section 6.6.5) to visualize what is discovered. Finally, I show how to transfer shades between attributes and users in order to predict how a user will interpret an attribute for which he has provided no labels (Section 6.6.6).

### 6.6.1 Experimental Design

We use provided image descriptors for all methods: concatenated GIST and color histograms for Shoes, and GIST, color, HOG, and self-similarity histograms for SUN. We use the Bayesian PMF implementation of [175]. We fix  $D = 50$ , then use the default parameter settings. For  $N = 1000$  and  $M = 195$ , MCMC with 500 samples takes about 21 minutes. We cross-validate all classifier parameters. We set  $K$  automatically per attribute based on the optimal silhouette coefficient within  $K = \{2, \dots, 15\}$ . Typically values of  $K \approx 7$  are chosen by the algorithm.

As noted in Section 6.1, during data collection annotators must explain their attribute labels. Specifically, we ask, “Please explain your response. What part or aspect of the image do you associate with the attribute [attribute name]? What part or aspect of the image led you to say that the attribute [attribute name] is present or not present?” Table 6.2 shows a sample of their responses. We draw on their explanations below to aid our quantitative evaluation, but they are never seen by the method.

### 6.6.2 Accuracy of Perceived Shade Predictions

We begin with a key result demonstrating how well shades capture perceived attributes. We apply the shades as described in Section 6.4 to predict user-specific labels. We compare to three methods: (1) `GENERIC`, which is the standard consensus approach [45, 85, 87, 40, 156, 17, 170, 110], (2) `USER-EXCLUSIVE`, which trains one attribute classifier per user using only his labeled







Image	Attribute	Present?	Explanation
	Ornate	No	"Ornate means decorated with extra items not inherent in the making of the object. This boot has a <b>camo print</b> as part of the object, but <b>no additional items</b> put on it."
	Ornate	Yes	"The <b>flowerprint pattern is unorthodox</b> for a rubber boot and really <b>stands out</b> against the jet black background."
	Open area	Yes	"This is an <b>enclosed area, but the room is very large</b> and the ceiling is very high, giving a lot of room. I think that this makes it an enclosed area that is also an open area. "
	Open area	No	"I <b>do not consider the image to show an open area because the area shown is enclosed by walls</b> . It is a larger space on the interior of the building so it does have some aspects of an open space."
	Comfortable	Yes	"The <b>heel is shorter and looks more sturdy</b> with the thickness of the heel which would make it more comfortable than your typical heel."
	Formal	Yes	"I believe the <b>formal aspect of this should be the color and design</b> of the the fabrics on this shoe. I felt this shoe would be used by a person who wanted to be formal yet comfortable. "

Figure 6.2: Example label explanations that annotators provided. Bold is my emphasis. In the first two rows, notice that the same type of shoe (one with patterns) can be perceived to have a different level of ornamentation, depending on whether the annotator believes patterns constitute ornamentation. Further, a room with large spaces (rows 3 and 4) can be perceived as an open area or not, depending on whether the annotator believes an area enclosed by walls can be considered open. Finally, in the last two rows we see two interesting examples of a high-heeled shoe (which is normally labeled as uncomfortable) considered comfortable due to its sturdy heel, and a sneaker-like shoe seen as formal due to its color and design. Also notice how well-thought out these user responses are, which indicates that the quality of data we collected is high.

Attribute	SHADES	GENERIC	USER-EXC	USER-ADP	ATTR DISC	IMG CLUST
Pointy	<b>76.3</b> (0.3)	74.0 (0.4)	67.8 (0.2)	74.8 (0.3)	74.5 (0.4)	74.3 (0.4)
Open	<b>74.6</b> (0.4)	66.5 (0.5)	65.8 (0.2)	71.6 (0.3)	68.5 (0.4)	68.3 (0.4)
Ornate	<b>62.8</b> (0.7)	56.4 (1.1)	59.6 (0.5)	61.1 (0.6)	58.3 (0.8)	58.6 (0.7)
Comfort.	<b>77.3</b> (0.6)	75.0 (0.7)	68.7 (0.5)	75.5 (0.6)	76.0 (0.7)	75.4 (0.6)
Formal	<b>78.8</b> (0.5)	76.2 (0.7)	69.6 (0.4)	77.1 (0.4)	77.4 (0.6)	77.0 (0.6)
Brown	<b>70.9</b> (1.0)	69.5 (1.2)	61.9 (0.5)	68.5 (0.9)	69.3 (1.2)	69.8 (1.2)
Fashion.	<b>62.2</b> (0.9)	58.5 (1.4)	60.5 (1.3)	62.0 (1.4)	61.2 (1.4)	61.5 (1.1)
Cluttered	<b>64.5</b> (0.3)	60.5 (0.5)	58.8 (0.2)	63.1 (0.4)	60.4 (0.7)	60.8 (0.7)
Soothing	<b>62.5</b> (0.4)	61.0 (0.5)	55.2 (0.2)	61.5 (0.4)	61.1 (0.4)	61.0 (0.5)
Open area	<b>64.6</b> (0.6)	62.9 (1.0)	57.9 (0.4)	63.5 (0.5)	63.5 (0.8)	62.8 (0.9)
Modern	<b>57.3</b> (0.8)	51.2 (0.9)	56.2 (0.7)	56.2 (1.1)	52.5 (0.9)	52.0 (1.1)
Rustic	<b>67.4</b> (0.6)	66.7 (0.5)	63.4 (0.5)	67.0 (0.5)	67.2 (0.5)	67.2 (0.5)

Table 6.2: Accuracy of predicting perceived attributes, with standard error in parentheses.

images, and (3) USER-ADAPTIVE, the transfer method from Chapter 5, which adapts the majority vote model with the same user-specific labeled data. All methods use linear SVMs. My method selects  $K$  automatically per attribute, yielding values between 5 and 10. We run 30 trials, sampling 20% of the available labels to obtain on average 10 labels per user (representing what a user might reasonably contribute to train the system). Note that the data is well-balanced, with positive/negative test data ratio between 30/70 and 49/51 (42/58 on average).

Table 6.2 shows the results. My shade discovery method outperforms all other methods. It is more reliable than GENERIC, which is the status quo attribute learning approach. For “open”, we achieve an 8 point gain over GENERIC and USER-EXCLUSIVE, which indicates both how different user perceptions of this attribute are, as well as how useful it is to rely on schools rather than individual users. SHADES also outperform my USER-ADAPTIVE



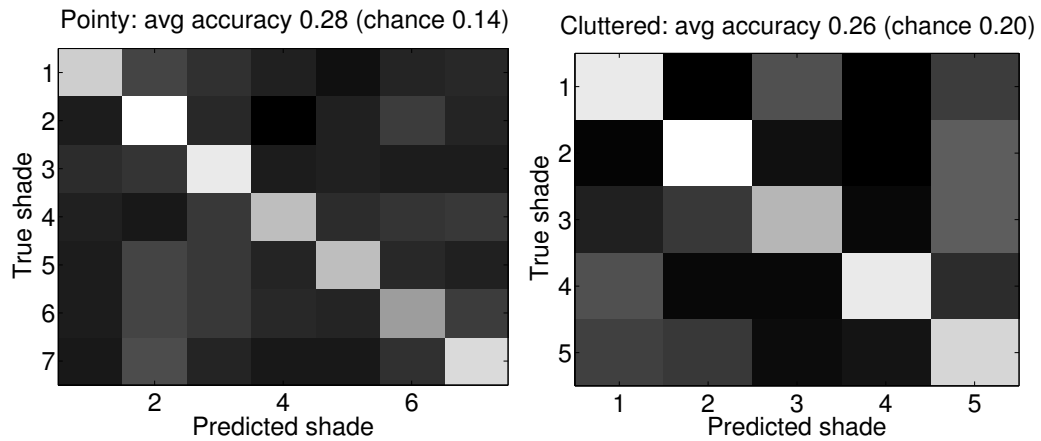


Figure 6.3: Accuracy of perceived shade predictions: Confusion matrices for multi-way shade classification, for the attributes “pointy” and “cluttered”.

approach of Chapter 5, while requiring the exact same labeling effort. While that method learns personalized models, shades leverage *common perceptions* and thereby avoid overfitting to a user’s few labeled instances. I also test models for alternative shade formation baselines—ATTRIBUTE DISCOVERY and IMAGE CLUSTERS—which I define and use below in Section 6.6.4. Neither of these methods is competitive with my approach.

While Table 6.2 measures binary attribute classification, my method can also perform multi-way shade classification. Here we cluster in the latent feature space of the images  $I_j$ , and again automatically select  $K$ . Figure 6.3 shows representative resulting confusion matrices for the attributes “pointy” and “cluttered”. Shades’ average multi-way accuracy over all attributes is 0.28, much better than chance (0.15 on average). This result indicates the discovered shades per attribute are indeed distinct and detectable.

These results clearly demonstrate the utility of shades. For all attributes, mapping a person’s use of an attribute to a shade allows us to *predict attribute presence more accurately*. This is achieved at no additional expense for each user. As a result, applications demanding descriptive attributes (e.g., image search, zero-shot learning, etc.) benefit from the more accurate representation.

### 6.6.3 Improved Personalized Search with Shades

In Section 5.4.4 in the previous chapter, I showed that user-adapted attributes lead to more successful results for multi-attribute queries. In this section, I demonstrate that our discovered attribute shades improve search results even further.

First, we collect additional data for the Shoes attributes in Table 6.1, such that the same images are labeled for all attributes, and all users label all attributes.<sup>4</sup> We ask each user to label 40 images for each attribute, out of a total set of 200 images that receive labels from any user. We use 50 images total for training, 75 for testing, and 75 for cross-validation. We repeat the shade formation and shade-based attribute prediction procedure as in Section 6.4, using the training data from each user.

We then pose multi-attribute queries with the test images. For each test image and user, we generate all  $q$ -tuples of the attributes with labels

---

<sup>4</sup>We omit the attribute “brown” since it only appears in a small set of images.

$q$	SHADES	GENERIC	USER-EXCLUSIVE	USER-ADAPTIVE
2	<b>0.533</b> (0.001)	0.501 (0.001)	0.433 (0.001)	0.509 (0.001)
3	<b>0.398</b> (0.001)	0.363 (0.001)	0.294 (0.001)	0.374 (0.001)
4	<b>0.297</b> (0.002)	0.265 (0.002)	0.201 (0.002)	0.279 (0.002)
5	<b>0.218</b> (0.005)	0.188 (0.005)	0.140 (0.004)	0.207 (0.005)
6	<b>0.171</b> (0.018)	0.129 (0.016)	0.117 (0.016)	0.164 (0.018)

Table 6.3: Multi-attribute query image search accuracy using shades, with standard error in parentheses.

from the user. Each of these tuples forms a multi-attribute query composed of  $q$  attributes that a user might issue during search, e.g., “I want to buy *ornate, formal* shoes.” We use the user’s labels as the ground truth for these queries, and examine the presence/absence predictions of the GENERIC, USER-EXCLUSIVE, USER-ADAPTIVE, and SHADES approaches on each  $q$ -attribute query. As in Section 5.4.4, we measure the fraction of these query images where the user’s ground truth labels and a model’s predictions agree on all  $q$  attributes per query.

Table 6.3 shows the results, for  $q = \{2, \dots, 6\}$ . My shades approach produces higher match rates, hence more accurate results, than any of the baselines, consistently with my result in Section 6.6.2. For  $q = 2$ , shades achieve a 6% relative gain over GENERIC, and 5% gain over USER-ADAPTIVE. This demonstrates that in order for attribute-based searches to be successful, the retrieval system needs to interpret the user’s attribute queries correctly; shades allow the learning of robust models which are personalized yet do not overfit to noise in a user’s labels. Note that chance performance corresponds

to the probability of randomly matching all  $q$  attribute ground truth labels. Hence, chance is 0.250, 0.125, 0.062, 0.031, and 0.015, respectively. All methods show a decrease in accuracy as more query words are used, since it becomes more difficult for a method to correctly predict the presence of *all* increasingly many attributes.

#### 6.6.4 Quantifying the Accuracy of Shade Formation

To further quantify how accurately our shades capture perceived interpretations, we next score how *coherent* the textual explanations (cf. Table 6.2) are among annotators in the same shade. Whereas random clusters would group diverse ground truth explanations together, good shades should align with coherent explanations. I stress that these explanations are never seen by my algorithm; they are for evaluation purposes only.

To measure coherency, we first perform probabilistic Latent Semantic Analysis (pLSA) [60] on the Porter-stemmed textual descriptions.<sup>5</sup> We treat each description for which  $L_{ij} = 1$  as a document and discover  $T = 200$  topics with pLSA. Then we map each explanation to its distribution of topics (a vector of  $T$  weights). This representation accounts for word meaning, not just word occurrences (e.g., “image” and “picture” will be treated as synonyms by pLSA). Next, we average the topics for all positive descriptions originating from annotators in a given shade, yielding one topic vector per shade. Finally,

---

<sup>5</sup>Note that one can use any topic modeling approach, but I choose pLSA due to its ability to account for synonyms (compared to a bag-of-words model), to produce proper topic distributions (compared to LSA), and due to its simplicity (compared to LDA).

we score the quality of a shade by its topic distribution entropy. Low entropy is better, as it indicates the shade corresponds to a more coherent set of descriptions. Topic entropy was also used for text analysis by Hall et al. [58].

We compare SHADES to two methods: (1) `ATTRIBUTE DISCOVERY`: a state-of-the-art non-semantic attribute discovery method [116], and (2) `IMAGE CLUSTERS`: an image clustering approach inspired by [92]. The first baseline discovers splits in the feature space that are discriminative for object categories. We use the code kindly provided by the authors; we train it with the 10 Shoe and 611 SUN categories in the training images used by my method. We then cluster images in the discovered attribute space. (I also tried using [116] with the semantic attributes as “categories”, but it performed significantly worse.) The second baseline is inspired by prior work for discovering word “senses” [92]; it clusters the image descriptors for all images labeled positive by at least one annotator. For both, to map an image cluster to ground truth descriptions, we look at the bag of images each annotator labeled as positive, find the image cluster to which the largest portion of the bag belongs, and assign it to be this user’s shade ID. All methods use  $K$ -means and remove clusters with fewer than 10 members, which tend to be too sparse to form a meaningful shade.

Figure 6.4 shows the results. We plot topic entropy (and standard error) as a function of the number of shades  $K$ , over all attributes and 30 runs. Shades are much more coherent overall, while image clustering falls short. The non-semantic attribute discovery method [116], while stronger

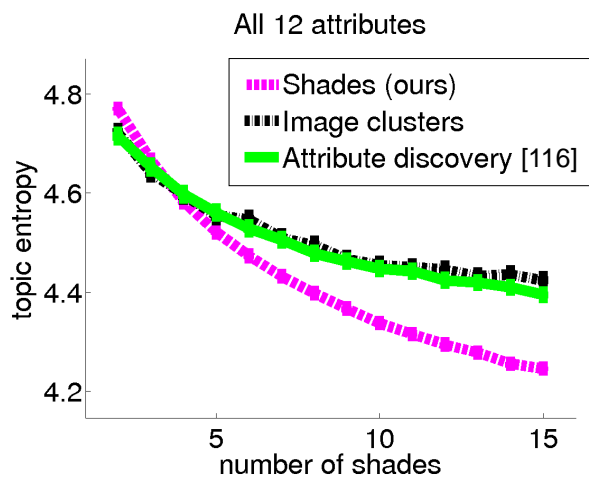


Figure 6.4: Quality of discovered attribute shades (low entropy is good).

than clustering, does not capture the shades of meaning since it lacks human input on the attribute interpretation. When  $K = 2$ , the baselines have lower entropy than our shades, showing that very coarse groups are sufficiently found with image clustering; however, these clusters are too coarse according to the silhouette coefficient model selection, which selects  $K = 5$  to  $K = 10$  shades as the optimal setting. This shows the shades we have discovered are meaningful and accurately capture the varied attribute meanings that users employ.

I now give some more information to help gauge the significance of these results. My method achieves entropy which is about 0.2 lower than the entropy of the baseline methods. In Table 6.4, I show some pairs of individual descriptions which have about 0.2 difference in their topic distribution entropies. Note that Figure 6.4 captures entropies of distributions over a number of descriptions, which are naturally higher than the topic entropy of a single description.

Again, lower entropy denotes a more focused explanation. In Table 6.4, the first explanation for “open” includes many unrelated details, while the second predominantly discusses the foot being seen. Similarly, a high-quality user cluster will correspond to documents that focus on a single or a few topics. The second explanation for “ornate” focuses on color, hence achieves lower entropy. The second explanation for “open area” focuses on the words “room” and “space”. Just like the second document in each pair, the clusters that my method obtains are more focused.

### 6.6.5 Visualizing Attribute Shades of Meaning

Next, I provide qualitative results. Figure 6.5 visualizes two shades each, for nine of the attributes. The images are those most frequently labeled as positive by annotators in a shade  $S_k$ . The (stemmed) words are those that appear most frequently in the annotator explanations (cf. Table 6.2) for that shade, after we remove words that overlap between the two. Font size reflects relative frequency. To aid readability, I also outline words that stand out as good representatives of the shade.

We see the shades capture nuanced visual sub-definitions of the attribute words. For example, for the attribute “brown”, one shade covers chocolate-colored shoes (top shade), while another is lighter and more gold (bottom shade). For “ornate”, one shade focuses on straps/buckles (top), while another focuses on texture/print/patterns (bottom). For “comfortable”, one shade emphasizes a low arch (top), while the other requires soft materials

Attribute	Entropies	Explanations
Open	2.85	“This shoe is open across the top of the foot, with a space between the ankle strap and the toe. It also has gaps along the sides of the toe.”
	2.62	“Open represents that amount of foot that can be <b>seen</b> when the show is worn. The opening on this shoe allows for a portion of the upper foot to be <b>seen</b> .”
Ornate	2.45	“I consider the shoe in Image 45 to be ornate (made in an intricate shape or decorated with complex patterns) because it is oddly shaped, with a pattern and added strapping and it has a zipper pull that stands out.”
	2.27	“I associate the pattern of the shoe with the attribute ornate. It is the way that the plaid is mixed in, its <b>color</b> , and the mixing of the <b>color</b> in the shoe laces as well that led me to say that the attribute ornate is present.”
Open area	2.41	“You can see the sky and even though the photo is of a building there is plenty of open space surrounding it as well as the photography being taken outside.”
	2.23	“Inside the net there is plenty of <b>space</b> , and <b>room</b> between the nets. There’s not too much <b>room</b> , but enough to be considered an open area. It’s also outside so out of the nets is plenty of <b>room</b> .”

Table 6.4: Pairs of annotation explanations with corresponding topic entropy. Lower entropy corresponds to more focused description (second example in each attribute). Bolded words are my emphasis (see text).





Figure 6.5: Top words and images for two shades per attribute (top and bottom for each attribute). Best viewed on PDF or in color. See text for description.

(bottom). For “open”, one shade includes open-heeled shoes, while another includes sandals which are open at the front *and* back. In SUN, the “open areas” attribute can be either outside (top) or inside (bottom). For “soothing”, one shade emphasizes scenes conducive to relaxing activities, while another focuses on aesthetics of the scene.

As discussed above, an important feature of my method is its ability to perform discovery independent of a particular image descriptor. To illustrate this, we next use the shades’ visual classifiers to examine their most informative *localized* features. We use  $L_1$  regularization when training one-vs.-rest logistic regression classifiers for each shade, in order to isolate a sparse set of features most discriminative for that shade. For each  $70 \times 70$  grid cell of the image, we sum the magnitude of the classifier weights for its features. Then we multiply those weights with the pixel intensities in order to visualize the relative impact of each portion of the image.

Figure 6.6 shows example results. Brighter cells indicate regions more discriminative for that shade. For “open”, we see one shade emphasizes openness at the back, and another openness at the toe. For “formal”, the top shade emphasizes the arch of the shoe, while the bottom one emphasizes the toes. Such examples illustrate how my method isolates visual properties that support a shade, yet would not be tightly grouped if simply clustering global descriptors.

Of course, learning discriminative spatially localized features is nothing new; the point is that shades are what enable the training image groups

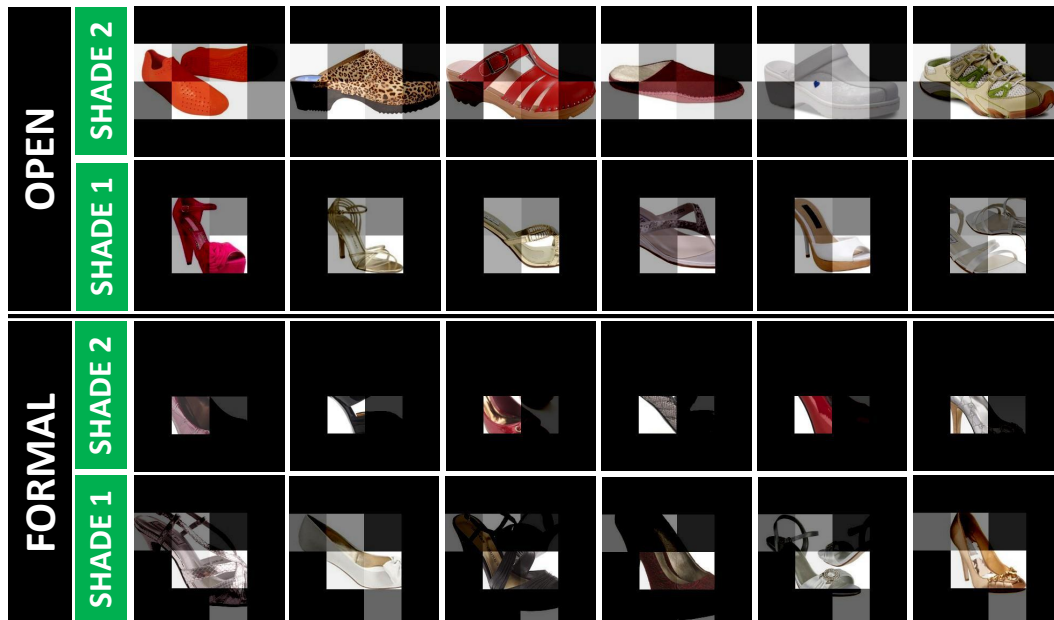


Figure 6.6: Image regions highlighted according to the importance of the localized features for learning the shades.

that make this discriminative selection feasible. Furthermore, recent work using crowds to isolate informative spatial regions [30, 27] has a different purpose (fine-grained image classification) and takes an entirely different approach (explicitly asking labelers to outline the regions needed to make their label decisions).

### 6.6.6 Exploiting Attribute Correlations for Transfer

So far, we have discovered the shades of each attribute disjointly from other attributes. However, the attributes that we use are not completely independent. For example, there is notable correlation between the attributes “fashionable” and “formal”. I propose to exploit these correlations to predict

how a user will perceive an attribute for which he has not supplied any labeled examples, by transferring labels for this attribute from other users, and from other attributes labeled by the same user.

As mentioned in Section 6.3, matrix factorization can also be used to “fill in” missing values in the (user, image) label space. The value of an entry  $L_{ij}$  can be computed as an inner product of the user  $A_i$ ’s and image  $I_j$ ’s latent factor vectors.

However, this label imputation can also exploit *multiple* (user, image) label matrices together, if we stack these matrices in a tensor. In this case, the label matrix  $\mathbf{L}$  becomes an  $M \times N \times Z$  label *tensor*, where  $Z$  denotes the number of attributes being considered at once. We can decompose  $\mathbf{L}$  as:

$$\mathbf{L} = \sum_{d=1}^D \mathbf{A}_{d,:} \circ \mathbf{I}_{d,:} \circ \mathbf{T}_{d,:}, \quad (6.3)$$

where the index  $d, :$  refers to the rows of the matrices and  $\circ$  refers to outer vector product.  $\mathbf{T}$  is the  $D \times Z$  matrix of latent factors for each of the  $Z$  attributes. We use the Bayesian tensor factorization of [175] for this formulation, which essentially extends the probabilistic matrix factorization approach of Salakhutdinov and Mnih discussed above to handle tensor data.

An entry  $L_{ijz}$  denotes how user  $i$  labeled image  $j$  for attribute  $z$ . Equation 6.1 then becomes

$$p(\mathbf{L}|\mathbf{A}, \mathbf{I}, \mathbf{T}, \sigma^2) = \prod_{i=1}^M \prod_{j=1}^N \prod_{z=1}^Z [\mathcal{N}(L_{ijz}|\langle A_i, I_j, T_z \rangle, \sigma^2)]^{\ell_{ijz}}, \quad (6.4)$$

Dataset	Tensor label imputation	Chance
Shoes	0.831 (0.001)	0.50
SUN	0.770 (0.001)	0.50

Table 6.5: Accuracy of imputing missing labels using other attributes, with standard error in parentheses. Utilizing attribute correlations allows us to accurately predict how a user will perceive a novel attribute, without having received any annotations for this attribute from this user.

where  $A_i$  and  $I_j$  denote columns of  $\mathbf{A}$  and  $\mathbf{I}$  as before,  $T_z$  denotes a column of  $\mathbf{T}$ , and we model the prior over the latent factors in  $\mathbf{T}$  as a spherical Gaussian, similar to  $\mathbf{A}$  and  $\mathbf{I}$ . See the Bayesian Probabilistic Tensor Factorization (BPTF) approach of [175] for more details.

Using this tensor label imputation approach, we can complete a transfer learning task of predicting how a user who has never labeled an attribute  $z$  will perceive this attribute, by relying on this user having labeled other attributes, and other users having labeled attribute  $z$ .

Table 6.5 shows the results. For Shoes, we use the new data collected in Section 6.6.3 as it ensures all users have labeled all attributes, while for SUN we lack such data and use the data collected in Section 6.1. We achieve a much higher accuracy than chance performance at 50%, thus showing that one can successfully transfer knowledge about one attribute to another.

As an extension, one can study how such transfer might help learn shade models more efficiently if we were to further exploit semantic relationships between the attributes.

## 6.7 Conclusions

The work in this chapter of my thesis addresses the gap between how people *describe* attributes and how they *perceive* them visually. I show how to discover people’s shared biases in perception, then exploit them with visual classifiers that can generalize to new images. The proposed approach to discover attribute shades brings together language, crowdsourcing, human perception, and visual representations in a new way.

The learned shades successfully tailor attribute predictions to cater to a user’s “school of thought”, boosting the accuracy of detecting perceived attributes. In systematic experiments, I quantify the impact of shades, both compared to standard paradigms and multiple state-of-the-art methods. The visualized shades show great promise to separate the (sub-)attributes involved in a person’s use of an attribute vocabulary during image search or organization of image content.

In the next chapter, I discuss another application for attributes for search, namely as latent variables in a model for learning object categories (e.g., tags for search), using active selection over both object and attribute label requests.

## Chapter 7

# Actively Selecting Annotations Among Objects and Attributes

In the previous chapters, I described how the unique properties of attributes make them a very effective handle for providing interactive feedback during search. Then I described how to ensure that the attribute terms that a user employs align with the attribute models that the system has learned. I now discuss one final advantage of using attributes for image search.

Image retrieval is traditionally started off by providing some query composed of one or multiple words. Quite frequently these words involve an object category, but the number of categories that a user might type is so vast that it is infeasible to collect large volumes of training data for each category. Therefore, in this chapter, I study techniques to select the training data used for learning object categories actively, so that the human effort in providing labeled data is minimized.<sup>1</sup>

---

<sup>1</sup>This work was published in the Proceedings of the International Conference on Computer Vision (ICCV) 2011 with the title “Actively Selecting Annotations Among Objects and Attributes” and authors Adriana Kovashka, Sudheendra Vijayanarasimhan, and Kristen Grauman. I wrote the code and conducted the experiments, while Sudheendra Vijayanarasimhan helped with some technical details, and all authors contributed to developing the algorithm, devising the experiments, and writing the paper.

Many state-of-the-art object recognition systems integrate robust visual descriptors with a supervised learning algorithm. This basic framework entails having people “teach” the machine learner about objects through labeled examples, which makes the data collection process itself of critical importance. As such, recent research explores interesting issues in gathering large datasets of web images [125, 145, 45, 26], mining external knowledge sources [120, 11, 17], creating benchmark challenges [38], and developing new methods to reduce the expense of manual annotations. Active learning methods in particular are a promising way to focus human effort, as the system can request labels only for those instances that appear most informative based on its current category models [112, 157, 158, 66, 62, 143].

Unfortunately, most existing active learning techniques assume that the labels of interest are the object category names, yet recent work shows the need to move “beyond labels” to even richer annotations such as descriptive *attributes* or relationships between objects [87, 84, 35, 170, 45, 57]. Furthermore, real-world applications of object recognition demand scaling to a very large number of categories, which at the surface suggests that the number of labels needed must grow proportionally with the number of total classes considered—even if one plans to collect labels with active learning.

I propose an active learning approach to address these issues. Whereas my work in Chapter 4 deals with actively eliciting a useful user feedback statement for image search, this chapter deals with actively training an object category model with the help of attributes. The main idea of this work is to



actively select image annotation requests among *both* object category labels as well as the objects' shared attributes, so as to acquire the labels expected to most reduce total uncertainty for multi-class object predictions. This means, for example, that during one active learning loop a person may be asked to name an object, whereas in the next he may be asked to state whether a particular attribute is present. The goal is to select those pairs of images and labeling questions that will be most useful given the current models.

By simultaneously weighing requests in both label spaces, I expect the learner to more efficiently refine its object models. This is because knowledge of an attribute's presence in an image can immediately influence many object models, since attributes are by definition shared across subsets of the object categories. At the same time, attributes' presence or absence in an image is often correlated (e.g., if something "has skin" it is unlikely to be "metallic" as well), suggesting that many images do not require a full annotation of all attributes. Please refer to Figure 1.5 in Chapter 1.

A novel aspect of the approach I propose is that it both weighs different annotation requests and also models dependencies within multi-label instances. Only limited prior work explores either one or the other aspect [157, 143, 112], and in a different context than the setting here. Furthermore, in contrast to prior active learning work, my approach exploits dependencies between the target label space and a latent but human-describable label space, and is the first to learn objects and attributes actively in concert. This can also be viewed as a new way to efficiently supervise joint multi-class training, in that

the actively selected attribute labeling questions are directly tied to properties shared across classes.

To implement the proposed idea, I adopt a discriminative latent model [170] that captures object-attribute and attribute-attribute relationships, and then define a suitable entropy reduction selection criterion to predict the influence a new label of either type will have throughout those connections. This criterion estimates the expected entropy change on all labeled and unlabeled examples, should the label under consideration be obtained. I adapt the existing classifier to extract the necessary posterior probability estimates, and show how to handle partially labeled examples (i.e., those with only some attributes known) such that they can have immediate influence on the active selection.

The proposed approach requires two main elements: a unified classification model to capture object and attribute relationships, and a way to weigh candidate requests for either label space. I first briefly describe the classifier (Section 7.1), and then explain how to actively improve it with an entropy-based selection criterion (Section 7.2).

## 7.1 Object-Attribute Model

In order to predict the impact of potential object and attribute label requests, we need a classifier that accounts for all four relationships portrayed in Figure 1.5. To this end, we directly adopt the discriminative model recently proposed by Wang and Mori [170]. I briefly summarize the necessary background in this section; see [170] for details.

The model is a multi-class *object classifier* that uses binary attributes as hidden variables. The relationships between the object categories and the attributes are learned parameters in the model. Relationships between attributes are represented in a tree-structured graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  whose vertices denote the  $K$  attributes and whose edges are restricted to pairs of attributes  $(j, k) \in \mathcal{E}$  that have the highest mutual information<sup>2</sup>; parameters reflecting the importance of those dependencies for distinguishing objects are then incorporated into the main classifier.

A fully labeled training example consists of an image  $\mathbf{x} \in \mathcal{X}$ , its object label  $y \in \mathcal{Y}$ , and an indicator vector of  $K$  attribute labels  $\mathbf{h} = [h_1, \dots, h_K]$ , with all  $h_i \in \mathcal{A}$ . We use binary-valued attributes, and so  $\mathcal{A} = \{0, 1\}$ . The classifier  $f_{\mathbf{w}} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathfrak{R}$  is parameterized by vector  $\mathbf{w}$ , and will be defined below. At test time, one predicts the object label  $y^*$  for image  $\mathbf{x}$  as:

$$y^* = \arg \max_{y \in \mathcal{Y}} f_{\mathbf{w}}(\mathbf{x}, y), \quad (7.1)$$

where, following the general latent SVM approach [42, 169], the discriminant function is maximized over all possible latent attribute label assignments:

$$f_{\mathbf{w}}(\mathbf{x}, y) = \max_{\mathbf{h}} \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{h}, y), \quad (7.2)$$

where  $\Phi(\mathbf{x}, \mathbf{h}, y)$  is a feature vector that depends on its arguments.

---

<sup>2</sup>as found with a maximum spanning tree on a fully connected graph

Wang and Mori define the model as follows<sup>3</sup>:

$$\mathbf{w}^T \Phi(\mathbf{x}, \mathbf{h}, y) = \mathbf{w}_y^T \phi(\mathbf{x}; y) + \sum_{j \in \mathcal{V}} \mathbf{w}_{h_j}^T \varphi(\mathbf{x}; j, h_j) + \sum_{(j,k) \in \mathcal{E}} \mathbf{w}_{j,k}^T \psi(h_j, h_k) + \sum_{j \in \mathcal{V}} v_{y,h_j}, \quad (7.3)$$

where  $\mathbf{w}$  is the concatenation of all the first parameters appearing in the summands, and the other terms are the features composing  $\Phi(\mathbf{x}, \mathbf{h}, y)$ .

Those four features are defined as follows:

- **Object class component:**  $\phi(\mathbf{x}; y)$  is the probability image  $\mathbf{x}$  has object label  $y$ , which is obtained by training a multi-class SVM (ignoring attributes).
- **Attribute class component:**  $\varphi(\mathbf{x}; j, h_j)$  is the probability that the  $j$ -th attribute is  $h_j$ , obtained by training a binary SVM for attribute  $j$  (ignoring object labels).
- **Attribute-attribute component:**  $\psi(h_j, h_k)$  is a binary vector of length 4, with a 1 in one entry denoting which of the four possibilities is true: that both, neither, the  $j$ -th, or the  $k$ -th attributes are present.
- **Object-attribute component:**  $v_{y,h_j}$  is a learned parameter reflecting the frequency of object being  $y$  and the  $j$ -th attribute being  $h_j$ .

Note that the two first components use separately trained traditional SVMs to produce feature values, which are then weighted by the learned parameters

---

<sup>3</sup>We omit the class-specific attribute classifier proposed in [170].

$\mathbf{w}_y$  and  $\mathbf{w}_{h_j}$ .

To train the model (learn  $\mathbf{w}$ ), we use the non-convex cutting plane method of [29], which allows latent attribute labels for the training examples. We use a mixture of observed and latent attribute labels when dealing with partially labeled examples during the active learning loop (see Section 7.2.3). We use the true values of training images’ attribute labels when computing the hinge loss function in [170].

During testing, we use linear programming to determine the attribute labels  $\mathbf{h}_y^*$  that maximize  $f_{\mathbf{w}}$  (for every  $y$ ), and then predict the object label  $y^*$  as in Equation 7.1.

## 7.2 Active Learning with Objects and Attributes

We take a pool-based active learning approach, where the active learner surveys all unlabeled data to determine which labels to request next. In particular, we use an entropy-based function to score all  $\langle \text{image}, \text{label request} \rangle$  pairs according to their expected information, and then select the top ranked requests for labeling. Each request asks for one label: the object class, or a specific attribute value. After a person answers the selected requests, the newly labeled instances are appended to the training set (whether for attributes or objects), and then the classifier parameters are updated accordingly. The entire process repeats, for as long as more labeling effort is available. The product is a classifier that predicts object labels. Figure 7.1 summarizes the main data flow.

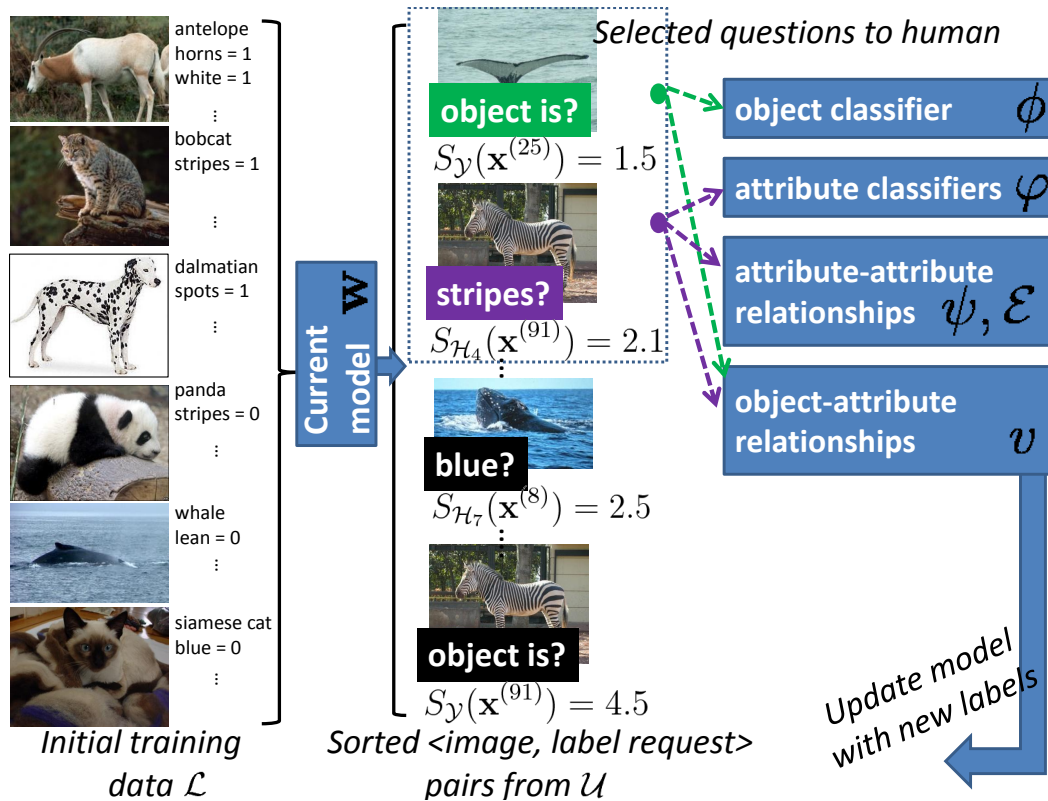


Figure 7.1: Overview of my approach. **Left:** the current model is determined by whatever labeled or partially labeled object and attribute data is available. Using that classifier, we score all  $\langle \text{image}, \text{label request} \rangle$  pairs in the unlabeled pool according to their expected entropy reduction. **Center:** The  $N$  top scoring pairs are presented to an annotator with the targeted object or attribute question. **Right:** Depending on the answers and label types, the annotator responses influence different components of the full model, as signified by the two sets of dotted arrows. Note that the four rightmost boxes parallel the four terms of the main model in Equation 7.3. **Loop:** Finally, we loop back, and repeat the selection process using the newly strengthened model. Best viewed in color.

In the following, I first define the sets of annotations that contribute to each component of the model (Section 7.2.1). Then I define the active selection function to rank the candidate label requests (Section 7.2.2), and then explain how those requests which the system acquires are used to update the model (Section 7.2.3).

### 7.2.1 Annotation Set Definitions

Let  $\mathcal{L}$  denote any labeled or partially labeled training data. Due to the different types of annotations and classifiers incorporated by the full model outlined above, we must maintain several separate training sets. As such, we think of  $\mathcal{L}$  as containing several (potentially overlapping) sets:  $\mathcal{L} = \{\mathcal{T}, \mathcal{T}_O, \mathcal{T}_{A_1}, \dots, \mathcal{T}_{A_K}, \mathcal{T}_A\}$ , where  $\mathcal{T}$  contains fully labeled images used to train the full model  $\mathbf{w}$ ,  $\mathcal{T}_O$  contains object-labeled images used to train the object classifier that yields feature  $\phi$ , each  $\mathcal{T}_{A_m}$  contains attribute-labeled images used to train the attribute classifier that yields feature  $\varphi$ , and  $\mathcal{T}_A$  contains attribute-labeled images used to compute the attribute relationship graph. Note that an annotation in  $\mathcal{L} \setminus \mathcal{T}$  still affects the full model, because it alters the inner components on top of which  $\mathbf{w}$  is learned.

Let  $\mathcal{U}$  denote all unlabeled (or partially unlabeled) data. Similar to above, we maintain separate sets according to the label “state” of a given example:  $\mathcal{U} = \{\mathcal{U}_O, \mathcal{U}_A\}$ , where examples in  $\mathcal{U}_O$  have no object label, and examples in  $\mathcal{U}_A$  lack one or more attribute labels.

### 7.2.2 Entropy-Based Selection Function

At the onset, we are given some initial pool of labeled data in  $\mathcal{L}$ . At each iteration of active learning, we need to decide which image to have annotated and which annotation to request for it. Thus, we must rank the pool of candidate  $\langle \text{image}, \text{label request} \rangle$  pairs in  $\mathcal{U}$ . A key point in my approach is that for a given image, there are  $K + 1$  options for the label query; it is either the object class, or one of the  $K$  attributes.

To this end, we define a selection function that scores the expected entropy reduction for a candidate request. Let  $(y^{(i)}, h_1^{(i)}, \dots, h_K^{(i)})$  denote the full labels for the  $i$ -th image  $\mathbf{x}^{(i)}$ . The total entropy over all labeled and unlabeled data given the labeled data  $\mathcal{L}$  is defined as:

$$H(\mathcal{L}) = - \sum_{u=1}^{|\mathcal{L} \cup \mathcal{U}|} \sum_{l=1}^{|\mathcal{Y}|} P_{\mathcal{L}}(y^{(u)} = l | \mathbf{x}^{(u)}) \log P_{\mathcal{L}}(y^{(u)} = l | \mathbf{x}^{(u)}), \quad (7.4)$$

where  $P_{\mathcal{L}}(y|\mathbf{x})$  denotes the posterior estimates obtained when the model is trained with labeled data  $\mathcal{L}$ . Note that entropy is measured over *object* predictions  $\mathcal{Y}$ , as that is the ultimate target label space.

In general, the unlabeled instance that maximizes the expected entropy reduction [135, 122] is:

$$\mathbf{x}^* = \arg \max_{\mathbf{x} \in \mathcal{U}} \left( H(\mathcal{L}) - \sum_{l=1}^{|\mathcal{Y}|} P_{\mathcal{L}}(y = l | \mathbf{x}) H(\mathcal{L} \cup \langle \mathbf{x}, l \rangle) \right), \quad (7.5)$$

or equivalently, if we drop the constant current entropy value:

$$\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathcal{U}} \left( \sum_{l=1}^{|\mathcal{Y}|} P_{\mathcal{L}}(y = l | \mathbf{x}) H(\mathcal{L} \cup \langle \mathbf{x}, l \rangle) \right). \quad (7.6)$$



In our case, we must consider expanding  $\mathcal{L}$  by *either* the object label  $y$  or an attribute label  $h_m$ . Thus we define two intermediate entropy scoring terms:

$$S_y(\mathbf{x}^{(i)}) = \sum_{l=1}^{|\mathcal{Y}|} P_{\mathcal{L}}(y^{(i)} = l | \mathbf{x}^{(i)}) H(\mathcal{L} \cup \langle \mathbf{x}^{(i)}, y^{(i)} = l \rangle), \quad (7.7)$$

where posteriors are obtained with the full object model, and

$$S_{\mathcal{H}_m}(\mathbf{x}^{(i)}) = \sum_{a=1}^{|\mathcal{A}|} P_{\mathcal{L}}(h_m^{(i)} = a | \mathbf{x}^{(i)}) H(\mathcal{L} \cup \langle \mathbf{x}^{(i)}, h_m^{(i)} = a \rangle), \quad (7.8)$$

where posteriors are obtained from the model’s inner attribute classifier  $\varphi$ . In both,  $\mathcal{L}$  refers to the current labeled data. Note that  $S_y$  and  $S_{\mathcal{H}_m}$  are comparable in that they both reflect the entropy of the *object* label prediction.

Finally, the best image and label request is given by:

$$(\mathbf{x}^*, \mathbf{q}^*) = \arg \min_{\mathbf{x} \in \mathcal{U}, \mathbf{q} \in \{\mathcal{Y}, \mathcal{H}_1, \dots, \mathcal{H}_K\}} S_{\mathbf{q}}(\mathbf{x}). \quad (7.9)$$

The lower the score in Equation 7.9, the more influence we expect the label request to have on the complete model. Because we consider the impact of a candidate labeling over all the data and model components, this selection function reveals which attribute or object-based question is most valuable, achieving the intuition given in Figure 1.5.

In order to compute the object class posterior probabilities required for entropy, we design a mapping from the raw  $f_{\mathbf{w}}$  function outputs to multi-class probabilities. First we estimate the pairwise probabilities for any two object classes  $l_A, l_B \in \mathcal{Y}$ , by fitting a sigmoid to output values for  $f_{\mathbf{w}}(\mathbf{x}, y =$

$l_A) - f_{\mathbf{w}}(\mathbf{x}, y = l_B)$  on the training data in  $\mathcal{T}$ . The difference between output values mimics the form of the latent SVM label constraints. Then we use the pairwise coupling approach [174] to obtain multi-class probabilities from these pairwise probabilities. In this way, we essentially adapt Platt’s method [111] to accommodate latent multi-class SVM outputs. For the attribute posteriors, we simply use Platt’s method on the binary SVM scores.

Procedurally, computing the best request requires cycling over the unlabeled or partially labeled images. Then, for each label request we could make for the current image, we cycle over each possible label response, and (1) add it to the labeled set temporarily, (2) retrain the model, (3) evaluate entropy under the new model, and (4) weight the resulting entropy by the probability of the hypothesized label under the old model. To request more than one label per iteration, we simply take the  $N$  queries with the lowest  $S_{\mathbf{q}}$  scores. Thus, one batch addition may include new labels for both objects and any of the attributes, and a given image may receive multiple labels.

### 7.2.3 Updates to Labeled and Partially Labeled Sets

Finally, I detail the implications that the above strategy has on the retraining step, whether adding true or hypothesized labels.

Recall that the model we are actively learning has two stages of training: the first updates the inner components (e.g., independent object or attribute classifiers), while the second updates the “outer” main parameters of  $\mathbf{w}$  (see Equation 7.3). Updates to either of the two annotation types do not affect

---

**Algorithm 1** The proposed active learning approach.

---

- 1: Given: labeled data  $\mathcal{L} = \{\mathcal{T}, \mathcal{T}_O, \mathcal{T}_{A_1}, \dots, \mathcal{T}_{A_K}, \mathcal{T}_A\}$ , and pool of unlabeled data  $\mathcal{U} = \{\mathcal{U}_O, \mathcal{U}_A\}$ .
  - 2: Compute initial attribute relationship graph  $(\mathcal{V}, \mathcal{E})$ .
  - 3: Compute features and train initial model using  $\mathcal{L}$ .
  - 4: **while** Labeling effort still available **do**
  - 5:   Compute  $S_y(\mathbf{x})$  for all images in  $\mathcal{U}_O$ .
  - 6:   Compute  $S_{\mathcal{H}_m}(\mathbf{x})$  for all images in  $\mathcal{U}_A$ , for all yet-unlabeled attributes among  $m = 1, \dots, K$ .
  - 7:   Select the  $N$  most informative image-label pairs (Equation 7.9), and ask human annotator.
  - 8:   Remove object-annotated images from  $\mathcal{U}_O$ .
  - 9:   Remove fully attribute-annotated images from  $\mathcal{U}_A$ .
  - 10:   Add new object-annotated images to  $\mathcal{T}_O$ .
  - 11:   Add images with new labels for attribute  $m$  to  $\mathcal{T}_{A_m}$ .
  - 12:   Infer values for any missing attribute labels for partially labeled images in  $\mathcal{U}_A \cap \mathcal{T}_O$ .
  - 13:   Add those and fully attribute-labeled images to  $\mathcal{T}_A$ .
  - 14:   Add images in  $\mathcal{T}_O \cap \mathcal{T}_A$  to  $\mathcal{T}$ ; remove them from  $U$ .
  - 15:   Recompute inner classifiers’ “features” using  $\mathcal{L}$ , update attribute graph.
  - 16:   Retrain the full model  $\mathbf{w}$  using  $\mathcal{T}$ .
  - 17: **end while**
- 

all inner components of the model at the same time, but they *do* always affect the full object prediction model parameters. In particular, new object labels are inserted into  $\mathcal{T}_O$ , and directly affect both the object classifier and learned object-attribute interaction terms. New attribute labels for the  $m$ -th attribute are inserted into  $\mathcal{T}_{A_m}$ , and directly affect the  $m$ -th attribute classifier, the attribute-attribute relationship graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , and the object-attribute interaction terms. These dependencies are reflected by the dotted arrows in the example shown in Figure 7.1.

Therefore, when we receive a new object label, we add it to  $\mathcal{T}_O$  and remove it from  $\mathcal{U}_O$ . When we receive a new label for attribute  $m$ , we add it to  $\mathcal{T}_{A_m}$ ; however, it is not removed from  $\mathcal{U}_A$  until all other attributes for that image are obtained. If a new label happens to complete all labels for a given image (i.e.,  $\mathbf{x}^{(i)} \in \mathcal{T}_O \cap \mathcal{T}_A$ ), we remove it from  $\mathcal{U}$  and insert it into  $\mathcal{T}$ .

In terms of updating the attribute relationship graph, if an object-labeled image in  $\mathcal{T}_O$  has only partial attribute labels, there are two options: (1) a *conservative* approach, where we simply wait until all attribute labels are present before adding it to  $\mathcal{T}_A$ , or (2) a *partial* approach, where we add the image to  $\mathcal{T}_A$  with its missing attribute labels inferred. To infer the missing labels, we add constraints in the linear programming problem that solves for  $h^*$  to reflect that any known attributes should be assigned their correct labels. After inferring these labels, we treat them as observed during training. For the partial approach, we keep the image in  $\mathcal{U}_A$ , so that its missing labels may still be added by a human annotator (if selected with active learning). I pursue this partial formulation in my experiments, as we expect more immediate impact of new labels to help the active learner.

Note that the partial update policy is applicable whether we are introducing a newly labeled instance received from an annotator (i.e., at the end of an active learning loop), or temporarily updating the model during the selection process. In the former they are permanent updates, while in the latter they are removed appropriately after the necessary posteriors are computed for Equation 7.9. Once we have updated the training and unlabeled sets ac-

cordingly, we retrain the inner classifiers, compute their features, and retrain the full model. See Algorithm 1 for a recap of the method.

## 7.3 Experimental Validation

I demonstrate my approach for object recognition on three challenging datasets. The main goal of my experiments is to demonstrate the advantage of choosing labels from both object and attribute types to validate the importance of a joint representation.

### 7.3.1 Experimental Design

We use the **Animals with Attributes** dataset introduced by Lampert et al. [87]. This dataset consists of 50 animal categories and 85 attributes. The attributes describe the fur color, fur patterns, size, anatomy, behavior, habitat etc. of the animals. We use three feature types as descriptors for these images: RGB histograms, PHOG, and rgSIFT, as given in [87]. We sample the categories and attributes from this dataset in different tasks, as described below.

We also use the **a-Yahoo-test** and **a-Pascal-train** datasets from [40]. The former consists of 12 classes and the latter of 20, which include animals, vehicles, household items, etc. These datasets use a set of 64 attributes, including shapes, textures, anatomy, and parts. Unlike the Animals with Attributes dataset, not every instance of a class in these datasets has the same attribute labels as all other instances in the same class. We use the provided bounding

boxes to maintain the assumption that there is only one object per image. All datasets are fairly challenging because of appearance variation and have been used to evaluate several recent approaches for learning from attribute labels.

I show results for four different splits of classes, two from the Animals with Attributes dataset (**AwA-1** and **AwA-2**), one from **a-Yahoo**, and one from **a-Pascal**.<sup>4</sup> I show examples of the datasets and class splits we use in Figure 7.2. We use splits to manage the cost of the selection process. (My method’s cost grows linearly with the number of object classes and quadratically with the number of evaluated unlabeled images.) For each split, we sample 200 images from an unlabeled pool in each iteration and compute the quality score of each candidate  $\langle \text{image}, \text{label request} \rangle$  pair in this set. The number of images in the full unlabeled pool and the separate test pool are: 1003/732 for AwA-1, 1002/993 for AwA-2, 903/287 for a-Pascal, and 703/200 for a-Yahoo. The initial number of labels on the learning curves (x-axis) is the number of training images per class<sup>5</sup> used to initialize the models times the number of categories times  $(K + 1)$ .

### 7.3.2 Baselines

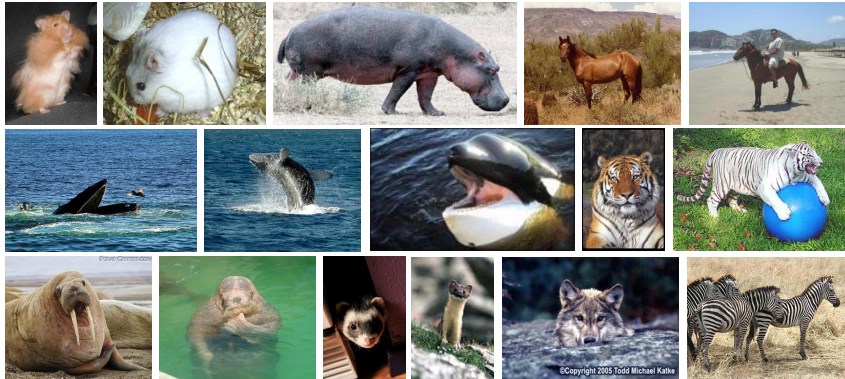
In my experiments, I compare my full active method, ACTIVE-OBJ+ATTR (OURS), which can request both object and attribute annotations, to a strong

---

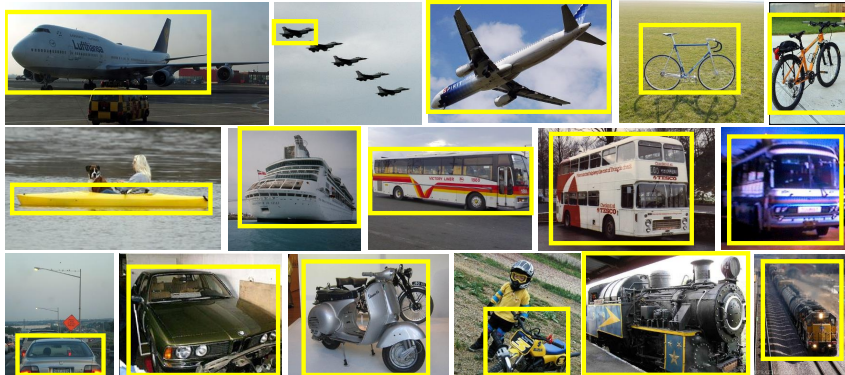
<sup>4</sup>Splits AwA-1 and AwA-2 classes: (*hamster, hippopotamus, horse, humpback whale, killer whale*) and (*tiger, walrus, weasel, wolf, zebra*). Split a-Yahoo: (*centaur, donkey, goat, monkey, wolf, zebra*). Split a-Pascal: (*aeroplane, bicycle, boat, bus, car, motorbike, train*).

<sup>5</sup>about 5; the exact numbers differ since we set the amount of initial training images proportionally to the number of images in each class

## Animals with Attributes



## a-Pascal



## a-Yahoo

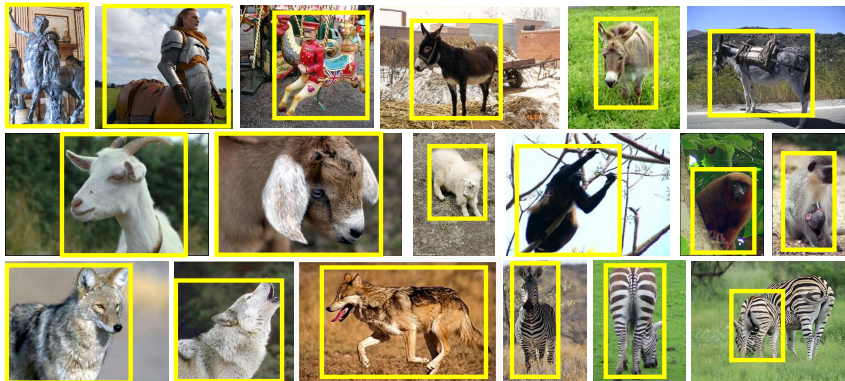


Figure 7.2: Example images from the datasets and classes I use in my experiments.

active baseline (ACTIVE-OBJ) which is just like my full method but can only request object annotations during the active loop. I also compare these methods to a passive baseline (RANDOM), which is also competitive since it randomly requests labels from the pool of candidate object *and* attribute labels. To my knowledge, no existing active learning approach learns from both object and attribute labels, making these the two best baselines to compare. Note that ACTIVE-OBJ has the disadvantage that additional object requests can only update the full model through the inner components, since an image has to be in both  $\mathcal{T}_O$  and  $\mathcal{T}_A$  to be added to  $\mathcal{T}$ . In one experiment, I also show the performance of OPTIMAL SELECTION—this result reveals how entropy would be affected if we knew the true labels of the unlabeled images rather than relying on the expected entropy reduction value.

### 7.3.3 Results and Discussion

**Active learning curves** We first study the effect of my method and the baselines on the confidence of the correct label on a held-out test set. Figure 7.3 reports the average probability of the correct label predicted by each method as a function of the number of labels added. These probabilities are computed over active learning iterations, where each time a selection of  $N = 5$  labels is added for all methods. These standard learning curves aim to demonstrate that to achieve some given accuracy, my method usually requires fewer labels than any of the baselines. Higher values in the curve indicate more improvement in the classifier and a higher confidence of classification.



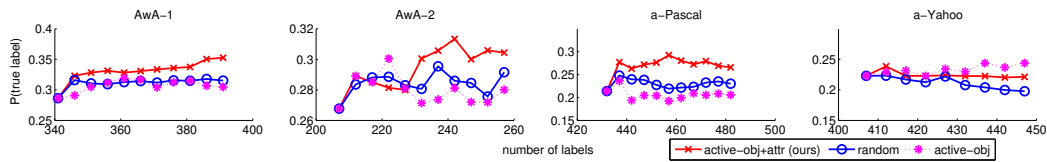


Figure 7.3: Representative learning curves from all three datasets showing the predicted probability of the correct label on a held-out test set with increasing number of labels obtained: first three are best, and fourth represents a failure case. My approach is constantly more accurate than the baselines, indicating that jointly selecting from object and attribute labels is worthwhile. (Higher curves are better.)

All three approaches improve upon the initial classifier with more labels, but at different rates per label. My joint approach shows the most significant gains in accuracy with fewer labels. RANDOM selection wastes annotator effort on less informative examples and labels. ACTIVE-OBJ is in general as good as or worse than RANDOM; the latter can happen since RANDOM has the advantage of additional attribute labels. The poorer performance of ACTIVE-OBJ in comparison with ACTIVE-OBJ+ATTR (OURS) validates my main claim: it is more advantageous to select labels actively among both objects and attributes rather than just objects. Note that I show confidence rather than simply accuracy, since confidence reveals more fully how models have improved. For reference, the confidence results after training on all images in each class (excepting test images) are: .3810, .3745, .3852, .4020. Therefore, using only 4% of the total labels on average, my method achieves 74.48% of the ultimate confidence level. In comparison, RANDOM achieves 67.39% and ACTIVE-OBJ achieves 67.20% of the ultimate confidence level using the same number of labels (last point on the x-axis).

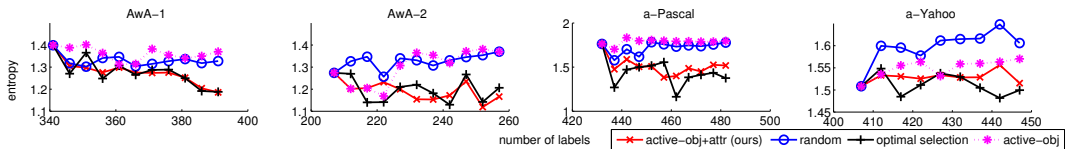


Figure 7.4: Entropy of all training and unlabeled examples with increasing number of obtained labels for my approach in comparison to optimal selection and the baselines (first three are good, and last is a failure case). As expected, my approach reduces the overall classification uncertainty faster than the baselines and similarly to optimal selection. (Lower curves are better.)

**Actual entropy reduction** Next, we examine how the uncertainty on the training and unlabeled sets changes as the different methods make their selections. Figure 7.4 reports the mean entropy on  $\mathcal{L} \cup \mathcal{U}$  as more labels are added for the three approaches and the optimal selection. We want entropy to decrease as more training data is added, so lower curves are better. Optimal selection shows the result of using ground truth information in order to compute the best possible selection based on entropy.<sup>6</sup> The overall entropy decreases steadily with more labels for both my approach and the optimal selection, showing that the classifier is able to better separate the examples into the different classes by jointly learning from object and attribute labels. Note that my approach performs quite similarly to optimal selection. The same is not true for the baselines, where the reduction in the overall uncertainty is slower.

The last figure in Figure 7.4 shows a case where entropy is poorly

---

<sup>6</sup>This result would strictly be an upper bound to my approach, but since multiple labels are added at a time, the entropy reduction predicted for individual labels and the actual entropy reduction can differ.

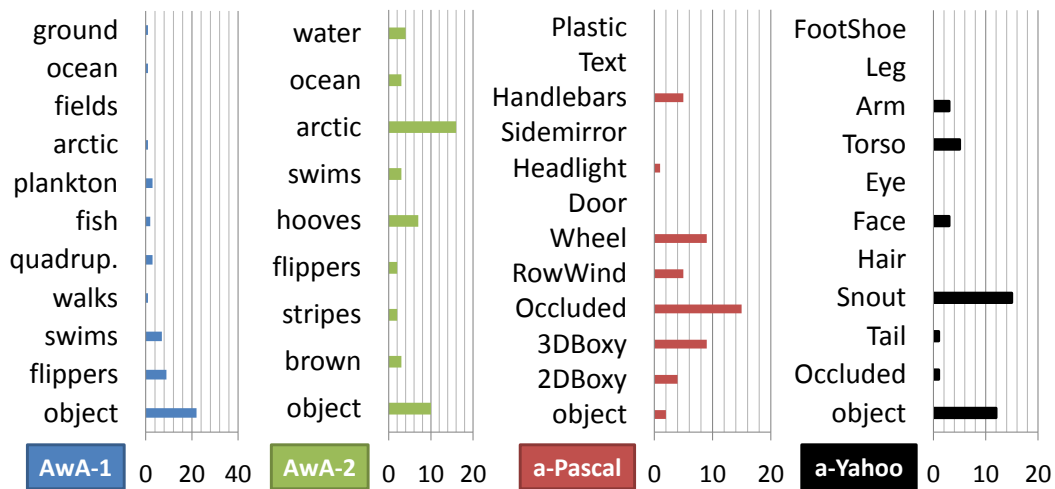


Figure 7.5: Distribution of requests per label type (object/attributes).

estimated—compare my method’s result versus the optimal selection in the fourth plot. This explains the failure case for the test set learning curves in the fourth plot in Figure 7.3.

**Qualitative results** To examine in more depth the selections made by my active learning approach, I present some qualitative results. In Figure 7.5, I show the distribution of label requests for the object and each of the attribute labels. We see that the majority ( $\sim 75\%$ ) of requests are for attribute labels. There is a slight tendency (not shown) that more object labels are requested earlier in the active learning loop. We see that the distribution for a-Yahoo is the least balanced one, which might indicate a particular relationship between objects and attributes that would explain the weaker performance of my method on this dataset. In Figure 7.6, I show some sample requests that



Figure 7.6: Sample  $\langle$ image, label $\rangle$  requests that my method generates for AwA-1. The 1<sup>st</sup> request may be explained by the lack of dark brown hamsters in the training set. The 2<sup>nd</sup> and 3<sup>rd</sup> requests are due to the similarity to classes that have the attributes in question. The 4<sup>th</sup> and 5<sup>th</sup> requests show an image which confuses the system and appears in multiple labeling requests. The 6<sup>th</sup> image is likely ambiguous because it violates the assumption of one object per image.

were made by my algorithm for AwA-1.

## 7.4 Conclusions

In this chapter, I proposed a method for actively selecting the best object or attribute labels on images in a way that can simultaneously affect multiple object categories. In contrast to existing work, my approach both weighs different annotation requests and also models dependencies between the target label space and a latent but human-describable label space. My results on three challenging datasets indicate that my method is indeed able to learn more quickly than either passive learning or a strong baseline approach that can only request object labels. The proposed strategy can be seen as a means to enhance multi-class object category learning, by efficiently strengthening models through shared attributes.

The category models that my method efficiently learns can be used to address user queries in terms of keywords like “cat” or “sofa”. In the context of

search, a large number of object categories need to be pre-learned, so efficiency is particularly important. One can envision a system that actively learns how to automatically tag images with categories using the method presented in this chapter, and then an image search module that initializes its first set of returned images using those tags and subsequently allows interactive feedback as described in the previous chapters.

## Chapter 8

### Future Work

There are numerous interesting extensions of the work I develop in this thesis. This work also invites some broader research directions.

One potentially useful extension is to develop an active selection approach that can choose between relative attribute and binary feedback requests. This approach would ensure that the hybrid approach presented in Chapter 3 receives the most useful responses. In order to preserve the computational efficiency of the selection strategy, the system must only evaluate a small fraction of the images on which it might request binary feedback. This might be achieved by evaluating as candidates a mixture of a small set of images which currently have high probabilities of being relevant, and a small set of images that have low such probabilities. Alternatively, one can form a tree of representative image clusters computed by hierarchical clustering, where the dataset is split in half at each tree level. Then the system can ask the user which of two images (at the top of two branches) is more similar to the target, until it reaches the right level of similarity to the target when the user can terminate the search.

A valuable extension of the work in Chapter 4 would be to account

for the quality of the attribute models in the formulation for selecting the next pivot for feedback. For example, if we trust the model for the attribute “pointy” more than the model for the attribute “shiny” as it is substantially more accurate on validation data, yet our estimates indicate a feedback request on “shiny” would lead to a slightly higher expected information gain than on “pointy”, we should perhaps still request feedback on “pointy”, because that estimate is more likely to be accurate. Therefore, one could develop a framework where the quality of a model is used to weight the expected entropy scores. Further, we may weight entropy scores by *how easy* it is for a user to provide response for the given attribute. In the case of the OSR dataset, it might be easier for a user to determine which of two scenes is more “natural”, than to determine which has a “diagonal plane”.

The active selection method could further be enhanced by developing a far-sighted [160] active selection technique that determines how to optimally select a batch of questions as opposed to making the myopically optimal choice at each round. Finally, one could consider a “mixed initiative” approach [136, 18, 173], where the system decides when to give the control back to the user. In this case, the WhittleSearch system would switch between free-form (user-guided) and active (system-guided) modes.

In Chapter 5, I proposed an efficient method for adapting attribute models to a user’s unique perception of this attribute. Utilizing the “schools of thought” proposed in Chapter 6 helps guard against potential noise in a user’s responses and against over-personalization. However, a user’s divergent

notion of an attribute might range in complexity, and a fixed number of user-specific labels might not be sufficient to capture it. Therefore, one needs to decide *whether and how to adapt*, as well as which images to request to have labeled. Adaptation can then be performed dynamically, by updating the choice of questions to ask the user based on the previous questions he answers. Furthermore, we can continue asking questions until we are certain that we have accurately captured the user’s unique attribute notion. If we can accurately predict the user’s response at iteration  $n + 1$  from the model we have learned at iteration  $n$ , then perhaps we can stop asking questions. We can also use the change in entropy (or lack thereof) between subsequent iterations to determine when our model has mostly captured the user’s perception.

In the context of shades (Chapter 6), it would be useful to develop a scheme for automatically and quickly predicting to which shade a user subscribes, so that we utilize training data from other users subscribing to this shade in learning an attribute prediction model for this user. It would also be valuable to develop an approach for discovering shades for relative attribute rankers (in addition to binary attribute classifiers), which is a straight-forward but yet untested extension.

Finally, in order to ensure that the attribute models have learned what we intended for them to learn, it would be valuable to develop a strategy for visualizing attribute models. The low-level appearance of attributes as captured by image features varies much more across instances of the attribute than it does for object category instances, so using existing visualization approaches



such as [162] will likely not work. Yet if we can show how an attribute expresses itself visually, we can gain some insight that would let us “debug” the attribute models we have learned.

## Chapter 9

### Conclusion

In this thesis, I propose techniques for interactive image search with the help of visual attributes. I propose a novel mode of feedback where a user directly describes how high-level properties of exemplar images should be adjusted in order to more closely match her envisioned target images, and show that the proposed relative attribute feedback is more powerful than traditional binary relevance feedback. Building on this idea, I also present an approach which actively selects the images and attributes on which the user should provide feedback. I show that this active selection method is more efficient in terms of both computation and user time, compared to relevant methods. Since this approach depends critically on having semantic attributes, I devise a method for efficiently capturing the user’s true perception of the attributes used in search. The resulting user-adaptive models align more closely with individual users’ perceptions of attributes compared to both generic models and ones learned solely from the given user’s data. Generalizing this idea, I describe an approach for disambiguating attribute terms in the joint space of visual perception and language, and use the discovered attribute shades of meaning to learn more robust attribute prediction models. Finally, I show how attributes can help learn object categories faster, in an active learning frame-

work where the computer vision learning system actively solicits annotations from a pool of object and attribute labels for training a joint object-attribute model. My work focuses on the modes in which an image retrieval system communicates with its users, and the techniques I propose are a promising step in closing the semantic gap.

While the novel search interaction that I propose in this thesis is applied to image retrieval, it can also be applied to other forms of information retrieval as well. For example, if a user is trying to find a song that she heard, and the system presents her with an initial guess based on a set of keywords that the user typed, the user can respond with “like this, but slower” or “more upbeat”. Similarly, recent work builds on WhittleSearch to enable users to describe and retrieve fonts using relative attributes [102] and perform affinity feedback on attributes [181]. Relevance feedback based on relative attributes can also be used in general document search, as long as the documents can be described with properties that both the system and the user understand.

The main novelty in my work, which makes my thesis a contribution to information retrieval in general, is the proposed feedback based on comparisons in terms of attributes (concepts). The feedback selection formulation is novel as well, and is applicable in other forms of information retrieval aside from image retrieval. Giving feedback based on attributes allows for efficient personalization via domain adaptation. Finally, discovering attribute shades of meaning is an attempt to bridge the gap between language and pictures, e.g., disambiguating a user’s statement such as “show me fancy brown furniture

like this”, which should allow for more efficient multi-modal search.

## Bibliography

- [1] Alex Acero, Li Deng, Trausti Kristjansson, and Jerry Zhang. HMM Adaptation using Vector Taylor Series for Noisy Speech Recognition. In *ICSLP*, 2000.
- [2] Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. Label-Embedding for Attribute-Based Classification. In *CVPR*, 2013.
- [3] Hani Altwaijry and Serge Belongie. Relative Ranking of Facial Attractiveness. In *WACV*, 2013.
- [4] Joshua Attenberg, Kilian Weinberger, and Anirban Dasgupta. Collaborative Email-Spam Filtering with the Hashing-Trick. In *CEAS*, 2009.
- [5] Yusuf Aytar and Andrew Zisserman. Tabula Rasa: Model Transfer for Object Category Detection. In *ICCV*, 2011.
- [6] Boris Babenko, Steve Branson, and Serge Belongie. Similarity Metrics for Categorization: from Monolithic to Category Specific. In *ICCV*, 2009.
- [7] Kobus Barnard and Keiji Yanai. Mutual Information of Words and Pictures. *Information Theory and Applications*, 2, 2006.

- [8] Kobus Barnard, Keiji Yanai, Matthew Johnson, and Prasad Gabbur. Cross Modal Disambiguation. *Toward Category-Level Object Recognition*, 2006.
- [9] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up Robust Features (SURF). In *CVIU*, 2008.
- [10] Ori Ben-Yitzhak, Nadav Golbandi, Nadav Har'El, Ronny Lempel, Andreas Neumann, Shila Ofek-Koifman, Dafna Sheinwald, Eugene Shekita, Benjamin Sznajder, and Sivan Yogev. Beyond Basic Faceted Search. In *WSDM*, 2008.
- [11] Tamara L. Berg, Alexander C. Berg, and Jonathan Shih. Automatic Attribute Discovery and Characterization from Noisy Web Data. In *ECCV*, 2010.
- [12] Tamara L. Berg and David A. Forsyth. Animals on the Web. In *CVPR*, 2006.
- [13] Arijit Biswas and Devi Parikh. Simultaneous Active Learning of Classifiers and Attributes via Relative Feedback. In *CVPR*, 2013.
- [14] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain Adaptation with Structural Correspondence Learning. In *EMNLP*, 2006.
- [15] Anna Bosch, Andrew Zisserman, and Xavier Muñoz. Representing Shape with a Spatial Pyramid Kernel. In *CIVR*, 2007.

- [16] Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Describing People: A Poselet-Based Approach to Attribute Classification. In *ICCV*, 2011.
- [17] Steve Branson, Catherine Wah, Florian Schroff, Boris Babenko, Peter Welinder, Pietro Perona, and Serge Belongie. Visual Recognition with Humans in the Loop. In *ECCV*, 2010.
- [18] Maya Cakmak and Andrea L. Thomaz. Mixed-Initiative Active Learning. In *ICML Workshop on Combining Learning Strategies to Reduce Label Cost*, 2011.
- [19] Rich Caruana. Multitask Learning. *Machine Learning*, 28(1):41–75, 1997.
- [20] Huizhong Chen, Andrew Gallagher, and Bernd Girod. What’s in a Name? First Names as Facial Attributes. In *CVPR*, 2013.
- [21] Jonghyun Choi, Mohammad Rastegari, Ali Farhadi, and Larry Davis. Adding Unlabeled Samples to Categories by Learned Attributes. In *CVPR*, 2013.
- [22] Brendan Collins, Jia Deng, Kai Li, and Li Fei-Fei. Towards Scalable Dataset Construction: An Active Learning Approach. In *ECCV*, 2008.
- [23] Ingmar Cox, Matt Miller, Thomas Minka, Thomas Papatomas, and Peter Yianilos. The Bayesian Image Retrieval System, PicHunter: Theory,

- Implementaion and Psychophysical Experiments. *IEEE Transactions on Image Processing*, 2000.
- [24] William Curran, Travis Moore, Todd Kulesza, Weng-Keen Wong, Sinisa Todorovic, Simone Stumpf, Rachel White, and Margaret Burnett. Towards Recognizing "Cool": Can End Users Help Computer Vision Recognize Subjective Attributes or Objects in Images? In *IUI*, 2012.
- [25] Scott Deerwester, Susan Dumais, and Thomas Landauer. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 1990.
- [26] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.
- [27] Jia Deng, Jonathan Krause, and Li Fei-Fei. Fine-Grained Crowdsourcing for Fine-Grained Recognition. In *CVPR*, 2013.
- [28] Chaitanya Desai, Deva Ramanan, and Charless Fowlkes. Discriminative Models for Multi-Class Object Layout. In *ICCV*, 2009.
- [29] Trinh-Minh-Tri Do and Thierry Artières. Large Margin Training for Hidden Markov Models and Partially Observed States. In *ICML*, 2009.
- [30] Jeff Donahue and Kristen Grauman. Annotator Rationales for Visual Recognition. In *ICCV*, 2011.



- [31] Matthijs Douze, Arnau Ramisa, and Cordelia Schmid. Combining Attributes and Fisher Vectors for Efficient Image Retrieval. In *CVPR*, 2011.
- [32] Gregory Druck, Burr Settles, and Andrew McCallum. Active Learning by Labeling Features. In *EMNLP*, 2009.
- [33] Kun Duan, Devi Parikh, David Crandall, and Kristen Grauman. Discovering Localized Attributes for Fine-grained Recognition. In *CVPR*, 2012.
- [34] Mark D Dunlop. The Effect of Accessing Non-matching Documents on Relevance Feedback. *ACM Transactions on Information Systems*, 15:137–153, April 1997.
- [35] Ian Endres, Ali Farhadi, Derek Hoiem, and David A. Forsyth. Benefits and Challenges of Collecting Richer Object Annotations. In *ACVHL*, 2010.
- [36] Seyda Ertekin, Haym Hirsh, and Cynthia Rudin. Approximating the Wisdom of the Crowd. In *NIPS Workshop on Computational Social Science and the Wisdom of Crowds*, 2011.
- [37] Caleb Everett. *Linguistic Relativity: Evidence Across Languages and Cognitive Domains*. Mouton De Gruyter, 2013.

- [38] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes VOC Challenge. *IJCV*, 88(2):303–338, 2010.
- [39] Ali Farhadi, Ian Endres, and Derek Hoiem. Attribute-Centric Recognition for Cross-category Generalization. In *CVPR*, 2010.
- [40] Ali Farhadi, Ian Endres, Derek Hoiem, and David A. Forsyth. Describing Objects by Their Attributes. In *CVPR*, 2009.
- [41] Li Fei-Fei, Rob Fergus, and Pietro Perona. A Bayesian Approach to Unsupervised One-Shot Learning of Object Categories. In *ICCV*, 2003.
- [42] Pedro Felzenszwalb, Ross Girshick, David McAllester, and Deva Ramanan. Object Detection with Discriminatively Trained Part Based Models. *TPAMI*, 32(9):1627–1645, 2010.
- [43] Marin Ferecatu and Donald Geman. Interactive Search for Image Categories by Mental Matching. In *ICCV*, 2007.
- [44] Rob Fergus, Li Fei-Fei, Pietro Perona, and Andrew Zisserman. Learning Object Categories from Google’s Image Search. In *CVPR*, 2005.
- [45] Vittorio Ferrari and Andrew Zisserman. Learning Visual Attributes. In *NIPS*, 2007.
- [46] Myron Flickner, Harpeet Sawhney, Wayne Nilback, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee,

- Dragutin Petkovic, David Steele, and Peter Yanker. Query by Image and Video Content: The QBIC System. *IEEE Computer*, 28(9):23–32, September 1995.
- [47] James Fogarty, Desney S. Tan, Ashish Kapoor, and Simon Winder. CueFlik: Interactive Concept Learning in Image Search. In *CHI*, 2008.
- [48] Jean Luc Gauvain and Chin-Hui Lee. Maximum A Posterior Estimation For Multivariate Gaussian Mixture Observations Of Markov Chains. *IEEE Transactions on Speech and Audio Processing*, 2, 1994.
- [49] Donald Geman and Bruno Jedynak. Model-Based Classification Trees. *IEEE Transactions on Information Theory*, 1998.
- [50] Bo Geng, Linjun Yang, Chao Xu, and Xian-Sheng Hua. Ranking Model Adaptation for Domain-Specific Search. *IEEE Transactions on Knowledge and Data Engineering*, March 2010.
- [51] Nicolo Giorgetti. GLPKMEX - a Matlab MEX Interface for the GLPK Library.
- [52] David Gleich. Matlab BGL. [http://www.stanford.edu/~dgleich/programs/matlab\\_bgl/](http://www.stanford.edu/~dgleich/programs/matlab_bgl/).
- [53] Ryan Gomes, Peter Welinder, Andreas Krause, and Pietro Perona. Crowd-clustering. In *NIPS*, 2011.

- [54] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic Flow Kernel for Unsupervised Domain Adaptation. In *CVPR*, 2012.
- [55] Raghuraman Gopalan, Ruonan Li, and Rama Chellapa. Domain Adaptation for Object Recognition: An Unsupervised Approach. In *ICCV*, 2011.
- [56] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 Object Category Dataset. Technical report, Caltech, 2007.
- [57] Abhinav Gupta and Larry Davis. Beyond Nouns: Exploiting Prepositions and Comparative Adjectives for Learning Visual Classifiers. In *ECCV*, 2008.
- [58] David Hall, Daniel Jurafsky, and Christopher D Manning. Studying the History of Ideas Using Topic Models. In *EMNLP*, 2008.
- [59] Judy Hoffman, Brian Kulis, Trevor Darrell, and Kate Saenko. Discovering Latent Domains for Multisource Domain Adaptation. In *ECCV*, 2012.
- [60] Thomas Hofmann. Probabilistic Latent Semantic Analysis. In *UAI*, 1999.
- [61] Qasim Iqbal and J. K. Aggarwal. CIRES: A System for Content-based Retrieval in Digital Image Libraries. In *International Conference on Control, Automation, Robotics and Vision*, 2002.

- [62] Prateek Jain and Ashish Kapoor. Active Learning for Large Multi-class Problems. In *CVPR*, 2009.
- [63] Dinesh Jayaraman, Fei Sha, and Kristen Grauman. Decorrelating Semantic Visual Attributes by Resisting the Urge to Share. In *CVPR*, 2014.
- [64] Thorsten Joachims. Optimizing Search Engines Using Clickthrough Data. In *KDD*, 2002.
- [65] Thorsten Joachims. Training Linear SVMs in Linear Time. In *KDD*, 2006.
- [66] Ajay Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. Multi-Class Active Learning for Image Classification. In *CVPR*, 2009.
- [67] Hiroshi Kajino, Yuta Tsuboi, and Hisashi Kashima. A Convex Formulation for Learning from Crowds. In *AAAI*, 2012.
- [68] Sing Bing Kang, Ashish Kapoor, and Dani Lischinski. Personalization of Image Enhancement. In *CVPR*, 2010.
- [69] Rianne Kaptein, Jaap Kamps, and Djoerd Hiemstra. The Impact of Positive, Negative and Topical Relevance Feedback. In *TREC*, 2008.
- [70] J. Kekalainen and K. Jarvelin. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

- [71] M. Kendall. A New Measure of Rank Correlation. *Biometrika*, 1938.
- [72] Jonathan Koren, Yi Zhang, and Xue Liu. Personalized Interactive Faceted Search. In *WWW*, 2008.
- [73] Yehuda Koren. Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model. In *KDD*, 2008.
- [74] Yehuda Koren. Collaborative Filtering with Temporal Dynamics. In *KDD*, 2009.
- [75] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer Magazine*, 2009.
- [76] Adriana Kovashka and Kristen Grauman. Attribute Adaptation for Personalized Image Search. In *ICCV*, 2013.
- [77] Adriana Kovashka and Kristen Grauman. Attribute Pivots for Guiding Relevance Feedback in Image Search. In *ICCV*, 2013.
- [78] Adriana Kovashka, Devi Parikh, and Kristen Grauman. WhittleSearch: Image Search with Relative Attribute Feedback. In *CVPR*, 2012.
- [79] Adriana Kovashka, Sudheendra Vijayanarasimhan, and Kristen Grauman. Actively Selecting Annotations Among Objects and Attributes. In *ICCV*, 2011.

- [80] R Kuhn, P Nguyen, J-C Junqua, L Goldwasser, N Niedzielski, S Fincke, K Field, and M Contolini. Eigenvoices for Speaker Adaptation. In *ICSLP*, 1998.
- [81] Bill Kules, Robert Capra, Matthew Banta, and Tito Sierra. What Do Exploratory Searchers Look at in a Faceted Search Interface? In *JCDL*, 2009.
- [82] Praveen Kulkarni, Gaurav Sharma, Joaquin Zepeda, and Louis Chevalier. Transfer Learning via Attributes for Improved On-the-fly Classification. In *WACV*, 2014.
- [83] Neeraj Kumar, Peter N. Belhumeur, and Shree K. Nayar. FaceTracer: A Search Engine for Large Collections of Images with Faces. In *ECCV*, 2008.
- [84] Neeraj Kumar, Alexander C. Berg, Peter N. Belhumeur, and Shree K. Nayar. Attribute and Simile Classifiers for Face Verification. In *ICCV*, 2009.
- [85] Neeraj Kumar, Alexander C. Berg, Peter N. Belhumeur, and Shree K. Nayar. Describable Visual Attributes for Face Verification and Image Search. *TPAMI*, 33(10):1962–1977, 2011.
- [86] Takio Kurita and Toshikazu Kato. Learning of Personal Visual Impression for Image Database Systems. In *ICDAR*, 1993.

- [87] Christoph Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to Detect Unseen Object Classes By Between-Class Attribute Transfer. In *CVPR*, 2009.
- [88] Kenneth Wai-Ting Leung, Wilfred Ng, and Dik Lun Lee. Personalized Concept-Based Clustering of Search Engine Queries. *IEEE Transactions on Knowledge and Data Engineering*, 2007.
- [89] Beita Li, Edward Chang, and Chung-Sheng Li. Learning Image Query Concepts via Intelligent Sampling. In *ICME*, 2001.
- [90] Shaoxin Li, Shiguang Shan, and Xilin Chen. Relative Forest for Attribute Prediction. In *ACCV*, 2012.
- [91] Tie Liu, Jian Sun, Nan-Ning Zheng, Xiaoou Tang, and Heung-Yeung Shum. Learning To Detect A Salient Object. In *CVPR*, 2007.
- [92] Nicholas Loeff, Cecilia Ovesdotter Alm, and David A. Forsyth. Discriminating Image Senses by Clustering with Multimodal Features. In *ACL*, 2006.
- [93] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 60(2):91–110, 2004.
- [94] Wei-Ying Ma and B. S. Manjunath. NeTra: A Toolbox for Navigating Large Image Databases. In *ICIP*, 1997.



- [95] Sean D. MacArthur, Carla E. Brodley, and Chi-Ren Shyu. Relevance Feedback Decision Trees in Content-Based Image Retrieval. In *IEEE Workshop on Content-Based Access of Image and Video Libraries*, 2000.
- [96] Dhruv Mahajan, Sundararajan Sellamanickam, and Vinod Nair. A Joint Learning Framework for Attribute Models and Object Descriptions. In *ICCV*, 2011.
- [97] Subhransu Maji. Discovering a Lexicon of Parts and Attributes. In *ECCV Workshop on Parts and Attributes*, 2012.
- [98] Andrew Makhorin. GLPK (GNU Linear Programming Kit), 2008.
- [99] Tim Matthews, Mark S. Nixon, and Mahesan Niranjan. Enriching Texture Analysis with Semantic Data. In *CVPR*, 2013.
- [100] Thomas Mensink, Jakob Verbeek, and Gabriela Csurka. Learning Structured Prediction Models for Interactive Image Labeling. In *CVPR*, 2011.
- [101] Milind Naphade, John R. Smith, Jelena Tesic, Shih-Fu Chang, Winston Hsu, Lyndon Kennedy, Alexander Hauptmann, and Jon Curtis. Large-Scale Concept Ontology for Multimedia. *IEEE Transactions on Multimedia*, 2006.
- [102] Peter O’Donovan, Jnis Lbeks, Aseem Agarwala, and Aaron Hertzmann. Exploratory Font Selection Using Crowdsourced Attributes. In *SIGGRAPH*, 2014.

- [103] Aude Oliva and Antonio Torralba. Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope. *IJCV*, 42:145–175, 2001.
- [104] Vicente Ordonez, Jia Deng, Yejin Choi, Alexander C. Berg, and Tamara L. Berg. From Large Scale Image Categorization to Entry-Level Categories. In *ICCV*, 2013.
- [105] Vicente Ordonez, Vignesh Jagadeesh, Wei Di, Anurag Bhardwaj, and Robinson Piramuthu. Furniture-Geek: Understanding Fine-Grained Furniture Attributes from Freely Associated Text and Tags. In *WACV*, 2014.
- [106] Devi Parikh and Kristen Grauman. Interactively Building a Discriminative Vocabulary of Nameable Attributes. In *CVPR*, 2011.
- [107] Devi Parikh and Kristen Grauman. Relative Attributes. In *ICCV*, 2011.
- [108] Amar Parkash and Devi Parikh. Attributes for Classifier Feedback. In *ECCV*, 2012.
- [109] Gabriella Pasi. Issues in Personalizing Information Retrieval. *IEEE Intelligent Informatics Bulletin*, 2010.
- [110] Genevieve Patterson and James Hays. SUN Attribute Database: Discovering, Annotating, and Recognizing Scene Attributes. In *CVPR*, 2012.

- [111] John C. Platt. Probabilistic Output for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In *Advances in Large Margin Classifiers*, 1999.
- [112] Guo-Jun Qi, Xian-Sheng Hua, Yong Rui, Jinhui Tang, and Hong-Jiang Zhang. Two-Dimensional Active Learning for Image Classification. In *CVPR*, 2008.
- [113] Ariadna Quattoni, Michael Collins, and Trevor Darrell. Transfer Learning for Image Classification with Sparse Prototype Representations. In *CVPR*, 2008.
- [114] Hema Raghavan, Omid Madani, and Rosie Jones. InterActive Feature Selection. In *IJCAI*, 2005.
- [115] Nikhil Rasiwasia, Pedro J. Moreno, and Nuno Vasconcelos. Bridging the Gap: Query by Semantic Example. *IEEE Transactions on Multimedia*, 2007.
- [116] Mohammad Rastegari, Ali Farhadi, and David A. Forsyth. Attribute Discovery via Predictable Discriminative Binary Codes. In *ECCV*, 2012.
- [117] Mohammad Rastegari, Devi Parikh, Ali Diba, and Ali Farhadi. Multi-Attribute Queries: To Merge or Not to Merge? In *CVPR*, 2013.
- [118] Daniel A. Reid and Mark S. Nixon. Using Comparative Human Descriptions for Soft Biometrics. In *International Joint Conference on Biometrics*, 2011.

- [119] Marcus Rohrbach, Michaela Regneri, Mykhaylo Andriluka, Sikandar Amin, Manfred Pinkal, and Bernt Schiele. Script Data for Attribute-based Recognition of Composite Activities. In *ECCV*, 2012.
- [120] Marcus Rohrbach, Michael Stark, György Szarvas, Iryna Gurevych, and Bernt Schiele. What Helps Where - And Why? Semantic Relatedness for Knowledge Transfer. In *CVPR*, 2010.
- [121] P. Rousseeuw. Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Computational and Applied Mathematics*, 20:53–65, 1987.
- [122] Nicholas Roy and Andrew McCallum. Toward Optimal Active Learning through Sampling Estimation of Error Reduction. In *ICML*, 2001.
- [123] Yong Rui, Thomas S. Huang, Michael Ortega, and Sharad Mehrotra. Relevance Feedback: A Power Tool for Interactive Content-Based Image Retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 1998.
- [124] Olga Russakovsky and Li Fei-Fei. Attribute Learning in Large-Scale Datasets. In *ECCV Workshop on Parts and Attributes*, 2010.
- [125] Bryan Russell, Antonio Torralba, Kevin Murphy, and William Freeman. LabelMe: A Database and Web-Based Tool for Image Annotation. *IJCV*, 77:157–173, 2008.

- [126] Amir Sadvnik, Andrew Gallagher, and Tsuhan Chen. It's Not Polite To Point: Describing People With Uncertain Attributes. In *CVPR*, 2013.
- [127] Amir Sadvnik, Andrew Gallagher, Devi Parikh, and Tsuhan Chen. Spoken Attributes: Mixing Binary and Relative Attributes to Say the Right Thing. In *ICCV*, 2013.
- [128] Kate Saenko and Trevor Darrell. Unsupervised Learning of Visual Sense Models for Polysemous Words. In *NIPS*, 2008.
- [129] Kate Saenko and Trevor Darrell. Filtering Abstract Senses from Image Search Results. In *NIPS*, 2009.
- [130] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting Visual Category Models to New Domains. In *ECCV*, 2010.
- [131] Ruslan Salakhutdinov and Andrey Mnih. Probabilistic Matrix Factorization. In *NIPS*, 2007.
- [132] Ruslan Salakhutdinov and Andrey Mnih. Bayesian Probabilistic Matrix Factorization using Markov Chain Monte Carlo. In *ICML*, 2008.
- [133] Babak Saleh, Ali Farhadi, and Ahmed Elgammal. Object-Centric Anomaly Detection by Attribute-Based Reasoning. In *CVPR*, 2013.
- [134] Walter Scheirer, Neeraj Kumar, Peter N. Belhumeur, and Terrance E. Boult. Multi-Attribute Spaces: Calibration for Attribute Fusion and Similarity Search. In *CVPR*, 2012.

- [135] Burr Settles. Active Learning Literature Survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.
- [136] Burr Settles and Xiaojin Zhu. Behavioral Factors in Interactive Training of Text Classifiers. In *NAACL HLT*, 2012.
- [137] H. Sebastian Seung, M. Opper, and Haim Sompolinsky. Query By Committee. In *COLT*, 1992.
- [138] Viktoriia Sharmanska, Novi Quadrianto, and Christoph Lampert. Augmented Attribute Representations. In *ECCV*, 2012.
- [139] Eli Shechtman and Michal Irani. Matching Local Self-Similarities across Images and Videos. In *CVPR*, 2007.
- [140] Jianbo Shi and Jitendra Malik. Normalized Cuts and Image Segmentation. *TPAMI*, 22(8):888–905, 2000.
- [141] Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Constrained Semi-Supervised Learning using Attributes and Comparative Attributes. In *ECCV*, 2012.
- [142] Behjat Siddiquie, Rogerio Feris, and Larry Davis. Image Ranking and Retrieval Based on Multi-Attribute Queries. In *CVPR*, 2011.
- [143] Behjat Siddiquie and Abhinav Gupta. Beyond Active Noun Tagging: Modeling Contextual Interactions for Multi-Class Active Learning. In *CVPR*, 2010.

- [144] John R. Smith, Milind Naphade, and Apostol Natsev. Multimedia Semantic Indexing using Model Vectors. In *ICME*, 2003.
- [145] Alexander Sorokin and David A. Forsyth. Utility Data Annotation with Amazon Mechanical Turk. In *CVPR Workshop on Internet Vision*, 2008.
- [146] Michael Stark, Michael Goesele, and Bernt Schiele. A Shape-based Object Class Model for Knowledge Transfer. In *ICCV*, 2009.
- [147] Raphael Sznitman and Bruno Jedynek. Active Testing for Face Detection and Localization. *TPAMI*, 32(10):1914–1920, 2010.
- [148] Omar Tamuz, Ce Liu, Serge Belongie, Ohad Shamir, and Adam T. Kalai. Adaptively Learning the Crowd Kernel. In *ICML*, 2011.
- [149] Jaime Teevan, Susan Dumais, and Eric Horvitz. Personalizing Search via Automated Analysis of Interests and Activities. In *SIGIR*, 2005.
- [150] Yuandong Tian and Jun Zhu. Learning from Crowds in the Presence of Schools of Thought. In *KDD*, 2012.
- [151] Kinh Tieu and Paul Viola. Boosting Image Retrieval. In *CVPR*, 2000.
- [152] Simon Tong and Edward Chang. Support Vector Machine Active Learning for Image Retrieval. In *ACM Multimedia*, 2001.
- [153] Daniel Tunkelang. Faceted Search. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 2009.

- [154] Koen E. A. van de Sande, Theo Gevers, and Cees G. M. Snoek. Evaluation of Color Descriptors for Object and Scene Recognition. In *CVPR*, 2008.
- [155] Roelof van Zwol, Börkur Sigurbjornsson, Ramu Adapala, Lluís Garcia Pueyo, Abhinav Katiyar, Kaushal Kurapati, Mridul Muralidharan, Sudar Muthu, Vanessa Murdock, Polly Ng, Anand Ramani, Anuj Sahai, Sriram Thiru Sathish, Hari Vasudev, and Upendra Vuyyuru. Faceted Exploration of Image Search Results. In *WWW*, 2010.
- [156] Daniel Vaquero, Rogerio Feris, Duan Tran, Lisa Brown, Arun Hampapur, and Matthew Turk. Attribute-Based People Search in Surveillance Environments. In *WACV*, 2009.
- [157] Sudheendra Vijayanarasimhan and Kristen Grauman. Multi-Level Active Prediction of Useful Image Annotations for Recognition. In *NIPS*, 2008.
- [158] Sudheendra Vijayanarasimhan and Kristen Grauman. What’s It Going to Cost You?: Predicting Effort vs. Informativeness for Multi-Label Image Annotations. In *CVPR*, 2009.
- [159] Sudheendra Vijayanarasimhan and Kristen Grauman. Large-Scale Live Active Learning: Training Object Detectors with Crawled Data and Crowds. In *CVPR*, 2011.



- [160] Sudheendra Vijayanarasimhan, Prateek Jain, and Kristen Grauman. Far-Sighted Active Learning on a Budget for Image and Video Recognition. In *CVPR*, 2010.
- [161] Sudheendra Vijayanarasimhan and Ashish Kapoor. Visual Recognition and Detection Under Bounded Computational Resources. In *CVPR*, 2010.
- [162] Carl Vondrick, Aditya Khosla, Tomasz Malisiewicz, and Antonio Torralba. HOGgles: Visualizing Object Detection Features. In *ICCV*, 2013.
- [163] Catherine Wah and Serge Belongie. Attribute-Based Detection of Unfamiliar Classes with Humans in the Loop. In *CVPR*, 2013.
- [164] Gang Wang and David A. Forsyth. Joint Learning of Visual Attributes, Object Classes and Visual Saliency. In *ICCV*, 2009.
- [165] Gang Wang, David A. Forsyth, and Derek Hoiem. Comparative Object Similarity for Improved Recognition with Few or No Examples. In *CVPR*, 2010.
- [166] Josiah Wang, Katja Markert, and Mark Everingham. Learning Models for Object Recognition from Natural Language Descriptions. In *BMVC*, 2009.
- [167] Xiaogang Wang, Ke Liu, and Xiaoou Tang. Query-Specific Visual Semantic Spaces for Web Image Re-Ranking. In *CVPR*, 2011.

- [168] Xuanhui Wang, Hui Fang, and ChengXiang Zhai. A Study of Methods for Negative Relevance Feedback. In *SIGIR*, 2008.
- [169] Yang Wang and Greg Mori. Max-Margin Hidden Conditional Random Fields for Human Action Recognition. In *CVPR*, 2009.
- [170] Yang Wang and Greg Mori. A Discriminative Latent Model of Object Classes and Attributes. In *ECCV*, 2010.
- [171] Peter Welinder, Steve Branson, Serge Belongie, and Pietro Perona. The Multidimensional Wisdom of Crowds. In *NIPS*, 2010.
- [172] Jacob Whitehill and Javier R. Movellan. Personalized Facial Attractiveness Prediction. In *Automatic Face and Gesture Recognition*, 2008.
- [173] Steven A. Wolfman, Tessa Lau, Pedro Domingos, and Daniel S. Weld. Mixed Initiative Interfaces for Learning Tasks: SMARTedit Talks Back. In *IUI*, 2001.
- [174] Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng. Probability Estimates for Multi-class Classification by Pairwise Coupling. In *Journal of Machine Learning Research*, volume 5, pages 975–1005, December 2004.
- [175] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff Schneider, and Jaime Garbonell. Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorization. In *SDM*, 2010.

- [176] Yan Yan, Romer Rosales, Glenn Fung, and Jennifer G Dy. Active Learning from Crowds. In *ICML*, 2011.
- [177] Jun Yang, Rong Yan, and Alexander G Hauptmann. Adapting SVM Classifiers to Data with Shifted Distributions. In *ICDM Workshops*, 2007.
- [178] Felix Yu, Liangliang Cao, Rogerio Feris, John Smith, and Shih-Fu Chang. Designing Category-Level Attributes for Discriminative Visual Recognition. In *CVPR*, 2013.
- [179] Eric Zavesky and Shih-Fu Chang. Cu-Zero: Embracing the Frontier of Interactive Visual Search for Informed Users. In *MIR*, 2008.
- [180] Cha Zhang and Tsuhan Chen. An Active Learning Framework for Content Based Information Retrieval. *IEEE Transactions on Multimedia*, 2002.
- [181] Hanwang Zhang, Zheng-Jun Zha, Shuicheng Yan, Jingwen Bian, and Tat-Seng Chua. Attribute Feedback. In *ACM Multimedia*, 2012.
- [182] Xiang Sean Zhou and Thomas S. Huang. Relevance Feedback in Image Retrieval: A Comprehensive Review. *Multimedia Systems*, 2003.

## Vita

Adriana Ivanova Kovashka was born in Sofia, Bulgaria. She attended the American College of Sofia, from which she received her high school diploma in 2004. She then attended Pomona College in California, where she majored in Computer Science and Media Studies, with a minor in German. In 2007, she participated in an REU program at Princeton University. In 2008, she began her PhD studies at the University of Texas at Austin, under the guidance and supervision of Professor Kristen Grauman. In 2012, she was an intern at the object detection group at Google Research. Upon receiving her PhD degree in August 2014, she will join the Computer Science Department at the University of Pittsburgh as an Assistant Professor.

Permanent contact: `adriana.kovashka@live.com`

This dissertation was typeset with L<sup>A</sup>T<sub>E</sub>X<sup>†</sup> by the author.

---

<sup>†</sup>L<sup>A</sup>T<sub>E</sub>X is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's T<sub>E</sub>X Program.