LEARNING PERCEPTUAL SIMILARITY FROM CROWDS AND MACHINES

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Michael James Wilber

May 2018

LEARNING PERCEPTUAL SIMILARITY FROM CROWDS AND MACHINES

Michael James Wilber, Ph.D.

Cornell University 2018

How might we teach machine learning systems about what wine tastes like, or how to appreciate the similarities in different kinds of artwork?

On its face, this question seems absurd because these notions of similarity are impossible to characterize in meaningful ways. Our work explores what happens when we can embrace this ambiguity. We use new kinds of semi-supervision to learn abstract, intuitive notions of perceptual similarity when labels or dense similarity measures are not available.

Before we can learn about perceptual similarity, we must first show how to capture intuitive notions of similarity from humans in an efficient and principled way that makes as few assumptions as possible about the data structure. Then, we outline ways to combine expensive human expertise with dense machine kernels to ease the human annotation burden. Finally, we will discuss our work on creating a large-scale dataset of artwork that the research community can use to explore these ideas.

BIOGRAPHICAL SKETCH

Mike grew up in the shadow of the mountains in Colorado Springs, Colorado. During college, he studied under Dr. Terry Boult and Dr. Walter Scheirer at the Vision and Security Technology (VAST) Lab at the University of Colorado, Colorado Springs before receiving his Bachelors of Innovation in Computer Science in 2013. From there, he joined Dr. Serge Belongie's research group at UC San Diego, following his advisor to Cornell Tech in Manhattan to continue his Ph. D. studies.

Mike's hobbies include reading, walking around nature, studying Japanese (though not fluently), and exploring the quiet coffee shops and libraries around the city.

This dissertation is dedicated to all past and future members of the Trans Crossroads Discord emotional support team.

As lighthouses for our community, they provide refuge from the storm, gently pointing the way towards our best selves whenever we run aground.

ACKNOWLEDGEMENTS

Oh heck, where could I possibly begin? I know I wouldn't be where I am without my mentor, Serge Belongie, whose patient guidance was instrumental to my graduate career. I have also had the pleasure of relying on the support and encouragement of many peers. This thesis is based off of joint work with Andreas Veit, Sam Kwak, Chen Fang, Hailin Jin, Aaron Hertzmann, John Collomosse, and Serge Belongie. Chatting with my peers and colleagues and everyone else in the SE(3) research group and their alumni is always a pleasure, but I owe particular gratitude to Andreas Veit, Yin Cui, Omid Poursaeed; the "UCSD Cohort" of Tsung-Yi Lin, Hani Altwaijry, Mohammad Moghimi, Grant Van Horn, Sam Kwak; and former mentors Boris Babenko and Catherine Wah. *("Once a king or queen of SE(3), always a king or queen of SE(3).")* Thanks also to Xiao Ma, Neta Tamir, Laurens van der Maaten, Vicente Malave, and Zack Chase Lipton for insightful discussions, and to Jan Jakeš, Tomas Matera, and Edward Cheng for their software tools that helped us collect grid triplets so quickly.

Additional thanks go to mentors Chen Fang, Aaron Hertzmann, Hailin Jin, James Davis, and Peter Wellinder, who all had a hand in my professional development. I also wish to especially thank mentors Terry Boult and Walter J. Scheirer. Their kindness and gentle mentorship through my undergraduate program launched my career and I cannot possibly thank them enough.

Also, special thanks to my family, Peggy and David, along with my sister Becca, who quietly offered support and encouragement through five years of complaining phone calls and distant airport rides.

Finally, I want to thank my partner Angela, whose kind support and compassion for the world consistently reassure me that I don't have to apologize for being me. ("Rawrawrrawrrr! <3")

As this acknowledgments section spills out onto a second page, I finally give thanks to the Cornell Ithaca computer science department, who provided a welcoming home during my first disorienting confusing year at Cornell.

I want to emphasize that this work would not have been possible without the kindness and generosity of all of these people as well as those too numerous to mention.

In terms of financial sponsorship, my research was partially supported by an NSF Graduate Research Fellowship award (NSF DGE-1144153), the Connected Experiences Lab supported by Oath Inc, a Facebook equipment donation to Cornell University, and Adobe Research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

There is a tension in the field of computer vision. As computer scientists, we primarily work with systems using reproducible, self-contained processes that have quantifiable inputs and discrete outputs. However, these systems are intended to model the real world, a messy unpredictable environment that does not lend itself well to easily-describable formulations. Worse still, there is interest in creating systems intended to mimic the human brain, an incomprehensible black-box system with complicated hidden inner state, inconsistent outputs, and extremely limited introspection capabilities.

It is thought that creating "thinking machines" by modeling the brain is a step toward "strong intelligence," the idea that artificially intelligent systems will one day be able to surpass humans in problem-solving abilities. However, achieving strong intelligence is orthogonal to the related goal of achieving "strong intuition," the unconscious humanistic awareness of the relationships between similar things. This sort of *intuition* is a major source of human creativity and enjoyment. Intuition is how music punks know that Prince music sounds more like Michael Jackson than it does to Avril Lavigne's work, how foodies know that oolong tea tastes more similar to sencha than it does to chai, and how graffiti artists know which styles and personal touches make for aesthetically pleasing artwork.

Unfortunately for the scientist – but very fortunately for the artist – the reasoning behind intuitive judgments are not necessarily expressible in words. Computer vision continues to make steady progress in domains where expert teachers are able to distill and serialize their intuition down to a codified cur-

riculum, but it is less obvious how to serialize our thoughts about, say, what coffee tastes like or why two artists' work is similar. For example, fine-grained species recognition can rely on detailed field guides, centuries of ornithological expertise, and an abundance of discrete labeled training data. However, what can we do when this "bottom-up" approach is not available? How can we learn about the layman's perception of the similarity of two food tastes, for example, without having to rely on expert chefs?

As one first step toward this goal, our general approach is to learn "concept embeddings," vector spaces where distance corresponds with similarity perception. To do this, our work focuses on three primary questions toward the goal of learning strong intuition:

1. What are the quantifiable *unit of similarity,* and how can we efficiently collect this information from large crowds of non-experts?
2. How can we learn similarity kernels on large-scale datasets that are infeasible to densely label by hand?
3. As an example domain, how can these ideas be applied to learn about artwork media and style on a very large scale? What can the limitations of computer vision systems on artwork teach us about the representation gap between the realistic world and the world rendered through an artist's hands?

In Chapter 2, we consider *triplets* as the units of strong intuition. Each triplet (x_1, x_2, x_3) is an assertion of the form "Object x_1 is more similar to object x_2 than object x_3." Triplets are desirable to other forms of annotation because they can be collected from crowds in a straightforward fashion without relying on domain expertise. We show that thousands of these triplet annotations can be col-

lected in a cost-efficient manner using novel user interfaces on crowdsourcing platforms. This lays the foundation for crowd annotation used in later work.

In Chapter 3, we extend this general idea further to learn intuitive similarity at very large scale, combining sparse expensive human expertise with dense, cheap automated kernels. Spaces learned this way require a fraction of the human labeling power than ordinary crowd kernels alone.

Finally, as a bridge back to explicit semantics, we lay the groundwork for exploring artistic intuition in Chapter 4 by collecting a very large-scale collection of digital artwork using a combination of computer vision and efficient crowdsourcing for annotation. We use this dataset to probe how current computer vision systems handle the "representation gap" between photography and stylized artwork.

CHAPTER 2

COST-EFFECTIVE HITS FOR RELATIVE SIMILARITY COMPARISONS

2.1 Introduction

Recently in machine learning [63, 31, 66, 47], there has been a growing interest in collecting human similarity comparisons of the form "Is a more similar to b than to c?" These comparisons are asking humans to provide constraints of the form $d(a,b) < d(a,c)$, where $d(x,y)$ represents some perceptual distance between x and y. We will refer to these constraints as triplets. By collecting these triplets from humans, researchers can learn the structure of a variety of data sets. For example, the authors of [47] were able to learn music genres from triplet comparisons alone with no other annotations. Specifically in computer vision, human similarity comparisons are useful for creating perceptually-based embeddings. In [2], the authors created a two dimensional embedding where one axis represented the brightness of an object, and the other axis represented the glossiness of an object. In this work we focus on creating perceptual embeddings from images of food.

For any set of n points, there are on the order of by n^3 unique triplets. Collecting such a large amount of triplets from crowd workers quickly becomes intractable for larger datasets. For this reason, a few research groups have proposed more intelligent sampling techniques [63, 31]. However, the difficulty of collecting a large number of triplets is also related to the time and monetary cost of collecting data from humans. To investigate this relationship more closely, we chose to study a triplet human intelligence task (HIT). In this work we provide a better understanding of how the HIT design affects not only the time and cost

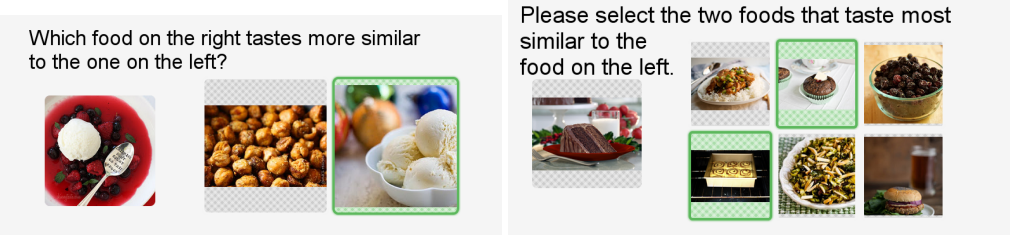

Figure 2.1: Questions of the form "Is object a more similar to b than to c?" have been shown to be a useful way of collecting similarity comparisons from crowd workers. Traditionally these comparsions, or triplets, would be collected with a UI shown at the left. In this work we collect triplets using a grid of images and ask the user to select the two most similar tasting foods to the anchor. The grid UI, right, allows us to collect 8 triplets whereas the triplet UI, left, only yeilds a single triplet.

of collecting triplets, but also the quality of the embedding, which is usually the researcher's primary concern.

Traditionally, an MTurk task designed to collect triplets would show crowd workers three images, labeled a, b, c. The worker is then asked to select either image b or image c, whichever looks more similar to image a. See the top of Fig. 2.1 for an example. Although this is the most direct design to collect triplets, it is potentially inefficient. Instead, we chose to investigate triplets collected from a grid of images. In the grid format, a probe image—analogous to image "a" in the triplet representation—is shown next to a grid of n images. The crowd worker is then asked to choose the k most similar images from the grid. This layout allows us to collect k images that are more similar to the probe image than the remaining $n - k$ images, yielding $k(n - k)$ triplets with one screen to the user. We can change the number of triplets per grid answer by varying n and k, but this also affects the amount of effort a crowd worker must exert to answer the question. We are not the first to realize that a grid is more efficient for collecting triplets—such techniques were also used by [68, 63]—but we believe we are the first to investigate more thoroughly the effectiveness of triplets collected with

a grid. This is important because previous authors acknowledge neither the efficiency gain nor the potential drawbacks of the grid triplets they rely on.

This paper outlines several UI modifications that allow researchers to multiply the number of triplets collected per screen for perceptual similarity learning. We show that simple changes to the crowdsourcing UI—*instead* of fundamental changes to the algorithm — can lead to much higher quality embeddings. In our case, using our grid format allows us to collect several triplet comparisons per screen. This leads to much faster convergence than asking one triplet question at a time. Researchers with tight deadlines can create reasonable embeddings with off-the-shelf algorithms and a low crowdsourcing budget by following our guidelines.

Our contributions are:

- A set of guidelines to use when collecting similarity embeddings, with insights on how to manage the trade-off between user burden, embedding quality, and cost;
- A series of synthetic and human-powered experiments that prove our methods' effectiveness;
- Evidence that each individual triplet sampled with a grid may capture less information than a uniformly random triplet, but that their quantity outweighs the potential quality decrease;
- A dataset of 100 food images, ingredient annotations, and roughly 39% of the triplet comparisons that describe it, to be made available upon publication.

2.2 Related Work

Perceptual similarity embeddings are useful for many tasks within the field, such as metric learning [21], image search/exploration [20], learning semantic clusters [24], and finding similar musical genres and artists [66, 47]. Our work is useful to authors who wish to collect data to create such embeddings. The common idea behind all of this work is that these authors use triplets to collect their embeddings.

In our work, we collect human similarity measurements of images in the form of triplets. The authors of [27] proposed an algorithm for collecting triplets from humans as well. However in [27], the triplets that were collected did not have a probe image. because they formulated the question differently [75] focuses on estimating user preferences from crowd sourced similarity comparisons. However [75] uses pairwise comparisons rather than triplets.

Our work bears much similarity to Crowd Kernel Learning [63] and Active MDS [31]. These algorithms focus on collecting triplets one at a time, but sampling the *best* triplets first. The idea behind these systems is that the bulk of the information in the embedding can be captured within a very small number of triplets, since most triplets convey redundant information. For instance, Crowd Kernel Learning [63] considers each triplet individually, modeling the information gain learned from that triplet as a probability distribution over embedding space. Active MDS [31] consider a set of triplets as a partial ranking with respect to each object in the embedding, placing geometric constraints on the locations where each point may lie. In our work we focus on altering UI design to improve speed and quality of triplet collection.

Figure 2.2: **Top:** An example cuisine embedding, collected with our 16-choose-4 grid UI strategy. This embedding cost us $5.10 to collect and used 408 screens, but yielded 19,199 triplets. It shows good clustering behavior with desserts gathered into the top left. The meats are close to each other, as are the salads. **Bottom:** An embedding with 408 random triplets. This embedding also cost $5.10 to collect, but the result is much dirtier, with worse separation and less structure. Salads are strewn about the right half of the embedding and a steak lies within the dessert area. From our experiments, we know that an embedding of such low quality would have cost us less than $0.10 to collect using our grid strategy.

2.3 Method

Instead of asking "Is a more similar to b or c?", we present humans with a probe image and ask "Mark k images that are most similar to the probe," as in Fig. 2.1. This way, with a grid of size n, a human can generate $k \cdot (n - k)$ triplets per task unit. This kind of query allows researchers to collect more triplets with a single screen. It allows crowd workers to avoid having to wait for multiple screens to load, especially in cases where one or more of the images in the queried triplets

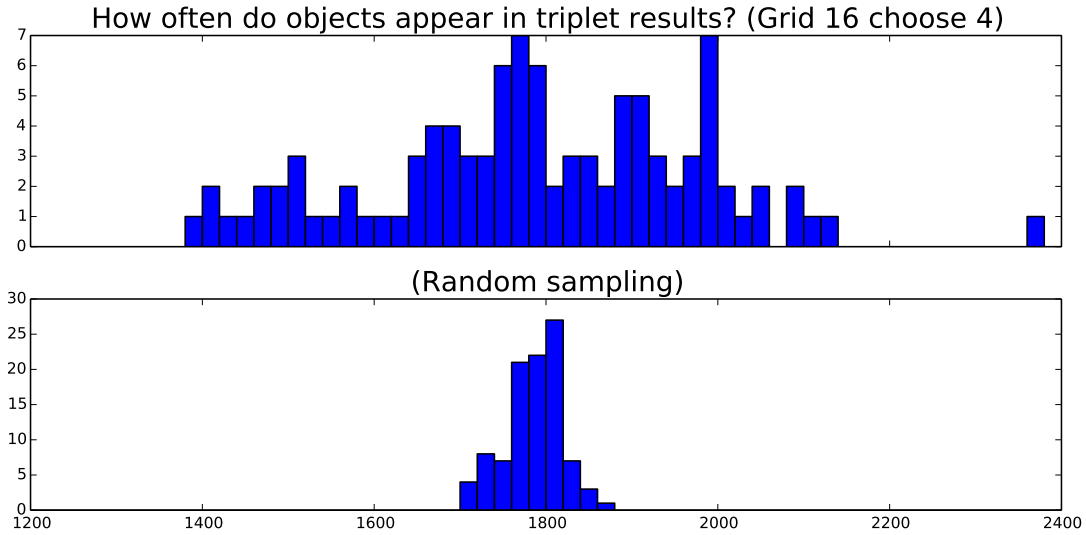

Figure 2.3: **Random triplets have a different distribution than grid triplets.** The top histogram shows the occurrences of each object within human answers for "Grid 16 choose 4" triplets. The bottom histogram shows a histogram of sampling random triplets individually. 59520 triplets were collected for both histograms. Each object occurs in our answers about $\hat{\mu} = 1785$ times, but the variation when using grid triplets (top) is much wider ($\hat{\sigma} \approx 187.0$) than the variation when sampling triplets uniformly (bottom, $\hat{\sigma} = 35.5$). This effect is not recognized in the literature by authors who use grids to collect triplets. We study its impact in our experiments.

do not change. This also allows crowd workers to benefit from the parallelism in the low-level human visual system [72]. Since many of these observations involve human issues, we conclude that the right way of measuring embedding quality is with respect to *human cost* rather than the number of triplets. This *human cost* is related to the time it takes crowd workers to complete a task *and* the pay rate of a completed task. Some authors [68, 63] already incorporate these ideas into their work but do not quantify the improvement. Our goal is to formalize their intuitive notions into hard guidelines.

It is important to note that the *distribution of grid triplets is not uniformly random*, even when the grid entries are selected randomly and even with perfect answers. To our knowledge, no authors that use grids acknowledge this poten-

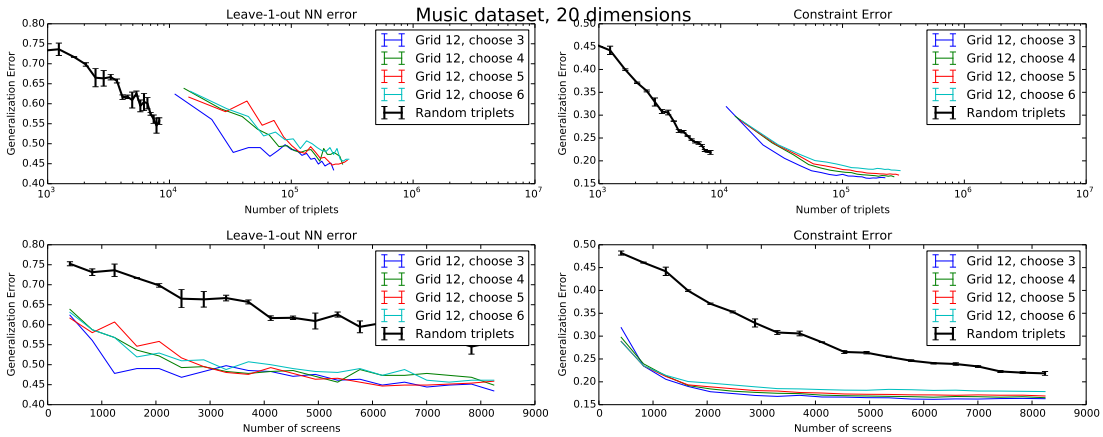

Figure 2.4: Over the course of a synthetic experiment, we collect triplets, either randomly one at a time (thick black line) or in batches using our grid UI (colored lines). When the embedding quality is viewed as the number of triplets gathered (top two graphs), it appears that sampling random triplets one at a time yields a better embedding. However, when viewed as a function of human effort, grid triplets create embeddings that converge much faster than individually sampled triplets. Here, quantity outweighs quality as measured by Leave-One-Out NN Error (left graphs) and Triplet Generalization Error (right graphs). See text for details.

tial bias even though it deteriorates each triplet's quality, as we will show in our experiments. Figure 2.3 shows a histogram of how many times each object occurs in our triplet answers. When using grid sampling, some objects can occur far more often than others, suggesting that the quality of certain objects' placement within the recovered embedding may be better than others. The effect is less pronounced in random triplets, where objects appear with roughly equal frequency. This observation is important to keep in mind because the unequal distribution influences the result.

2.4 Synthetic Experiments

We aimed to answer two questions: *Are the triplets acquired from a grid of lower quality than triplets acquired one by one?* Second, *even if grid triplets are lower quality, does their quantity outweigh that effect?* To find out, we ran synthetic "Mechanical Turk-like" experiments on synthetic workers. For each question, we show a probe and a grid of n objects. The synthetic workers use Euclidean distance within a groundtruth embedding to choose k grid choices that are most similar to the probe. As a baseline, we randomly sample triplet comparisons from the groundtruth embedding using the same Euclidean distance metric. After collecting the test triplets,we build a query embedding with t-STE [66] and compare this embedding to the groundtruth. This way, we can measure the quality of our embedding with respect to the total amount of human effort, which is the number of worker tasks. This is not a perfect proxy for human behavior, but it does let us validate our approach, and should be considered in conjunction with the actual human experiments that are described later.

Datasets. We evaluated our UI paradigm on three datasets. First, we used MNIST1k, a handwritten digit dataset containing 1,000 random digits across 10 classes. To generate groundtruth comparison triplets, we use Euclidean distance between feature vectors. Second, we use the music similarity dataset from [66] as a point of comparison. This set contains 9,107 human-collected triplets for 412 artists. Finally, we present results on a subset of LFW [29], the Labeled Faces in the Wild dataset. We considered identities that have between 32 and 77 images in the set, using the face attribute vectors extracted by [39]. This leaves us with a total of 938 73-dimensional feature vectors from 20 identities. To generate groundtruth triplets, we again considered Euclidean distance. These

three datasets provide us with a healthy balance of synthetic and real-world nonvectorial data.

Metrics. Our goal is not to build a competitive face or written digit recognizer; rather, we simply wish to evaluate the quality of a perceptual embedding constructed with the help of synthetic workers. To do this, we evaluate each embedding's quality using two metrics from [66]: Triplet Generalization Error, which counts the fraction of the groundtruth embedding's triplet constraints that are violated by the recovered embedding; and Leave-One-Out Nearest Neighbor error, which measures the percentage of points that share a category label with their closest neighbor within the recovered embedding. As pointed out by [66], these metrics measure different things: Triplet Generalization Error measures the triplet generator UI's ability to generalize to unseen constraints, while NN Leave-One-Out error reveals how well the embedding models the (hidden) human perceptual similarity distance function. We use these metrics to test the impact that different UIs have on embedding quality.

Results. Across all three datasets, our experiments show that even though triplets acquired via the grid converge faster than random triplets, each individual grid triplet *is of lower quality* than an individual random triplet. Figure 2.4 shows how the music dataset embedding quality converges with respect to the number of triplets. If triplets are sampled one at a time (top two graphs), random triplets converge much faster on both quality metrics than triplets acquired via grid questions. However, this metric does not reveal the full story because grid triplets can acquire several triplets at once. When viewed with respect to the number of *screens* (human task units), as in the bottom two graphs in Figure 2.4, we now see that the grid triplets can converge far faster than random

with respect to the total amount of human work. This leads us to conclude that "quality of the embedding *wrt.* number of triplets" is the wrong metric to optimize because framing the question in terms of triplets gives researchers the wrong idea about how fast their embeddings converge. A researcher who only considers the inferior performance of grid triplets on the "per-triplet" metric will prefer sampling triplets individually, but they could achieve much better accuracy using grid sampling even in spite of the reduced quality of each individual triplet, and as we shall see in our human experiments, this translates into decreased cost for the researcher. In other words, efficient collection UIs are better than random sampling, even though each triplet gathered using such UIs does not contain as much information.

Why does this happen? In all cases, the 12 images within the grid were chosen randomly; intuitively, we expect a uniform distribution of triplets. However, because certain objects are more likely than others to be within each grid's "Near" set, certain objects will appear in the triplet more often than others. This leads to a nonuniform distribution of correct triplets, as shown in Fig. 2.3. Here, we can see that the non-uniformity creates a difference in performance.

The other two datasets—MNIST and Face—show very similar results so we do not report them here. In all cases, any size of grid UI outperforms random selection. However, we do see a small spread of quality across different grid sizes. As in the music dataset, the error is lowest when we force our synthetic workers to select 3 close images out of 12 as opposed to selecting the 4, 5, or 6 closest images. This difference is more pronounced in the "Leave-One-Out NN" metric. This could be because selecting the 3 closest images allows the metric to be more precise about that image's location in the embedding since

it is compared to fewer neighbors. Our synthetic workers always give perfect answers; we do not expect imperfect humans to reflect this effect.

Figure 2.5: Example images from our dataset. The images in our dataset span a wide range of foods and imaging conditions. The dataset as well as the collected triplets will be made available upon publication.

2.5 Human Experiments

These synthetic experiments validate our approach, but they have several problems. In particular, there is no reason why humans would behave similarly to a proxy oracle as described above. Further, we must also consider the effort of our workers, both in terms of the time it takes to complete each task and how much money they can make per hour—metrics that are impossible to gather via synthetic means. To verify that these approaches build better embeddings even when humans provide inconsistent triplets, we ran Mechanical Turk experiments on a set of 100 food images sourced from Yummly recipes with no groundtruth. The images were filtered so that each image contained roughly one entree. For example, we avoided images of sandwiches with soups. Example images are shown in Fig. 2.5. For each experiment, we allocated the same amount of money for each hit, allowing us to quantify embedding quality with

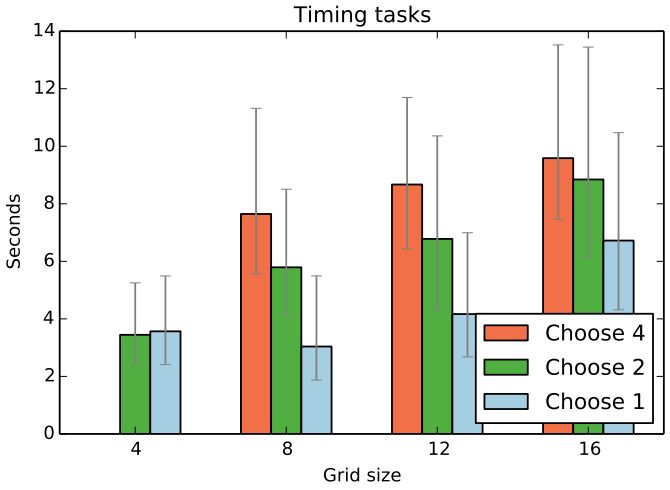

Figure 2.6: We show the median time that it takes a human to answer one grid. The time per each task increases with a higher grid size (more time spent looking at the results) and with a higher required number of near answers (which means more clicks per task). Error bars are 25 and 75-percentile.

respect to cost. Upon publication, the dataset as well as the collected triplets will be available for download.

Design. For each task, we show a random probe and a grid of n random foods. We ask the user to select the k objects that "taste most similar" to the probe. We varied n across $(4, 8, 12, 16)$ and varied k across $(1, 2, 4)$. We ran three independent repetitions of each experiment. We paid \$0.10 per HIT, which includes 8 usable grid screens and 2 catch trials. To evaluate the quality of the embedding returned by each grid size, we use the same "Triplet Generalization Error" as in our synthetic experiments: we gather all triplets from all grid sizes and construct a reference embedding via t-STE. Then, to evaluate a set of triplets, we construct a target embedding, and count how many of the reference embedding's constraints are violated by the target embedding. Varying the number of HITs shows how fast the embedding's quality converges.

Baseline. Since we wish to show that grid triplets produce better-quality em-

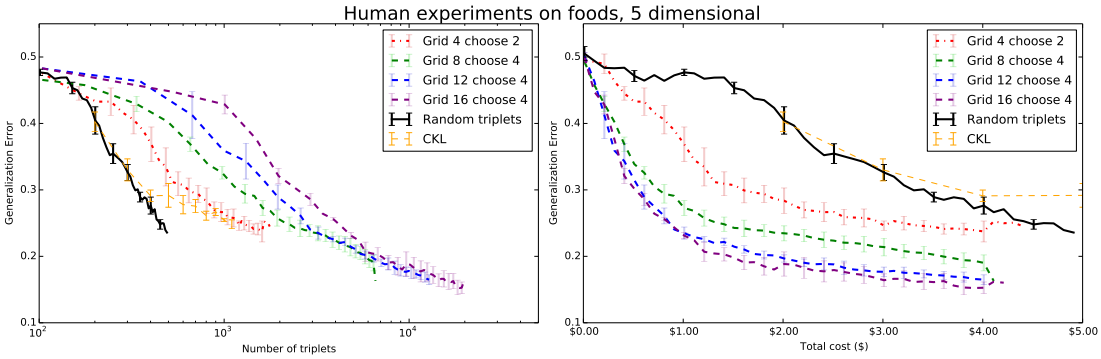

Figure 2.7: Results of our human experiments on the food dataset. Left graph: Triplet generalization error when viewed with respect to the total number of triplets. Right: The same metric when viewed with respect to the total cost (to us) of constructing each embedding. The left graph implies that a randomly-sampled embedding appears to converge faster. However, when quality is viewed with respect to cost, we find that an embedding generated using a 16-choose-4 grid cost $0.75, while an embedding with random triplets of similar quality costs $5.00. It is clear that the grid UI saves money; in this case, by over a factor of 6.

beddings at the same cost as random triplets, we should collect random (a, b, c) comparisons from our crowd workers for comparison. Unfortunately, collecting all comparisons one at a time is infeasible (see our "Cost" results below), so instead, we construct a groundtruth embedding from all grid triplets and uniformly sample random constraints from the embedding. This is unlikely to lead to much bias because we were able to collect 39% of the possible unique triplets, meaning that t-STE only has to generalize to constraints that are likely to be redundant. All evaluations are performed relative to this reference embedding.

2.5.1 Results

Two example embeddings are shown in Fig. 2.2.

Cost. Across all experiments, we collected 14,088 grids, yielding 189,519

unique triplets. Collecting this data cost us $158.30, but sampling this many random triplets one at a time would have cost us $2,627.63, which is far outside our budget[1]. If we had used the 16-choose-4 grid strategy (which yields 48 triplets per grid), we would be able to sample all unique triplets for about $140—a feat that would cost us $6737.50 by sampling one at a time.

Grid n choose k		Error at $1	Time/screen (s)	Wages ($/hr)
n: 4,	k: 1	0.468	3.57	**$10.09**
	k: 2	0.369	3.45	**$10.45**
n: 8,	k: 1	0.400	3.04	**$11.85**
	k: 2	0.311	5.79	**$6.22**
	k: 4	0.273	7.65	$4.71
n: 12,	k: 1	0.406	4.17	**$8.64**
	k: 2	0.294	6.78	$5.31
	k: 4	0.235	8.67	$4.15
n: 16,	k: 1	0.413	6.72	$5.36
	k: 2	0.278	8.84	$4.07
	k: 4	0.231	9.59	$3.76
Random		0.477	–	–
CKL		0.403	–	–

Table 2.1: Results of our actual Mechanical Turk experiments. We ask workers to choose the k most similar objects from a grid of n images. We invest $1 worth of questions, giving us 100 grid selections. When n and k are large, each answer yields more triplets. Large grids require more time to complete, but many of our tasks (bold) still pay a respectable wage of more than $6 per hour.

Quality. As we spend more money, we collect more triplets, allowing t-STE to do a better job generalizing to unseen redundant constraints. All embeddings converge to lower error when given more triplets, but this convergence is not monotonic because humans are fallible and there is randomness in the embedding construction. See Fig. 2.7 for a graphical comparison of grids with size 4,8,12, and 16. When viewed with respect to the number of triplets, random triplets again come out ahead; but when viewed with respect to cost, the largest

[1]There are $100 \cdot 99 \cdot 98/2 = 485,100$ possible unique triplets and each triplet answer would cost one cent. We additionally need to allocate 10% to Amazon's cut and 20% of our tasks are devoted to catch trials.

grid converges more quickly than others, and even the smallest grid handily outperforms random triplet sampling.

This time, we observe a large separation between the performance of various grid sizes. Grid 16-choose-4, which yields $4 \cdot 12 = 48$ triplets per answer, uniformly outperforms the rest, with Grid 12-choose-4 (at $4 \cdot 8 = 32$ triplets per answer) close behind. Both of these outperform 8-choose-4 (16 triplets/answer) and 4-choose-2 (4 triplets/answer).

We also compare our performance with the adaptive triplet sampling strategy of [63]. CKL picks triplets one-at-a-time but attempts to select the best triplet possible to ask by maximizing the information gain from each answer. In our experiments, it did not outperform random sampling; further analysis will be future work.

Though catch trials comprised 20% of the grid answers we collected, we found that the results were generally of such high quality that no filtering or qualification was required.

Time. Fig. 2.6 shows how fast each human takes to answer one grid question. Our smallest task was completed in 3.5 seconds ( ), but even our largest grid (16 choose 4) can be completed in less than 10 seconds. Times varies widely between workers: our fastest worker answered 800 questions in an average of 2.1 seconds per grid task for 8-choose-1 grids.

Worker Satisfaction. At our standard 1¢-per-grid/$0.10-per-HIT rate, our workers are able to make a respectable income, shown in Tab. 2.1. The smallest tasks net more than $10/hour by median, but even our largest task allows half of our workers to make $3.76 for every hour they spend. If the fastest, most

skilled worker sustained their average pace in 8-choose-1 grids, they could earn over $17 per hour.

Since there is a trade-off between grid size and worker income, it is important to consider just how far we can push our workers without stepping over the acceptable boundaries. Across all of our experiments, we received no complaints, and our tasks were featured on multiple HIT aggregators including Reddit's `HitsWorthTurkingFor` subreddit and the "TurkerNation" forums as examples of bountiful HITs. Our workers did not feel exploited.

According to the `HitsWorthTurkingFor` FAQ [2], "the general rule of thumb ... is a minimum of $6/hour." Though HITs below this amount may be completed, the best workers may pass for more lucrative HITs. Being featured in forums such as `HitsWorthTurkingFor` gave us an advantage since our hit was visible to a very large audience of potential skilled turkers. Though high payouts mean higher cost, in our case, the benefit outweighed the drawback.

2.6 Guidelines and conclusion

Throughout this paper, we have shown that taking advantage of simple batch UI tricks can save researchers significant amounts of money when gathering crowdsourced perceptual similarity data. Our recommendations can be summarized as follows:

- Rather than collecting comparisons one-at-a-time, researchers **should use a grid** to sample comparisons in batch, or should use some other UI

[2] http://reddit.com/r/HITsWorthTurkingFor/wiki/index

paradigm appropriate to their task. However, researchers should not assume that such "batch" comparisons are of identical quality to uniformly random sampling—this is a trade-off that should be considered.

- If cost is an issue, researchers should **quantify their results with respect to dollars spent**. We found that using our simple UI paradigm can creates embeddings of higher quality than those created using algorithms that pick the best triplet one-at-a-time.
- Researchers should **continuously monitor the human effort of their tasks,** so that they can calculate an appropriate target wage and stand a better chance of being featured on "Good HIT" lists and be seen by more skilled Turkers.
- When using grids to collect triplets, researchers should **consider the trade-off between size and effort.** Consider that an n-choose-k grid can yield

$$k(n-k) \tag{2.1}$$

triplets per answer. Since this has a global maximum at $n = 2k$, one appropriate strategy is to select the largest n that yields a wage of \$6/hour and set k equal to $n/2$.

There are several opportunities for future work. First, we should better quantify the relationship between n, k, and task completion time to build a more accurate model of human performance. Second, we should continue investigating triplet sampling algorithms such as "CKL" as there may be opportunities to adaptively select grids to converge faster than random, giving us advantages of both strategies.

2.7 Acknowledgments

We especially thank Jan Jakeš, Tomas Matera, and Edward Cheng for their software tools that helped us collect grid triplets so quickly. We also thank Vicente Malave for helpful discussions. This work was partially supported by an NSF Graduate Research Fellowship award (NSF DGE-1144153, Author 1) and a Google Focused Research award (Author 3).

CHAPTER 3

LEARNING CONCEPT EMBEDDINGS WITH COMBINED HUMAN-MACHINE EXPERTISE

3.1 Introduction

Supervised learning tasks form the backbone of many state-of-the-art computer vision applications. They help researchers classify, localize, and characterize actions and objects. However, if the researcher's goal is instead to interactively explore the latent structure of a dataset, discover novel categories, or find labeling mistakes, it is unclear what kind of supervision to use. Sometimes the data does not fall into well-defined taxonomic categories, or perhaps it is simply too expensive to collect labels for every object. Sometimes the expert wishes to capture a *concept*—some intuitive constraint that they cannot articulate—about how the data should be structured, but does not have the time to specify this concept formally. If we wish to build models that capture concepts, we need a new approach.

Our overall goal is to generate a *concept embedding*. Distances within this space should correspond with a human's intuitive idea of how similar two objects are. Many researchers use similar embeddings to enhance the performance of classifiers [61, 11, 68], build retrieval systems [66, 47], and create visualizations that help experts better understand high-dimensional spaces [15, 14].

Concepts cannot always be inferred from appearance. Within the past few years, huge research advances have begun to produce systems that are excellent at comparing images based on visual cues. For example, one can imagine build-

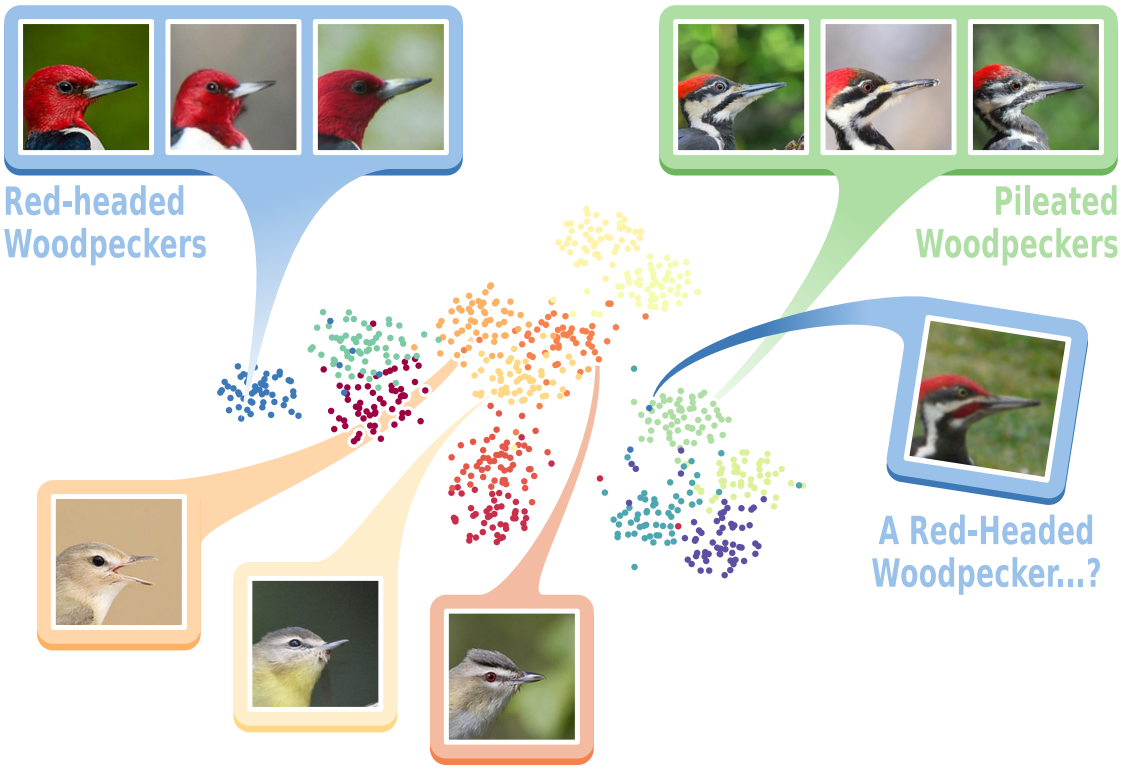

Figure 3.1: Our SNaCK embeddings capture human expertise with the help of machine similarity kernels. For example, an expert can use this concept embedding of a subset of CUB-200 to quickly find labeling mistakes. Red-headed Woodpeckers are visually dissimilar to Pileated Woodpeckers, but SNaCK moved a Red-headed Woodpecker into the Pileated Woodpecker cluster because of its appearance. **This is probably a labeling mistake in CUB-200, and this SNaCK embedding helped us discover it.** The cluster of three visually similar vireo species in the embedding center may be another good place to look for label problems.

ing a CNN to compare food dishes based solely on their appearance. However, if the concept we wish to capture is similarity in taste, the task becomes harder. Although taste and appearance are often correlated, any poor diner who has confused guacamole and wasabi knows that foods that taste very different may look deceptively similar because the strongest visual cues may not be reliable. This particular taste difference is difficult to capture without expert guidance. Similarly, when classifying birds, the goal is often not to group similar-looking birds together, but to group birds of the same species together. Experts know

that appearance is important for this classification task, but there are often large visual differences between the appearance of male and female birds of the same species or between juveniles and adults. In these cases, domain-specific expertise can greatly improve the resulting embedding.

Expert annotations can be expensive to collect. In order to capture abstract concepts known only by humans, the expert must provide *hints* [1] to help guide the learning process. Unfortunately, asking experts to exhaustively and authoritatively annotate the dataset is not always possible [4]. Further, hints are most useful when they are task-specific [14]: if the user wishes to discover some relationship that is not apparent between objects, they should be able to specify whatever hints they feel would best capture those constraints. Previous work that uses perceptual annotations [66, 71] note that collecting all hints based on relative similarity comparisons can take quadratic or cubic cost. Hiring actual domain experts is often out of the question, and even crowdsourcing websites such as Mechanical Turk can be prohibitively expensive.

It seems reasonable that one can use machine kernels to speed up the process of collecting hints. In this work, we show how to overcome the inherent human scalability problems by using human hints to refine a concept embedding generated by an automatic similarity kernel. Our main contributions are as follows:

- We present a **novel algorithm**, "*SNE-and-Crowd-Kernel Embedding*" (SNaCK), that combines expert triplet hints with machine assistance to efficiently generate concept embeddings;
- We show how to use our SNaCK embeddings for tasks such as visualization, concept labeling, and perceptual organization, and show that SNaCK

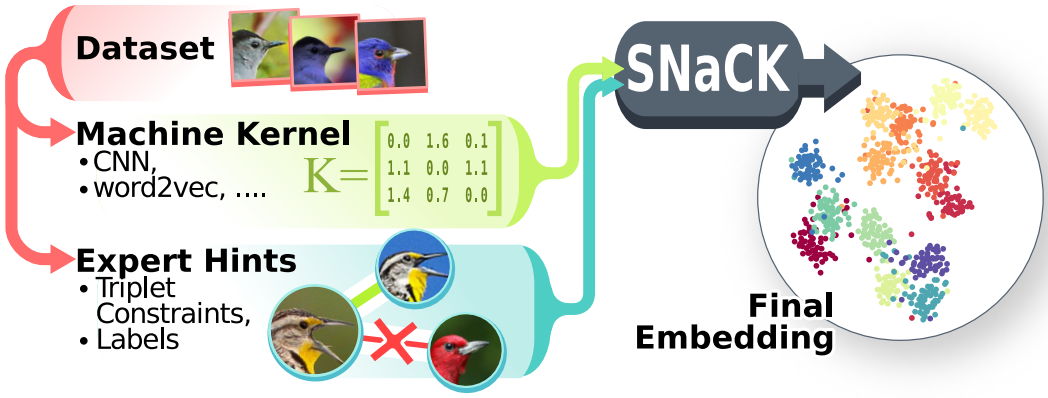

Figure 3.2: Overview of our SNE-and-Crowd-Kernel ("SNaCK") embedding method. As input, SNaCK accepts a dataset of objects, a similarity kernel K, and a set of expert constraints in the form of "Object i should be closer to j than it is to k", which may be inferred from crowdsourcing or label information. The output is a low-dimensional concept embedding that satisfies the expert hints while preserving the structure of K.

embeddings are competitive with the state of the art in these tasks.

We also present the following minor contributions:

- A dataset of 950,000 crowdsourced perceptual similarity annotations on 10,000 food dishes from Yummly;
- A deep-learned food classifier that greatly improves upon the previous state-of-the-art performance on the Food-101 dataset [9];
- A proof that two perceptual embedding algorithms in common use, CKL and t-STE, are equivalent for the common 2D case for certain parameter settings. To our knowledge, this connection has never been acknowledged or explored before.

3.2 Background and related work

Perceptual embeddings. Our work builds upon a large body of existing perceptual embedding literature. Notably, our method combines aspects of both *t-Distributed Stochastic Neighbor Embedding* (t-SNE, from [65]) and *Stochastic Triplet Embedding* (t-STE, from [66]). The objective and motivation behind these approaches are fundamentally different: t-SNE creates a low-dimensional visualization using an automatic kernel from a higher-dimensional space and t-STE generates an embedding from scratch that satisfies as many human-provided similarity constraints as possible. Nevertheless, we show that they are complementary. Interestingly, there is a strong mathematical similarity between t-STE and the *Crowd Kernel Learning* (CKL) method described in [63]; in fact, in the supplementary material, we show that *CKL and t-STE are equivalent* for certain parameter choices. To our knowledge, this connection has not been explored before.

Triplet constraints and other kinds of hints. In our work, we use *triplet constraints*, where the crowd or the expert provides tuples of the form (i, j, k) to indicate that object i seems more similar to object j than i does to object k. We take these constraints to mean that object i should thus be *closer* to object j than i is to k in the desired concept embedding. These relative comparisons allow the expert to directly specify perceptual constraints about objects. When compared to other forms of supervision, triplets are one of the most flexible options in practical use because they do not rely on *a priori* knowledge, are invariant to scale, and are stable between and within subjects. Consider other forms of supervision: placing objects into *category labels* may not map to the abstract concepts the expert wishes to capture and it requires the entire taxonomy

to be known up-front. Even with unlimited time and a patient expert, the label results may be subject to scrutiny: one human expert solving the ImageNet Large Scale Visual Recognition Challenge [59] took approximately a minute to label each image and still made 5.1% error. The CUB-200 [67] dataset also has labeling errors, which we will show in Sec. 3.4.1.

Pairwise similarity judgments are another common form of supervision, but they have own problems. The classic 7-point Likert scale induces quantization into the metric and may not be reliable between people. Several researchers [49, 35, 15] note that methods based on triplet comparisons are more stable than such pairwise measures. In an experiment comparing the speed and effectiveness of pairwise, triplet, and spatial arrangement embeddings, [15] found that triplet comparisons yield the least variance of human perceptual similarity judgments than other methods, though triplet tasks also took humans the longest to complete. One disadvantage of triplet constraints is that triplet embeddings require at least $O(n^3)$ triplet constraints to be uniquely specified [36], even though many triplets are strongly correlated and do not contribute much to the overall structure [61]. This is why we propose using a machine vision system to do most of the heavy lifting and reduce the number of required triplet constraints.

Incorporating human judgments in automatic systems. Of course, we are not the first researchers to show the benefits of combining human and machine expertise. For example, [11, 68] build a classification system by bringing humans "into the loop" at runtime. Other work allows humans to specify an attribute relationship to influence the label training [7]. These approaches are most useful when classification is the end goal rather than visualization or per-

ceptual organization. Another branch of work starts from an automatically-generated distance matrix and uses human constraints to further refine the recovered clustering or distance metric, typically by asking the human to provide pairwise "Must-link" or "Must-not-link" constraints [46, 74, 76, 64, 77]. In some works, the human can provide an attribute explanation for their choice [40]. In Sec. 3.4.1, we show that our approach is competitive with many of these constrained clustering algorithms in a semi-supervised labeling task.

Other particularly relevant contributions re-cast t-STE as a multiple metric learning problem [79]. Here, the humans are asked to evaluate multiple aspects of objects' similarity (eg. similarity of different parts), and the final embedding is learned to jointly satisfy as many aspects as possible. Similarly, [3] learns multiple maps from a single set of triplet questions. Our work is similar in spirit, but our focus on jointly learning both human and machine-judged similarity rather than just multiple aspects of human similarity sets us apart from these works and others such as [24], which focus on creating more efficient user interfaces to gather data from crowdsourcing without using machine vision to accelerate the process.

Embeddings from deep learning and Siamese networks. Finally, an interesting branch of work revolves around teaching CNNs to satisfy triplet questions as part of the overall pipeline [69, 73]. One method based on this approach currently holds the state-of-the-art accuracy on the LFW face verification challenge [61]. Methods like this are very appealing if one wishes to build a classifier. Other methods [25] train Siamese networks on pairwise distance matrices to output the embedding directly. Though our work does use deep learning as part of our pipeline, deep learning is not necessary for our approach.

3.3 "SNE-and-Crowd-Kernel" (SNaCK) embeddings

Our hybrid embedding algorithm, *SNE-and-Crowd-Kernel* (SNaCK), jointly optimizes the objective functions of two different low-dimensional embedding algorithms.[1] The first algorithm, t-SNE (*t-Distributed Stochastic Neighbor Embedding* [65]), uses a distance matrix to construct a low-dimensional embedding. Its goal is to ensure that objects which are close in the original high-dimensional space are also close in the low-dimensional output without constraining points that are far in the original space. The second method, t-STE *(Stochastic Triplet Embedding* [66]), allows experts to supply triplet constraints that draw from their domain knowledge and task-specific hints. We will show that this surprisingly simple joint optimization can capture the benefits of both objectives. See Fig. 3.2 for an overview.

3.3.1 Formulation

Consider N objects. We wish to produce a d-dimensional embedding $Y \in \mathcal{R}^{N \times d}$. Let $K \in \mathcal{R}^{N \times N}$ be a distance matrix, and let $T = \{t_1, \ldots, t_M\}$ be a set of triplet constraints. Each constraint $t_\ell = (i, j, k)$ implies that in the final embedding, object i should be closer to object j than it is to k, meaning $\|y_i - y_j\|^2 \leq \|y_i - y_k\|^2$. According to [65], the loss function for t-SNE can be interpreted as finding the low-dimensional distribution of points that maximizes the information gain from the original high-dimensional space.

$$C_{tSNE} = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}, \tag{3.1}$$

[1] Our code is available on the companion website, http://vision.cornell.edu/se3/projects/concept-embeddings

where

$$p_{j|i} = \frac{\exp(-K_{ij}^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-K_{ik}^2/2\sigma_i^2)} \tag{3.2}$$

$$p_{ij} = \frac{1}{2N}(p_{j|i} + p_{i|j}) \tag{3.3}$$

$$q_{ij} = \frac{(1+||y_i - y_j||^2)^{-1}}{\sum_{k\neq l}(1+||y_k - y_l||^2)^{-1}} \tag{3.4}$$

and σ_i is chosen to satisfy certain perplexity constraints.

The loss function for t-STE, given in [66], can be interpreted as the joint probability of independently satisfying all triplet constraints. It is defined as

$$C_{tSTE} = \sum_{(i,j,k)\in T} \log p_{(i,j,k)}^{tSTE}, \tag{3.5}$$

where

$$p_{(i,j,k)}^{tSTE} = \frac{\left(1+\frac{||y_i-y_j||^2}{\alpha}\right)^{-\frac{1+\alpha}{2}}}{\left(1+\frac{||y_i-y_j||^2}{\alpha}\right)^{-\frac{1+\alpha}{2}} + \left(1+\frac{||y_i-y_k||^2}{\alpha}\right)^{-\frac{1+\alpha}{2}}} \tag{3.6}$$

Interestingly, when $\alpha = 1$ (as suggested in [66] for two-dimensional visualizations), C_{tSTE} becomes a special case of the cost function C_{CKL} from [63] for certain parameter choices. We explore this relationship in the supplementary material. Because they are equivalent, we use C_{tSTE} in our cost function, defined as

$$C_{SNaCK} = \lambda \cdot C_{tSTE} \; + \; (1-\lambda)\cdot C_{tSNE} \tag{3.7}$$

To optimize this cost, we use gradient descent on $\frac{\partial C_{SNaCK}}{\partial Y}$. Our implementation derives from the t-SNE implementation in `scikit-learn`, so we inherit their optimization strategy. In particular, we use t-SNE's early exaggeration [65] heuristic for 100 iterations and then continue until the 300th iteration.

The λ parameter specifies the relative contribution of the machine-computed kernel and the human-provided triplet constraints on the final embedding. For

each experiment, we pick λ up front such that the norm of $\frac{\delta C_{tSTE}}{\delta Y}$ is approximately equal to $\frac{\delta C_{tSNE}}{\delta Y}$ in cross validation.

3.3.2 SNaCK example: MNIST

To briefly illustrate why this formulation is better than t-STE or t-SNE alone, Fig. 3.3 shows a toy example on MNIST data. In this example, suppose the expert wishes to capture the concept of primality by partitioning the dataset into prime numbers $\{2, 3, 5, 7\}$, composite numbers $\{4, 6, 8, 9\}$ and $\{0, 1\}$. Also, for the purpose of this simple example, assume that rather than labeling the digits directly, the expert compares images based on concept similarity, *i.e.*, primes are more similar to primes than to other images. By running t-SNE on flattened pixel intensities, Fig. 3.3 (A) illustrates that the embedding does a reasonable job of clustering numbers by their label but clearly cannot understand primality because this concept is not apparent from visual appearance. To compensate, we sample triplet constraints of the form (i, j, k) where i and j share the same concept and k does not. However, we only sample 1,000 constraints for these 2,000 images. t-STE (B) attempts to discover the differences between the numbers in a "blind" fashion, but since it cannot take advantage of any visual cues, the underconstrained points are effectively random. If given many more constraints, eventually t-STE can only collapse everything into three points for each of the three abstract concepts. Our SNaCK embedding (C) displays the desired high-level concept grouping into primes/non-primes/others, and it can capture the structure of each class. Points with too few constraints are corrected by the t-SNE loss and the t-STE loss captures the appropriate structure.

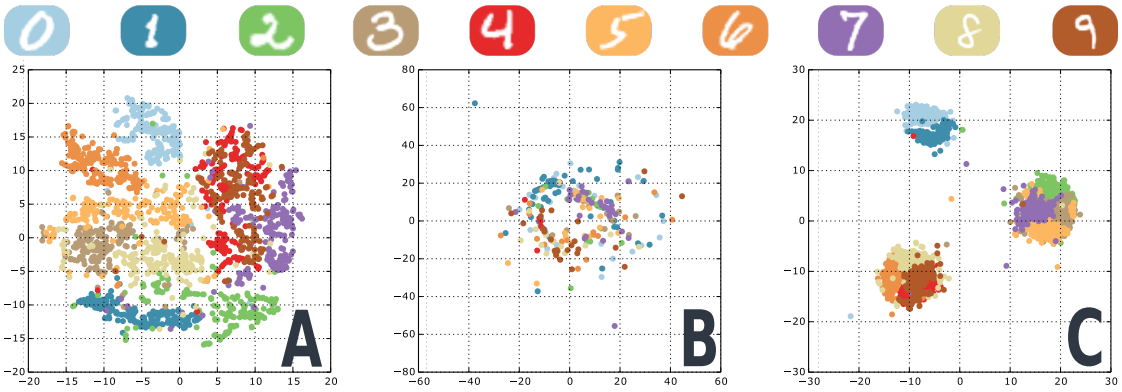

Figure 3.3: A simple MNIST example to illustrate the advantages of SNaCK's formulation. Suppose an expert wishes to group MNIST by some property that is not visually apparent, in this case: prime, composite or $\{0, 1\}$. (A) shows t-SNE on 2,000 MNIST digits using flattened pixel intensities. (B) shows t-STE on 1,000 triplets of the form (i, j, k), where i and j share the same concept but k does not. (C) shows a SNaCK embedding using the same flattened pixel intensities and the same triplet constraints. The SNaCK embedding is the only one that captures the intra-class structure from (A) and the desired abstract grouping of (B). See 3.3.2 for details.

3.4 Experiments

Our MNIST example demonstrates SNaCK's utility in a domain where concepts can be derived from category labels and everything is known *a priori*. How does SNaCK perform on domains where a fixed taxonomy or fixed category labels are not necessarily known up front? To explore this question, we perform a series of experiments: first, we showcase SNaCK's ability to help label a subset of CUB-200 in a semi-supervised fashion. In this setting, SNaCK learns concepts that are equivalent to category labels and outperforms other semi-supervised learning algorithms. Second, our experiments on a dataset of 10,000 unlabeled food images demonstrate SNaCK's ability to capture the concept of food taste using crowdsourcing. We evaluate the embedding's generalization error on a held-out set of crowdsourced triplet constraints. Finally, we showcase SNaCK's ability to embed a set of pictographic characters, showing how an expert can

interactively explore and refine the structure of an embedding where no prior knowledge is available.

3.4.1 Incrementally labeling CUB-200-2011

In this scenario, we show how SNaCK embeddings can help experts label a new dataset. Suppose an expert has a large dataset with category annotations and an unlabeled smaller set containing new classes similar to those they already know. The expert wishes to use their extensive preexisting knowledge to quickly label the new set with a minimum amount of human effort. Our goal is to show that SNaCK allows the expert to collect high-quality labels more quickly than other methods. Here, the "concepts" we learn are equivalent to category labels. These experiments are inspired by [41]. See Fig. 3.4.

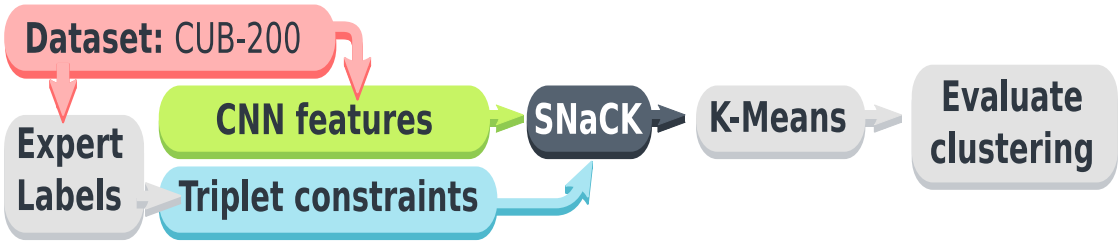

Figure 3.4: Experiment overview on CUB-200. See text for details.

Dataset. For this task, we use the "Caltech-UCSD Birds 200-2011" (CUB-200) dataset [67]. We assume the expert has access to all images and labels of 186 classes in the dataset (to train a machine kernel) and wishes to quickly label a testing set of 14 classes of woodpeckers and vireos. This subset contains 776 images and was defined in [19]. We only use profile-view bird images where a single eye and the beak is visible. Images are rotated, scaled, and possibly flipped so the eye is on the left side of the image and the beak is on the right side;

part locations are collected using crowdsourcing. The image is then cropped to the head. This is the same normalization strategy as [10].

Automatic similarity kernel. To generate K, we fine-tune a CNN to a classification task on all images in the 186 known classes. This allows the expert to leverage their extensive pre-existing dataset to speed up label collection for the novel classes. Our network is a variation of the "Network-in-Network" model [42], which takes cropped normalized bird heads as input and outputs a 186-dimensional classification result. We started from the pre-trained ImageNet model in the Caffe model zoo [32] and fine-tuned the network for 20,000 iterations on an Amazon EC2 GPU instance. To do this, we replaced the last layer with a 186-class output and reduced the learning rate for the other layers to a tenth of the previous value.[2] Finally, $K^{CNN}_{i,j}$ is the Euclidean distance between features in the final layer before softmax. To evaluate the importance of specialized kernels, we also compare this K^{CNN} kernel to Euclidean distances between pre-trained GoogLeNet [62] features, and Euclidean distance between HOG features.

Expert constraints. To generate triplet constraints in a semi-supervised fashion, we reveal the labels for n images of the dataset and sample all triplets between these images that satisfy same/different label constraints to generate $T_n = \{(i,j,k) \mid \ell_i = \ell_j \neq \ell_k, \max(i,j,k) \leq n\}$. This allows us to vary the amount of expert effort required to label the novel images. Note that in this test, our concepts to learn are equivalent to class labels, so all of our sampled constraints are derived from ground truth. Our food experiments, described in the next section, will demonstrate SNaCK's ability to learn more abstract concepts captured

[2] When trained using the standard training/testing protocol on all of CUB-200, this kind of model achieves 74.91% classification accuracy, which is comparable to the state-of-the-art [10].

from subjective human judgments.

Comparisons and metrics. To perform labeling with SNaCK, we generate an embedding of all 776 images and use KMeans to find clusters. To evaluate, we assign all points within each discovered cluster to their most common ground truth label and calculate the accuracy of this assignment. See Fig. 3.6 for example embeddings varying the number of expert label annotations. We compare against other semi-supervised learning and constrained clustering systems: *Label Propagation* [5], the multiclass version of the *Constrained Spectral Clustering KMeans* (CSPKmeans) method described in [70], and *Metric Pairwise-Constrained KMeans* (MPCKmeans) [6]. Label propagation uses K^{CNN} and the n revealed labels. The constrained clustering systems use K^{CNN} and pairwise "Must-Link" and "Cannot-Link" constraints as input, so we reveal n image labels and sample all possible pairwise constraints between them. As baselines, we calculate CNN features and try to cluster them with KMeans and spectral clustering, which do not benefit from extra human effort. Finally, we also compare against the cluster results of using K-Means on a t-STE embedding from the same triplet constraints used by SNaCK.

Results are shown in Fig. 3.5. SNaCK outperforms all other algorithms, but label propagation and MPCKMeans also perform well. CSPKmeans is eventually outpaced by naively asking the expert for image labels, perhaps because it was designed for the two-class setting rather than our 14-class case. These experiments show that t-STE benefits from an automatic machine kernel (compare SNaCK to t-STE), but we can improve the machine kernel with a small number of expert annotations (compare KMeans or Spectral Clustering to SNaCK).

Using a kernel that captures bird similarity well is particularly important for

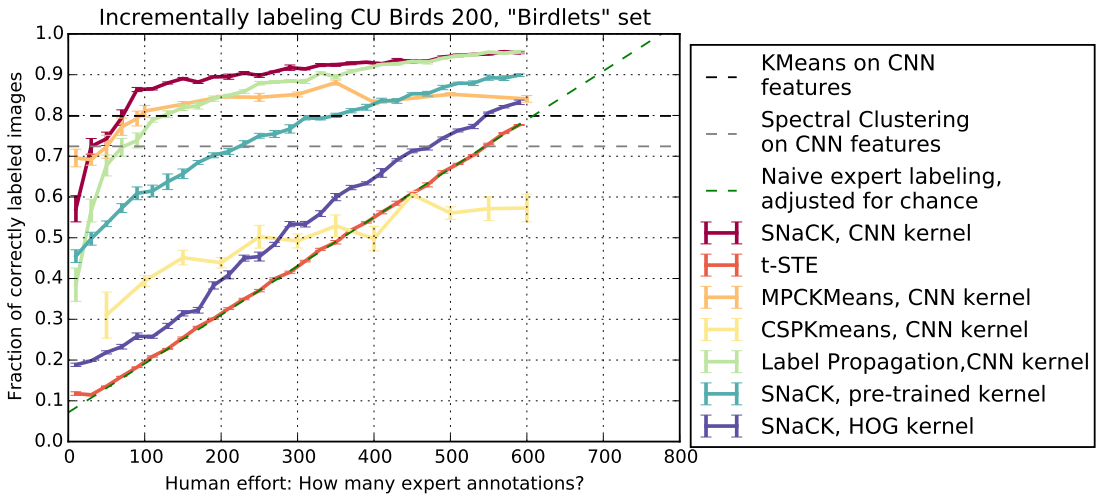

Figure 3.5: Incremental labeling accuracy of several semi-supervised methods. X axis: how many labels are revealed to each algorithm. Y axis: Dataset labeling accuracy. Error bars show standard error of the mean ($\sigma/\sqrt{n}$) across five runs. With 14 clusters, chance is ≈ 0.071. See Sec. 3.4.1 for details.

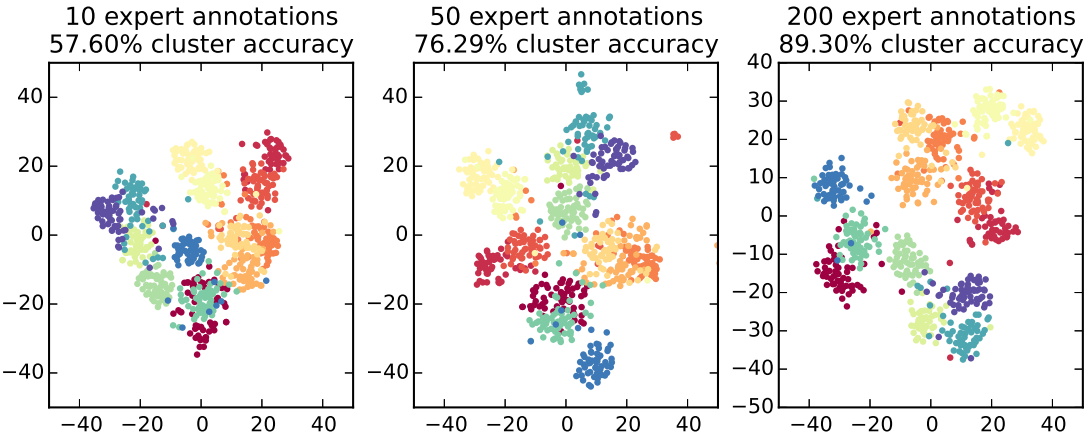

Figure 3.6: Embedding examples on CUB-200 Woodpeckers and Vireos, showing the "SNaCK" method with (left-to-right) 10, 50, and 200 expert label annotations. Colors indicate ground truth labels. As the number of expert annotations increases, clusters within the SNaCK embedding become more consistent.

this task. All of the algorithms which use K^{CNN} generally outperform SNaCK when using a pre-trained GoogLeNet kernel. HOG features, which use no learning, are only slightly better than naive labeling. Finally, t-STE cannot use any visual kernel, so it can only consider the images the expert already revealed.

Sometimes the machine kernel disagrees with the expert hints. This may happen for interesting reasons, such as mistakes in the training data. For example, Figure 3.1 shows an instance of a Red-headed Woodpecker that was moved into a cluster containing many Pileated Woodpeckers. Even though the human constraints encourage this sample to lie near similarly labeled examples, this individual looks overwhelmingly similar to a Pileated Woodpecker, so the t-SNE loss overpowered the t-STE constraints. If the embedding is colored with ground truth labels, this mistake shows up as a single differently-colored point in the expected cluster, which is immediately apparent to an expert.

Discovering labels for semi-supervised classifiers

Does better incremental labeling translate into increased classification performance? In this scenario, we extend our previous experiment: we use SNaCK to discover labels for a training set and measure the accuracy of a simple SVM classifier on a testing set. Our goal is to decide whether just letting an expert reveal n labels and training on this smaller set is better than revealing n labels and using SNaCK to discover the rest. Will a classifier trained on many noisy, discovered labels perform differently than a classifier trained on a smaller, perfect training set?

Dataset. This task uses the same set of 14 woodpeckers and vireos from

CUB-200 as before, but the procedure is different. We split our set into 396 training and 380 testing images using the same train/test split as CUB-200. We then discover labels on the training images using varying numbers of expert annotations and train a linear SVM classifier on all CNN features using the discovered labels. Finally, we report accuracy on the 380 testing images. The idea is that the quality of the discovered labels influences the accuracy of the classifier: a poor labeling method will cause the classifier to be trained on incorrect labels. Because all methods use the same type of classifier, we are evaluating the quality of our *discovered labels*, not the classifier itself.

Comparisons. As a baseline, we compare SVM classifiers trained on SNaCK-discovered labels to an SVM classifier trained on a smaller, better set of n correct labels provided by expert ground truth. This corresponds to the "Naive Human Sampling" method in Fig. 3.5. We also compare baselines where the SVM training set labels are discovered using KMeans, spectral clustering, and label propagation.

Results are shown in Fig. 3.7. Classifiers trained on noisy labels discovered from SNaCK embeddings significantly outperform classifiers that are trained on smaller training sets, even though many of SNaCK's labels are incorrect. This is particularly true for fewer than 50 annotations. Accuracy of SNaCK, Label Propagation, and naive label sampling saturates at about 85%, which is likely due to the linear SVM's limited generalization ability.

Interestingly, classification accuracy of labels discovered with MPCKMeans does not monotonically improve with more expert annotations. This surprises us, but Fig. 3.5 does show that MPCKmeans saturates to a smaller value in our semi-supervised labeling experiments, indicating that it cannot perfectly satisfy

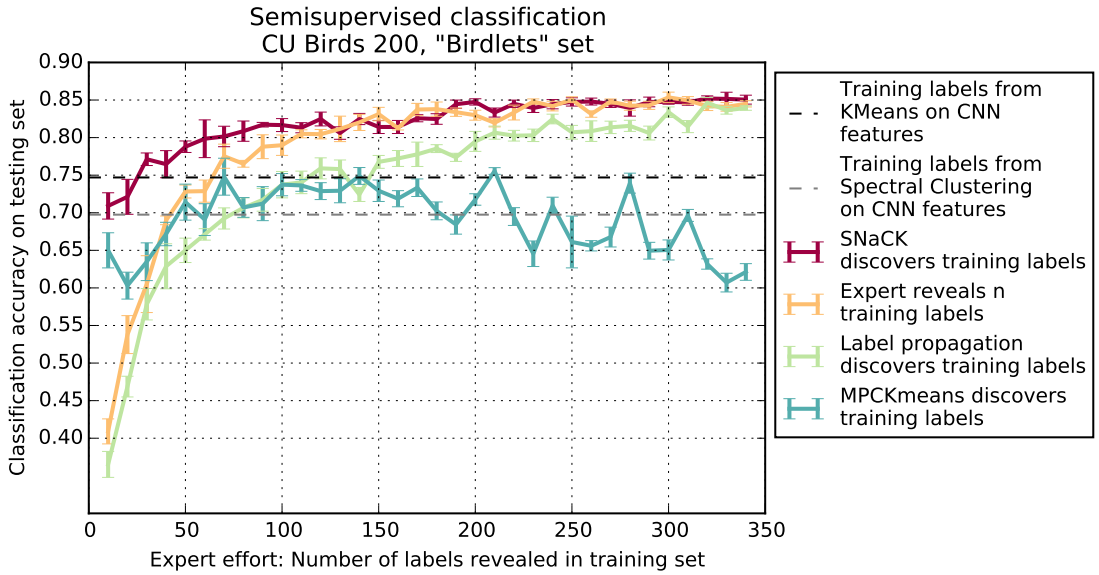

Figure 3.7: Classification accuracy of a linear SVM classifier trained on labels discovered by different methods. X axis: how many training labels are revealed to each algorithm. Y axis: Accuracy of classifier trained with these labels on the test set. Error bars show standard error of the mean ($\sigma/\sqrt{n}$) across five runs. See Sec. 3.4.1 for details.

(and thus does not benefit from) additional constraints.

Using SNaCK, an expert can build a classifier that achieves 78.8% classification accuracy by labeling 50 images (12.6% of the dataset). A standard SVM that achieves this level of accuracy requires a training set of 95 perfectly labeled images, showing that SNaCK can cut down the expert's work load to build training sets for classifiers.

3.4.2 Experiments on Yummly-10k

In this scenario, we use SNaCK to generate embeddings of food dishes. The goal is to create a concept embedding that captures the concept of taste. Two foods should be close in this embedding if they *taste similar*, according to sub-

Figure 3.8: Left: Example SNaCK embedding on Yummly-10k, combining expertise from Kernel 2 (CNN features) and 950,000 crowdsourced triplet constraints. Middle/right: Close-ups of the embedding. On a large scale, SNaCK groups major food kinds together, such as desserts, salad, and main courses. On a small scale, each food closely resembles the taste of its neighbors. See the supplementary material version for larger versions of this figure.

jective human judgments. This is different from the earlier bird experiments because we can no longer rely on labels or taxonomies to help refine the embedding; all expert hints must come directly from unquantified human perception annotations. See Fig. 3.9.

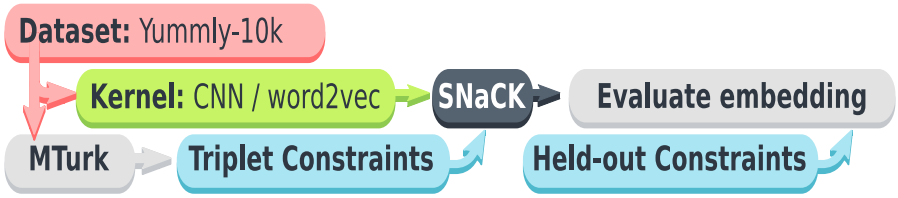

Figure 3.9: Experiment Overview for Yummly-10k. See text for details.

Dataset. For this experiment, we used 10,000 food images from the *Yummly* recipe web site, dubbed *Yummly-10k*. This data contains a variety of meals, appetizers, and snacks from different cultures and styles. We filtered the images by removing all images shorter or thinner than 300 pixels and removed all drinks and non-edibles. As metadata, Yummly includes weak ingredients lists and the title of the dish, but it does not include food labels.

Automatic similarity kernels. SNaCK is not specific to any specific kernel

representation, so we compare two kinds of similarity measures. *Food Kernel 1* is a semantic similarity measure of the best matching between two foods' ingredient lists, and *Food Kernel 2* is a visual similarity measure based on a convolutional neural network. To create $K_{i,j}^{\text{word2vec}}$ *(Food Kernel 1)*, let I_i and I_j be food i and j's ingredients lists from Yummly. Let $w(\cdot)$ be an ingredient's *word2vec*[48] representation, scaled to unit norm, and let cost matrix $C(a, b) = w(a) \cdot w(b)$ for $a \in I_i, b \in I_j$. Finally, let $f : I_i \to I_j$ be the maximum-weight assignment between the two ingredient lists. Then, $K_{i,j}^{\text{word2vec}} = -\sum_{a \in I_i} C(a, f(a))$. This way, *Food Kernel 1* determines foods that share many common ingredients are more similar than foods that have many dissimilar ingredients.

To build *Food Kernel 2*, we fine-tuned a CNN to predict a food label. Because *Yummly-10k* does not have any labels, we train on the *Food-101* dataset from [9]. Similarly to our earlier bird experiments, our network is a variation of the "Network-in-Network" model trained to classify 101 different foods. It was trained for 20,000 iterations on an Amazon EC2 GPU instance by replacing the last layer and reducing the learning rate. The final kernel is defined as the Euclidean distance between these CNN features. Our CNN model provides an excellent kernel to start from: when trained via the standard Food-101 protocol, this model achieves rank 1 classification accuracy of 73.5%. The previous best accuracy on this dataset is 56.40% from [9]; the best non-CNN is 50.76%. Of course, building a good classification model is not our focus, but we report this accuracy to show that the automatic kernel we use is effective at distinguishing different foods.

Expert annotation. Because we want our embedding to properly capture the concept of food taste, we collect our expert annotations directly from humans on

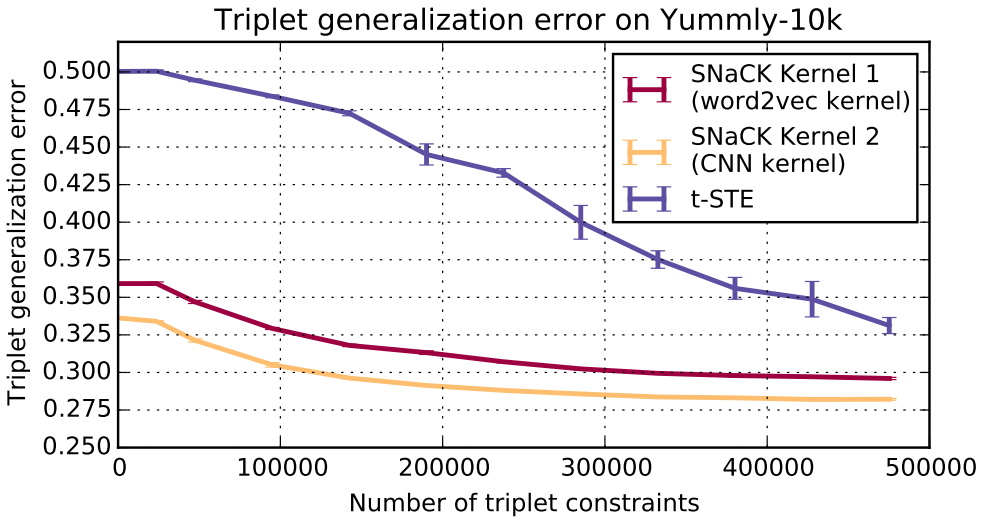

Figure 3.10: Increasing the number of crowdsourced triplet constraints allows all methods to improve the embedding quality, measured as the fraction of unsatisfied held-out triplet constraints ("triplet generalization error"). However, SNaCK-based methods converge much more quickly than t-STE and require less expert annotation to get a better result.

Amazon Mechanical Turk using the crowdsourcing interface of [71]. For each screen, we show a reference food image i and a grid of 12 food images. The human is asked to "Please select 4 food images that *taste similar* to the reference food i." We then generate all possible triplet constraints $\{(i, j, k), j \in S, k \notin S\}$, where S is the user's selection. Each HIT has 10 screens and yields 320 triplet constraints. In total, we collected 958,400 triplet constraints.[3]

Experiment design. There are no labels associated with taste in our Yummly data, so we must use other metrics to evaluate the quality of our perceptual embeddings. To do this, we adopt the "Triplet Generalization Error" metric common to previous work [28, 79, 66, 71]. We split all triplet constraints into training and testing sets and generate embeddings with varying numbers of training triplet constraints. Triplet generalization error is defined as the fraction

[3]Triplets are available from the companion website, http://vision.cornell.edu/se3/projects/concept-embeddings

of violated testing triplet constraints, which measures the embedding's ability to generalize to constraints the expert did not specify. We compare our two SNaCK kernels to t-STE.

Results are shown in Fig. 3.10 and an example embedding is shown in Fig. 3.8. As more triplet constraint annotations become available, all methods produce embeddings of higher quality. SNaCK with Kernel 2 eventually converges to 28% while t-STE reaches 33% error. Note that t-STE starts from random chance (50%) because it starts with no information, while SNaCK-based methods initially start with lower error because the Stochastic Neighbor loss on the automatic kernel encourages an initial embedding that contains some fine-grained information. Kernel 2 consistently outperforms Kernel 1, indicating that in this experiment, deep-learned visual features may be a better indication of food taste than the similarity of food ingredient lists. However, even the "weaker" semantic ingredient information provides a much better initial kernel than nothing at all.

3.4.3 Interactively discovering the structure of pictographic character symbols

In this section we describe possible tools for exploring unlabeled data. We chose to analyze a set of 887 pictographic characters, colloquially known as Emoji. Using CNN features pre-trained on ImageNet, we can create an embedding that does a good job of grouping visually similar Emoji together. However, if the goal is to capture the concept of emotion within the set of Emoji, then similarity of visual features alone may be inadequate. For example, in Fig. 3.11.A, a group

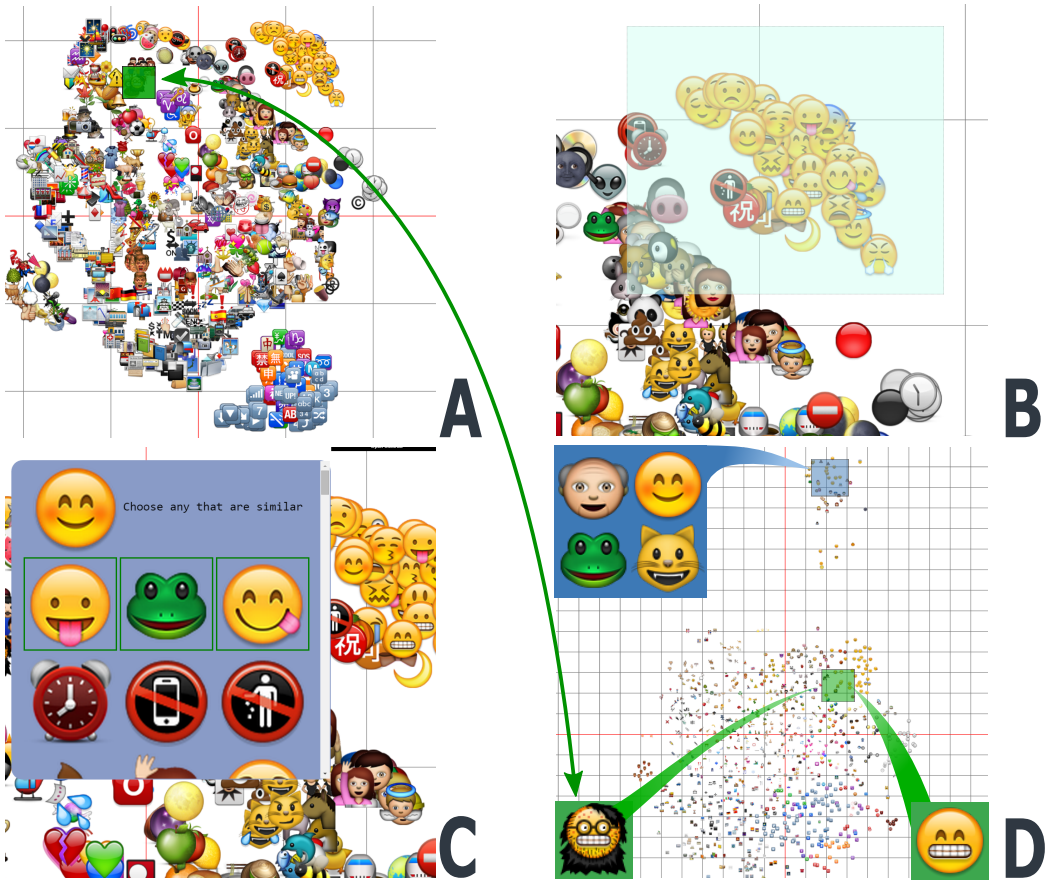

Figure 3.11: An example GUI used to interactively explore and refine concept embeddings. (A) shows a t-SNE embedding of Emoji using pre-trained ImageNet features. The user selects a set of images (B) and indicates which ones share the same emotion (C). For example, the user selected the smiling frog because it has a similar emotion to the top left image. The updated SNaCK embedding (D) moves smiling emoji away from unrelated images, regardless of the artistic style of the faces. Additionally one of the highlighted fearful faces, separate from the main cluster of faces in (A) has moved to be near faces with a similar expression without collecting triplets between them.

of yellow faces are clustered at the upper right, but this group contains different emotions and does not contain similar images in other artistic styles.

To interactively refine the embedding, the expert selects a reference Emoji and drags a box around several images. The expert then indicates which of these images share the same emotion as the reference. In the example in Fig. 3.11,

a smiling Emoji was selected and compared to all the Emoji in the green box (Fig. 3.11.B). After two bounding box selections and a few minutes of work, we are able to collect 20,000 triplets and separate many of the smiling Emoji from the rest of the embedding. From here, we could further inspect these Emoji and separate the emotion of laughing from smiling.

As mentioned in the MNIST experiments, the SNaCK embeddings are capable of taking advantage of visual cues when triplet information is not available. An example of this can be seen in Fig. 3.11.D. A fearful face with glasses is moved from the left side of embedding to be near other faces with similar expressions. SNaCK was able to do this without requiring triplets to be collected between these faces. These examples give a brief illustration of how SNaCK can be useful for examining unlabeled data.

3.5 Conclusion

Our SNaCK algorithm can learn concept embeddings by combining human expertise with machine similarity. We showed that SNaCK can help experts quickly label new sets of woodpeckers and vireos, build training sets for classifiers in a semi-supervised fashion, and capture the perceptual structure of food taste. We also presented a snapshot of a tool that can help experts interactively explore and refine a set of pictographic characters. In the future, we will pursue intelligent sampling for active learning of embeddings, and will extend our system to explore large video datasets.

3.6 Acknowledgments

This work was partially supported by an NSF Graduate Research Fellowship award (NSF DGE-1144153, Author 1), a Google Focused Research award (Author 4), and AOL-Program for Connected Experiences (Authors 1 and 4). We also wish to thank Laurens van der Maaten and Andreas Veit for insightful discussions.

CHAPTER 4

BAM! THE BEHANCE ARTISTIC MEDIA DATASET FOR RECOGNITION BEYOND PHOTOGRAPHY

"Art is an effort to create, beside the real world, a more humane world." – André Maurois

Recent advances in Computer Vision have yielded accuracy rivaling that of humans on a variety of object recognition tasks. Most work in this space is focused on understanding *photographic imagery* of everyday scenes. For example, the widely-used COCO dataset [43] was created by "gathering images of complex everyday scenes containing common objects in their natural context." Outside of everyday photography, there exists a diverse, relatively unexplored space of *artistic imagery*, offering depictions of the world *as reinterpreted through human artwork*. Besides being culturally valuable, artwork spans broad styles that are not found in everyday photography and thus are not available to current machine vision systems. For example, current object classifiers trained on ImageNet and Pascal VOC are frequently unable to recognize objects when they are depicted in artistic media (Fig. 4.1). Modeling artistic imagery can increase the generality of computer vision models by pushing beyond the limitations of photographic datasets.

In this work, we create a large-scale artistic style dataset from Behance, a website containing millions of portfolios from professional and commercial artists. Content on Behance spans several industries and fields, ranging from creative direction to fine art to technical diagrams to graffiti to concept design. Behance does not aim to be a historical archive of classic art; rather, we start

Figure 4.1: State of the art object detectors such as SSD trained on Pascal VOC can reliably detect objects in everyday photographs (top row), but do not generalize to other kinds of artistic media (see scores under each image). In this work, **we create a large-scale artistic dataset** spanning a breadth of styles, media, and emotions. We can use this dataset to improve the generality of object classifiers—our object classifier's scores are above 0.95 for all these images.

from Behance because it represents a broad cross-section of contemporary art and design.

Our overall goal is to create a dataset that can teach machines to understand

and categorize artistic images in ways that are valuable to humans. This is important because existing artistic datasets are too small or are focused on classical artistic styles, ignoring the breadth of contemporary digital artwork. To solidify the scope of the problem, we choose to explore three different facets of high-level image categorization: object categories, artistic media, and emotions. These artistic facets are attractive for several reasons: they are readily understood by non-experts, they can describe a broad range of contemporary artwork, and they are not apparent from current photographic datasets.

We keep the following goals in mind when deciding which attributes to annotate. For object categories, we wish to annotate objects that may be drawn in many different visual styles, collecting fewer visually distinct categories but increasing the density (instances per category) and breadth of representation. ImageNet and COCO, for example, contain rich fine-grained object annotations, but these objects only appear in everyday photos and thus only cover a narrow range of artistic representation. For media attributes, we wish to annotate pictures rendered with all kinds of professional media: pencil sketches, computer-aided vector illustration, watercolor, and so on. Finally, emotion is an important categorization facet that is relatively unexplored by current approaches.

There are several challenges, including annotating millions of images in a scalable way, defining a categorization vocabulary that represents the style and content of Behance, and integrating this resource into existing computer vision systems.

Our contributions are threefold:

- **A large-scale dataset**, the Behance-Media Dataset, containing almost 60

	Size	Scope	Annotations
A-SUN [53]	0.014m	Photos of scenes	Objects, context
Behance-2M (Private) [17]	1.9m	Contemporary artwork	User/View behavior
Recognizing Image Style [34]	0.16m	Photos, paintings	Art genre, photo techniques
AVA [51]	0.25m	Photos	Aesthetics, content, style
Visual sentiment ontology [8]	0.31m	Photos, videos	Adj/Noun pairs
OpenImages [37]	9.2m	Photos	Content labels
Behance-Media	**60m**	**Contemporary artwork**	**Emotion, Media, Objects**

Table 4.1: A comparison of several related datasets. Our Behance-Media dataset is much larger than the others and includes a broad range of contemporary artwork.

million images, a subset of which will be released upon publication. We also create an **expert-defined vocabulary** of binary artistic attributes that spans the broad spectrum of artistic styles and content represented in Behance.

- An **iterative label bootstrapping algorithm** that allows us to annotate this dataset at low cost while satisfying quality guarantees by focusing the crowd's attention on the most worthwhile images to label.
- An **analysis** of the dataset, showing how it may be used to improve generality of existing computer vision systems. We also use our dataset to teach machines to recognize images with different styles and emotions.

We believe this dataset will provide a starting foundation for researchers who wish to build systems that better understand artwork.

4.1 Related Work

Attributes and other mid-level representations have a long and rich history in vision. Two seminal works that introduce attributes are the "semantic multinomials" of Rasiwasia and Vasconcelos [56], which lift images into a semantic space useful for performing visual searches, and the work by Farhadi *et al.* [18],

which use human-describable attributes to perform zero-shot learning of new objects. Attributes have been applied to face recognition via the work of Kumar *et al.* [39] later extended by Scheirer *et al.* [60] and scene understanding by Patterson *et al.* [53], Redi *et al.* [57], and others.

Attributes have also been applied to aesthetics and other artistic qualities, usually with a focus on photography. For instance, Obrador *et al.* [52], Dhar *et al.* [16], and Murray *et al.* [51] collect descriptive attributes such as interestingness, symmetry, light exposure, and depth of field. Work by Peng *et al.* [54] attempts to study regions of photographs that induce a certain emotion in a viewer. Other work such as Jou *et al.* [33] and Borth *et al.* [8] use emotions to build ontologies of Adjective/Noun pairs to describe images.

Others describe image style in terms of low-level feature correlations as in work done by Gatys *et al.* [22], Lin *et al.* [44], and others. The application studied in Gatys' work is transferring texture from one image to another, but we argue there is more to artistic style than low-level texture transfer. We are more concerned about high-level image categorization.

Ours is not the only dataset focused on artwork. We compare related artistic datasets in Tab. 4.1. Most are focused exclusively on everyday photographs [51, 53, 8], but some [34] include classical paintings. For example, Crowley and Zisserman [12] studied how VOC categories appear in paintings. Likewise, Ginosar *et al.* [23] discuss person detection in cubist art. The work of Fang *et al.* [17] also studies Behance imagery, but does not collect descriptive attributes. Recently, Google released a large-scale dataset called "Open Images" [37]. As of this writing, there is no report explaining how this dataset was collected. Open Images contains some media-related labels including "comics",

"watercolor paint", "graffiti", etc. but it is unclear how the quality of the labeling was evaluated and each of these labels contain less than 400 human-verified images. Further, there are no labels relating to different emotions. Open Images contains some artistic imagery, but that is not its focus. To our knowledge, our work is the first work seeking to release a large-scale dataset of a broad range of contemporary artwork with emotion, media, and content annotations.

Our work is most similar in spirit to Karayev *et al.* [34], which studies photographic image style. They collect annotations for photographic techniques, composition, genre, and mood on Flickr images, as well as a set of classical painting genres on Wikipaintings. Our focus is on non-photorealistic contemporary art, which is also covered by Fang *et al.* [17]. Fang *et al.*'s work trains a style prediction network to predict image "pseudoclasses," which are clusters of images that encompass consistent styles according to user behavior. Our approach is explicit: we directly annotate semantically meaningful attributes from that feature space.

Finally, several works [78, 38, 13] show how to use deep learning to amplify human effort. The design of our crowdsourced dataset collection process is loosely based on the LSUN dataset annotation pipeline [78], which builds a very large-scale object detection dataset using a combination of deep learning and crowdsourcing.

4.2 The Behance Media Dataset

Our dataset is built from "Behance," a portfolio website for professional and commercial artists. Behance contains over ten million projects and sixty million

Figure 4.2: Top: Sampling of images within projects with the "Cat" tag. Projects with the "Cat" tag may contain other animals (1), title cards (3,5), or unrelated pictures (5,6). Bottom: Top classifications from a classifier trained to distinguish the "Cat" tag. Images are more related, but this tends to learn many small animals. The precision of cats in the top 100-scoring images is only 36%.

images. Images on Behance are grouped into Projects, the fundamental unit of categorization. Each Project is associated with metadata, including a title, optional description, several user-supplied tags, and up to three annotations specifying the field of the Project.

Artwork on Behance spans many fields, such as sculpture, painting, photography, graphic design, graffiti, illustration, and advertising. Graphic design and advertising make up roughly one third of Behance. Photography, drawings, and illustrations make up roughly another third. This artwork is posted by professional artists to show off samples of their best work. We encourage the reader to visit `http://behance.net` to get a sense of the diversity and quality of imagery on this site. Example images from Behance are shown in Fig. 4.4.

Selecting attribute categories. In this work, we choose to annotate our own artistic binary attributes. Attribute names are rendered in sans serif font. Our attributes capture three categorization facets:

- **Emotion attributes:** We label images that are likely to make the viewer

feel calm/peaceful, happy/cheerful, sad/gloomy, and scary/fearful.

- **Media attributes:** We label images created in 3D computer graphics, comics, oil painting, pen ink, pencil sketches, vector art, and watercolor.
- **Entry-level object category attributes:** We label images containing bicycles, birds, buildings, cars, cats, dogs, flowers, people, and trees.

We chose these attributes as follows: The four emotion attributes are seen on Plutchik's *Wheel of Emotions* [55], a well-accepted model for emotions from the psychological literature. This model was also used in [33]. From this model, we chose the emotions that are likely to be visually distinctive. The seven media attributes were chosen on the expert advice of a resident artist to roughly correspond with the genres of artwork available in Behance that are easy to visually distinguish. Our goal is to strike a balance between distinctive media while covering the broad range available in Behance. For instance, oil paint and acrylic are considered to be different media by the artistic community, but are very hard for the average crowd worker to distinguish visually. The content attributes represent entry-level object categories and were chosen to have some overlap with Pascal VOC while being representative of Behance content. We focus on entry-level categories because these categories are likely to be rendered in a broad range of styles throughout Behance.

Although this work is only concerned with a small set of labels (arguably a proof-of-concept), the dataset we release could itself be the basis for a real PASCAL/COCO-sized labeling effort which requires consortium-level funding.

Tags are noisy. Behance contains user-supplied tags, and one may wonder whether it is feasible to train attribute classifiers directly from these noisy tags alone. This idea was previously studied in Izadinia *et al.* [30] and Misra *et*

al. [50]. However, unlike that work, we cannot create our dataset from tags alone for two reasons. First, not all of our attributes have corresponding tags. Second, tags are applied to each *project*, not each image. For example, even though a project called "Animal sketches 2012" may have the "Dog" tag, we do not know which image that tag should apply to. Training on tags alone is too noisy and reduces the final classifier precision. As a toy experiment, Fig. 4.2 shows a sampling of images from projects with the "Cat" tag, but many of these images do not contain cats. A binary classifier trained on this tag only learns to distinguish different small animals and is not fine-grained enough to find cats. The precision of cats among the top 100 detections is only about 36%. To increase this accuracy, we must rely on human expertise to collect labels.

4.2.1 Crowdsourcing

Our dataset requires some level of human expertise to label, but it is too costly to collect labels for all images. To address this issue, we use a hybrid human-in-the-loop strategy to incrementally learn a binary classifier for each attribute. At each step, humans label the most informative samples in the dataset. The resulting labels are added to each classifier's training set to improve its discrimination. The classifier then ranks more images, and the most informative images are sent to the crowd for the next iteration. After four iterations, the final classifier re-scores the entire dataset and images that surpass a certain score threshold are assumed to be positive. This final threshold is chosen to meet certain precision and recall targets on a held-out validation set. This entire process is repeated for each attribute we wish to collect. Our hybrid annotation strategy is loosely based on the LSUN dataset annotation pipeline described in [78]. An

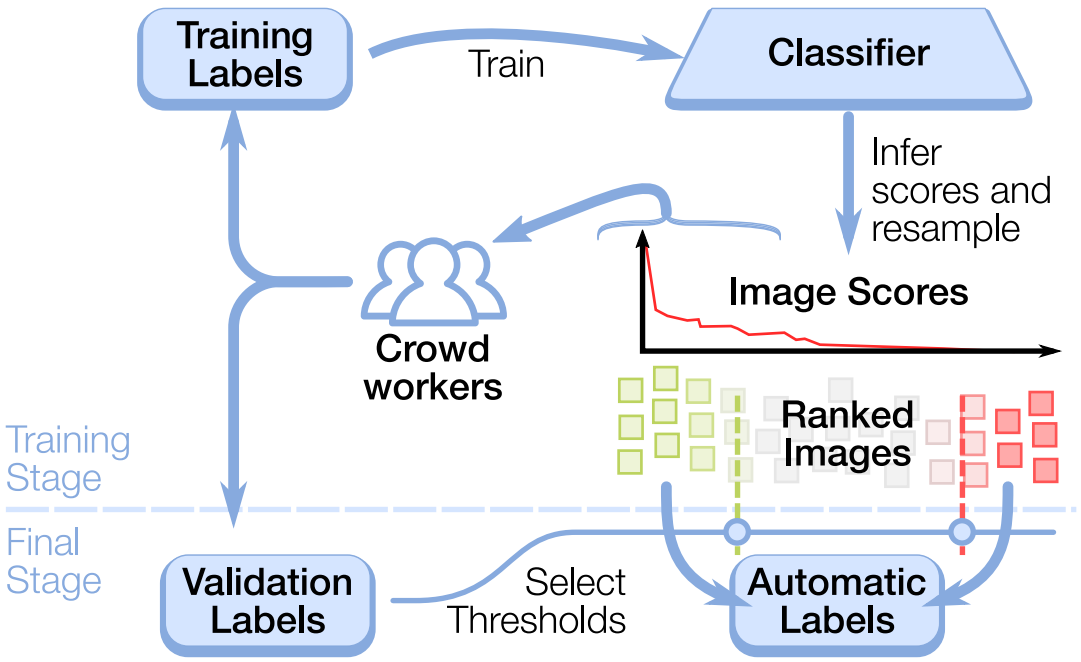

Figure 4.3: A diagram of our crowdsourcing pipeline. First, we train a set of classifiers on all labels collected so far. We then use this classifier to rank a random sample of images. High-scoring images are sent back to the crowd, and the resulting labels are added to the training and validation set. After four iterations, the validation set is used to select positive and negative thresholds with certain precision and recall targets. Images meeting these thresholds are added to the automatic label set.

overview of this process is shown in Fig. 4.3.

The human crowdsourcing task. The heart of our human-in-the-loop system is the actual human annotation task. We collect annotations for each attribute independently. To do this, we rely on Amazon Mechanical Turk, a crowdsourced marketplace. Crowd workers (*"Turkers"*) complete Human Intelligence Tasks for a small cash payment. In each HIT for a given attribute, we show the Turker 10 handpicked positive/negative example images and collect 50 binary image annotations. Turkers indicate whether each image has the attribute of interest. Each HIT only collects labels for a single attribute at a time to avoid confusion. For quality control, we show each image to two separate

Turkers and only use answers where both Turkers agree.

We also collect sparse text annotations for a subset of these images. Every 10 images, we present an annotation recently provided by the Turker and ask for a brief 3-word caption to justify their choice. For emotion attributes, we ask why the image might or might not make an average Turker feel that emotion; for media attributes, we ask how the Turker knows the image was or was not drawn with that medium; for content attributes, we merely ask what the object of interest looks like. The Turker must write at least three words before continuing, but captions are not checked for grammar or coherence. This has the effect of encouraging annotators to carefully consider and justify their choices. Feedback from the Turkers indicate that they found this extra captioning task to be annoying and to slow them down; however, it seemed to greatly increase the quality of the provided attribute labels based on manual inspection of results.

These annotations also provide useful clues about what qualities workers use to describe style. The words that maximize TF/IDF scores among positive annotations are informative: workers tend to use nouns when describing object categories (such as "bouquet", "rose", "petal", "vase" for Flower) and visual adjectives when describing media and emotion (such as "translucent", "frayed", "blotchy", "bleeding", "overlap" for Watercolor). The supplementary material contains more examples of informative words. These annotations will be released alongside the final dataset.

It is always important to balance the trade-off between squeezing high-quality work out of annotators while being respectful of their effort and abilities. The subjectivity of our task makes this trade-off harder to manage. Beyond occasionally asking Turkers for justification, we did not feel a need to force them

to surpass accuracy thresholds on "gold standard" tasks. We did not reject any HIT responses and annotators were always paid for their time, opting to instead ignore low-quality responses without consensus. From manual inspection, we found that with adequate examples, our annotators generally understood the task and answered to the best of their abilities.

Iterative learning. Starting from the initial label set, the dataset is enlarged by an iterative process that alternates between training a classifier on the current label set, applying it to unlabeled images, and sending unconfident images back to the crowd for more labeling. We start the process by constructing a *tentative training set*. For media and content attributes, we sample images with handpicked tags as positives and random images as negatives. For example, the Dog attribute is seeded by a classifier trained on positive images from Behance tagged with "Dog". The first classifier is trained on this tentative training set with the expectation that this classifier's guesses will be quickly refined by the crowd. For emotion images, there are no suitable tags, so we start by collecting a training set from the crowd, randomly sampled from photography and fine art fields.

On each iteration, we train a deep learning classifier using 90% of the total collected crowd labels. The last 9% are always held out for validation. To select the next round of images to show to the crowd, we define an "interestingness" score threshold on the validation set such that the precision of relevant validation images above this threshold is 50%. The crowd labels 5,000 "interesting" images above this threshold. This way, the crowd always sees an even split of likely-positive and likely-negative images. The resulting crowd labels are added to the training set for the next iteration.

After four iterations, we arrive at a final classifier that has good discrimination performance on this attribute. We score the entire dataset with this classifier and use thresholds to select the final set of positives and negatives. The positive score threshold is chosen such that the precision of higher-scoring validation images is 90%, and the negative threshold is chosen such that the recall of validation images above this threshold is 95%. In this way, we can ensure that our final labeling meets strict quality guarantees.

It is important to note that the resulting size of the dataset is determined solely by the number of relevant images in Behance, our desired quality guarantees, and the accuracy of the final classifier. A better attribute classifier can add more images to the positive set while maintaining the precision threshold. If we need more positive data for an attribute, we can sacrifice precision for a larger and noisier positive set.

Classifier. For content attributes, our classifier is a fine-tuned 50-layer ResNet [26] originally trained on ImageNet. For emotion and media attributes, we found it better to start from *StyleNet* [17]. This model is a GoogLeNet [62], fine-tuned on a style prediction task inferred from user behavior. Each network is modified to use binary class-entropy loss to output a single attribute score. To avoid overfitting, we only fine-tune for three epochs on each iteration. See Fig. 4.4 for examples of Behance images.

4.2.2 Resulting dataset statistics

Our final dataset includes positive and negative examples for 20 attributes. The median number of positive images across each attribute is 54,000, and the me-

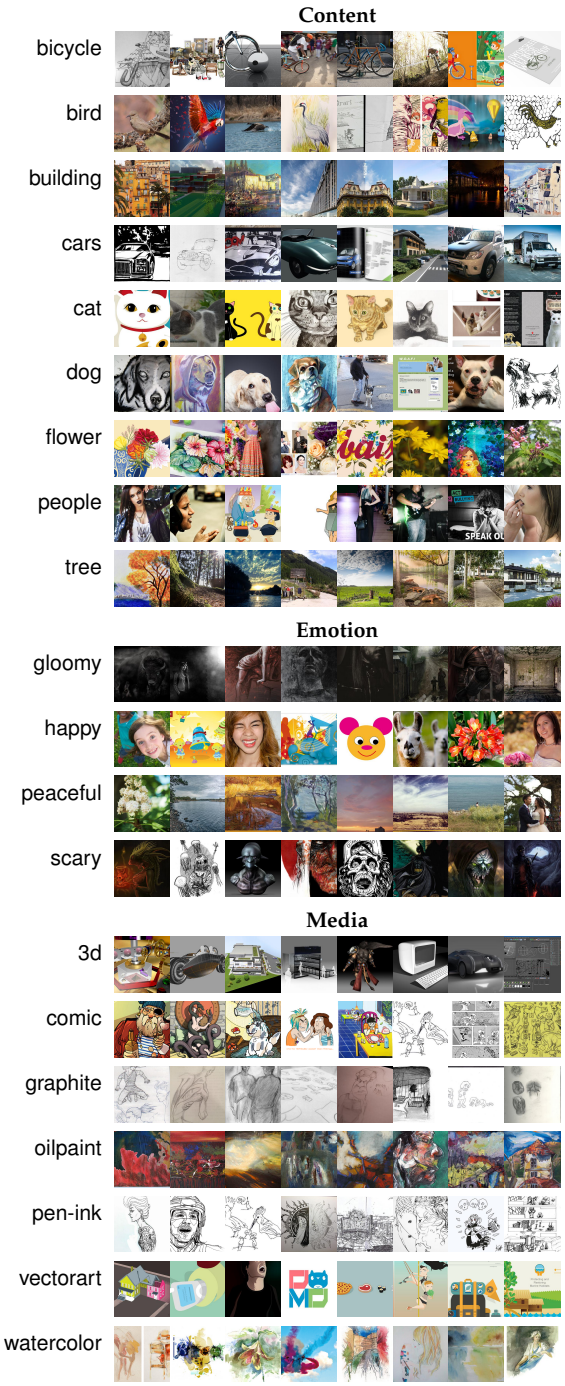

Figure 4.4: Example images from Behance Artistic Media. We encourage the reader to zoom in for more detail.

dian number of negative images is 8.7 million. The "People" attribute has the most positive images (1.74 million). Humans are commonly featured as art sub-

jects, so this is not surprising. The attribute with the least positives is "Cat" with 19,244 images. We suspect this is because our labeler cannot easily distinguish cats from other cat-like renditions. As shown in supplementary material, cats on Behance are commonly rendered in many different styles with very high intra-class variation. Statistics for all attributes are shown in Fig. 4.5.

Deep learning amplifies the effort of our human annotators by a factor of 505 averaged across all attributes. Here, "amplification" is defined as the number of automatically inferred labels divided by the number of images seen by the crowd. When labeling a dataset as large and diverse as Behance, automatic systems can quickly throw away easy negatives, focusing the crowd's attention on potentially relevant images. This means most of the amplification effect comes from negative images. If we alternatively define amplification as the number of automatically-labeled *positive* images divided by the number of crowd-labeled positive images, the average amplification factor is 17.4.

4.2.3 Final quality assurance

As a quality check, we tested whether the final labeling set meets our desired quality target of 90% precision at 95% recall. For each attribute, we show annotators 100 images from the final automatically-labeled positive set and 100 images from the final negative set. Images are presented in random order using the same interface used to collect the dataset. Fig. 4.6 shows worker agreement on the positive set as a proxy for precision. The mean precision across all attributes is 90.4%, where precision is the number of positive images where at least one annotator indicates the image should be positive.

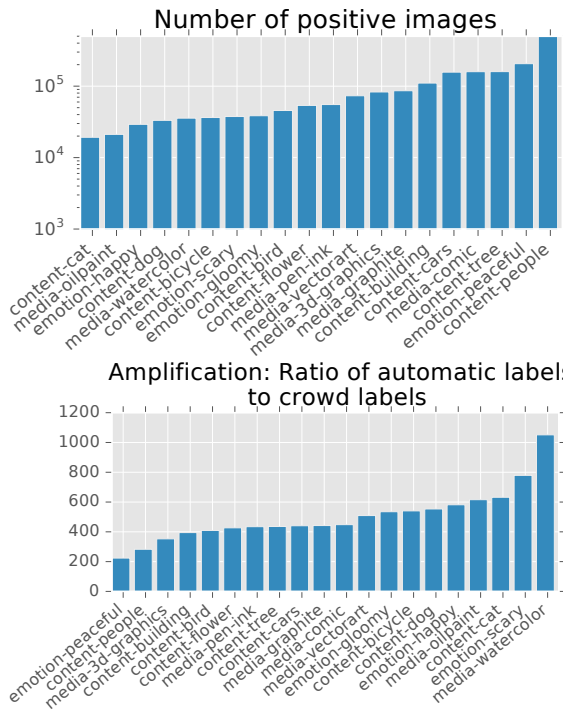

Figure 4.5: Top: Number of positive images in the final set. Bottom: Amount of amplification for each attribute (number of automatic labels divided by number of crowd labels)

To measure recall, we examine how many workers found positive images in the negative set. Across all attributes, at least one worker indicated the image was negative for 98.9% images in the negative set, surpassing our original recall target of 95%.

4.3 Experiments

We can use this dataset to teach machine vision systems about high-level image categorization. First, we explore the representation gap in pre-trained object detectors, showing that existing systems cannot detect objects rendered across many different art styles. Training on artistic imagery improves detection performance. Second, we compare different feature extraction strategies on emo-

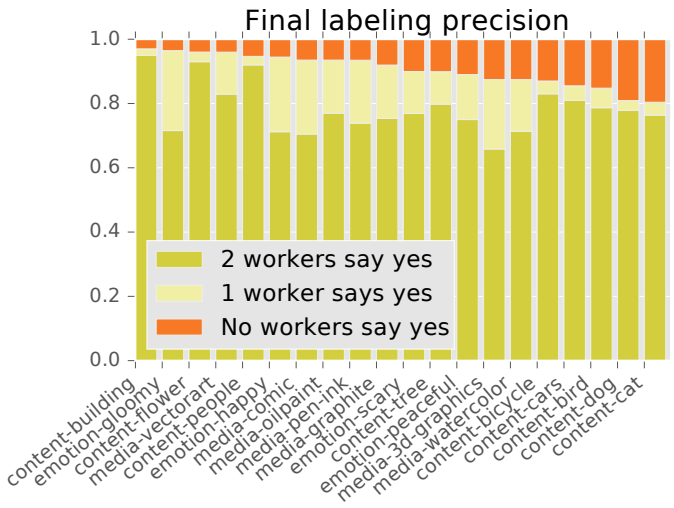

Figure 4.6: Final quality assurance: Showing worker agreement of automatically-labeled positive images in the final dataset.

tion and media attributes. Finally, we use Behance-Media to improve the performance of style classification tasks on other datasets.

4.3.1 Detecting objects in artwork

How well do existing object detectors generalize to artistically-rendered objects? We expect this task to be difficult because existing object detectors trained on ImageNet or VOC are only exposed to a very narrow breadth of object representations. Objects in photographs are constrained by their real-world appearance.

We consider 6 content attributes that correspond to Pascal VOC categories: Bicycle, Bird, Cars, Cat, Dog, People. We then extract scores for these attributes using two object detectors trained on VOC: YOLO [58] and SSD [45]. For the sake of comparison, we use these detectors as binary object classifiers by using the object of interest's highest-scoring region from the detector output. We also compare to ResNet-50 classifiers [26] trained on ImageNet, taking the maximum

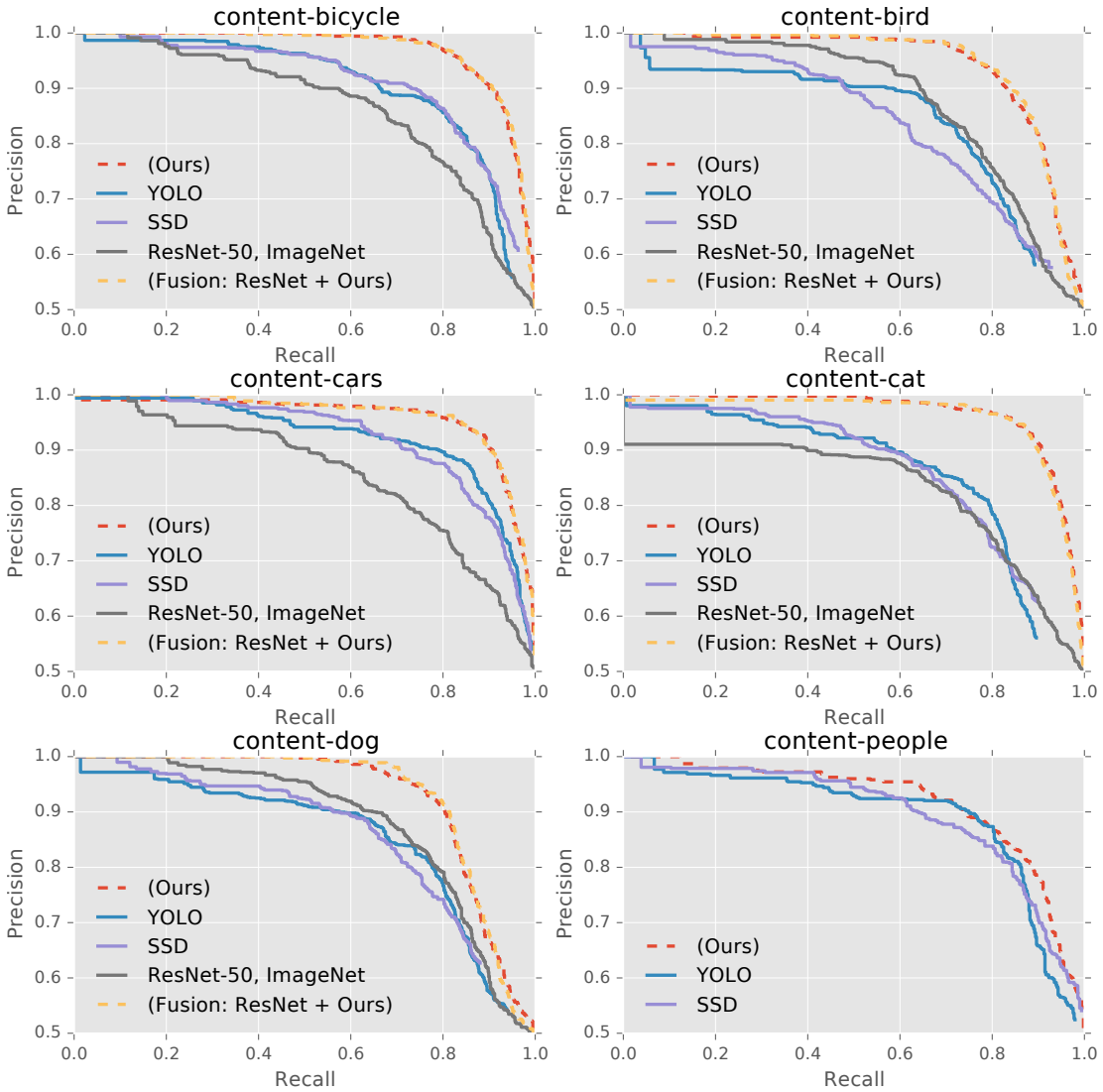

Figure 4.7: PR curves for different VOC object categories comparing our model, YOLO, SSD, ResNet-50, and fusion of ours and ResNet-50.

dimension of the ImageNet synsets that correspond with the VOC category of interest. In this way, we can measure how well existing object detectors and classifiers already find objects in art without extra training. We also compare to our final attribute classifier trained in Sec. 4.2.1.

We evaluate these methods on 5,000 positives and 5,000 negatives on each attribute's *human-labeled* validation set to avoid potential bias from the auto-

AP	Ours	Yolo	SSD	RN50	Fusion
Bicycle	**0.9703**	0.9008	0.9116	0.8702	0.9704
People	**0.9103**	0.8863	0.8952	—[1]	—
Bird	**0.9400**	0.8516	0.8387	0.8768	0.9453
Cat	**0.9660**	0.8583	0.8620	0.8026	0.9501
Cars	**0.9551**	0.9140	0.9194	0.8519	0.9628
Dog	**0.9272**	0.8510	0.8582	0.8818	0.9293

Table 4.2: Average precision across different VOC categories using our model, YOLO, SSD, ResNet-50, and fusion of ours and ResNet-50. [1]: We do not report people results because there are relatively few ImageNet people categories.

matic labeler. The results are shown as precision/recall curves in Fig. 4.7 and AP is shown in Tab. 4.2. Vision systems trained on photography datasets like VOC (YOLO, SSD) and ImageNet (RN50) perform worse than vision systems that have seen objects in artwork during training. Indeed, from manual inspection, most false negatives of these systems involve objects rendered with unique artistic styles. Specific failure cases are shown in Fig. 4.1. This shows that existing datasets have a *representational gap* that can be amended by seeing more training data.

We can improve performance slightly by fusing ImageNet and Behance scores together with a simple linear combination. The resulting "Fusion" model performs slightly better than our own model and ResNet-50 on all but two attributes.

4.3.2 Emotion and media: Which features are best?

Here, we compare the performance of different feature learning strategies on our dataset's emotion and style attributes, complementing the content attributes studied above. Our hypothesis is that features trained on object categories

should perform better on object attributes and features trained on style prediction tasks should perform better on style attributes. To test this hypothesis, we extract features from the final linear layer of a pre-trained ResNet-50 model and features from StyleNet [17]. We then measure the precision and recall of a linear SVM on held-out human labels for each attribute.

Performance for six attributes is given in Fig. 4.8. For all four emotion attributes, the AP of linear classifiers on StyleNet features is greater than the AP of linear classifiers on ImageNet-derived features. StyleNet features also have higher AP than ImageNet-derived features on four out of six media attributes. However, ImageNet-derived features have higher AP than StyleNet features on all nine content attributes.

This supports the view that vision systems trained to distinguish object categories can more easily transfer domain knowledge to distinguishing artistic objects as well. One might wonder whether information about artistic style would be more important for this task, but that is not the case; instead, fine-tuning a network for style prediction tasks [17] makes it more suitable to distinguish emotion and media attributes at the cost of reducing its object detection performance.

4.3.3 Using Behance-Media to improve the performance of existing models

In this section, we show that automatic labels from Behance-Media can improve style classification on existing datasets. We evaluate on the three datasets intro-

duced in [34]: 80,000 images in 20 photographic styles on Flickr, 85,000 images from the top 25 styles on Wikipaintings, and the 14,000 images with 14 photographic styles from the hand-labeled set of AVA [51]. For comparison to previous work [17], we report AVA classification accuracy calculated only on the 12,000 images that have a single style label.

Our joint attribute model (*JAM*) training works as follows. Each training sample (x, i, ℓ) is a tuple of image x, attribute index i, and label $\ell \in \{-1, 1\}$. It is not suitable to train this model using ordinary cross entropy because each attribute is not mutually exclusive. Thus, we must use a loss function with two properties: each attribute output should be independent of other attributes and unknown attribute values should not induce any gradient. We lift image x to a 20-dimensional partial attribute vector $\hat{y} \in \mathcal{R}^{20}$, where $\hat{y}_{j \neq i} = 0$ and $\hat{y}_{j=i} = \ell$. This allows us to train using a soft-margin criterion,

$$loss(x, y) = \frac{1}{20} \sum_i \log(1 + \exp(-\hat{y}_i y_i)). \tag{4.1}$$

Our JAM model is a fine-tuned ResNet-50 model with a linear projection from 1,000 to 20 dimensions. Except for the last layer, they share the same architecture and are merely trained using different loss functions. We trained our model for 100 epochs, starting with a learning rate of 0.1 and multiplying it by 0.93 every epoch. The training set includes roughly 2 million images evenly sampled between attributes and evenly distributed between positive and negative images drawn from the automatically-labeled images in Behance-Media.

Results are shown on Table 4.3. On all three challenges, our model shows improved results compared to both the original ResNet-50 and StyleNet. This shows that Behance imagery is rich and diverse enough to improve style recognition tasks on other datasets. This is particularly interesting because Flickr

	JAM	ResNet-50 (ImageNet)	StyleNet [17]
Flickr	**0.389**	0.376	0.372
Wikipaintings	**0.508**	0.505	0.414
AVA	**0.615**	0.603	0.560

Table 4.3: Performance of our joint model for style detection on other datasets

AVA are both focused on photographic style. Categories in AVA are chosen to be useful for aesthetic quality prediction tasks. In a sense, we have shown that a model's knowledge of emotions and media could potentially transfer to photographic style and aesthetic prediction.

4.4 Conclusion

Computer vision systems need not be constrained to the domain of photography. Here, we show how the rich field of artistic imagery can benefit machine vision systems. We propose a new dataset, "Behance-Media." This dataset is derived from Behance, a repository of millions of images posted by professional and commercial artists, representing a broad snapshot of contemporary artwork. We collected a rich vocabulary of emotion, media, and content attributes that are both visually distinctive and representative of the diversity found in Behance. However, though Behance does include tag metadata, we showed that these tags are too noisy to learn directly. Further, the scale of Behance makes brute-force crowdsourcing unattractive.

To surmount these issues, we collected labels via a hybrid human-in-the-loop system that uses deep learning to amplify human annotation effort. This allows existing machine vision systems to focus crowd attention on the images

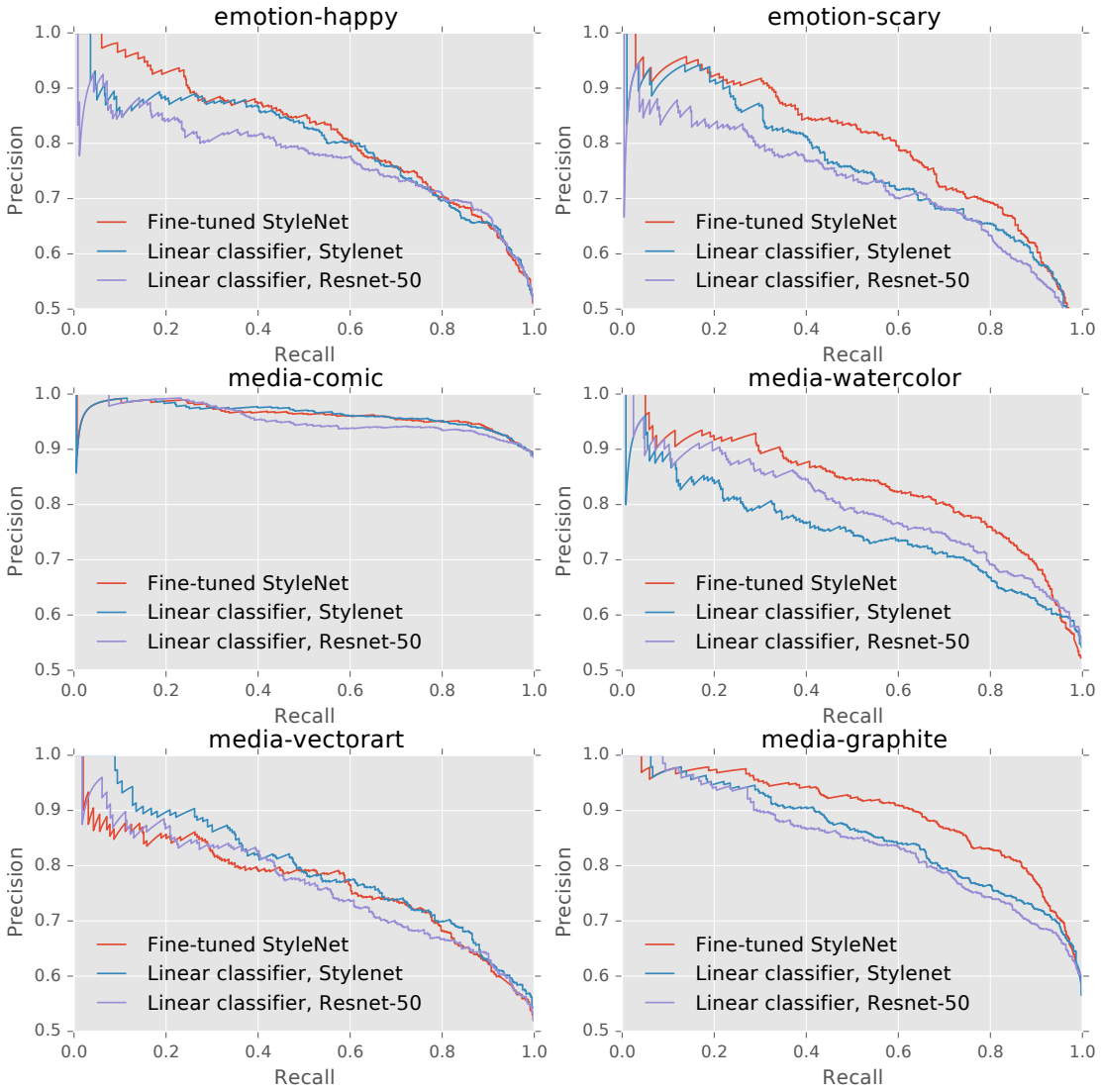

Figure 4.8: Performance of different features on six emotion and media attributes.

that need human expertise. Our annotation pipeline collects labels at a fraction of the cost of brute force labeling while meeting precision/recall guarantees of our choosing.

The resulting dataset is useful for several computer vision tasks. We use it to highlight the representation gap of current object detection systems trained on photography, showing that Behance captures a wider gamut of representation

styles than current sets such as VOC and ImageNet. We also use Behance to improve the performance of style classification tasks on other datasets, showing that researchers can train on our dataset for a marked improvement in performance.

We believe our dataset provides a good foundation for further research into the unexplored realm of large-scale artistic imagery.

CHAPTER 5

CONCLUSION

Overall, our focus was on creating models that have *strong intuition* without having to rely on densely labeled training data carefully curated by human experts.

We anticipate several real-world applications of the work presented here. The embedding methods used in Chapter 2 and Chapter 3 are useful when visualizing large-scale datasets using expert refinement. For example, an expert journalist could use SNaCK to embed a large corpus of related news articles using an automatic article similarity kernel. The expert would then be able to use hand-tuned constraints to refine the embedding to gain a better understanding of how news spreads through social media. Alternatively, one could build a "visual similarity field guide" using the data collected in previous work [68]. This field guide could be used by amateur birders to better understand families of similar-looking easily-confused birds, similar to Merlin Bird ID[1]. Finally, the large-scale crowdsourcing used in Chapter 4 is an effective way to collect large-scale image datasets to use worker time more efficiently.

First, Chapter 2 first explored the *triplet query* as the fundamental unit of perceptual similarity. Triplet queries capture a small piece of information about the relationships between similar objects. To turn this into useful inferences, we construct a *concept embedding* that satisfies as many triplet queries as possible. That way, distances within this embedding correspond to perceptual similarity. However, collecting raw triplet queries still requires a great deal of effort from our human annotators. To address this issue, we show how to use a novel anno-

[1]See http://merlin.allaboutbirds.org/

tation UI design to increase the number of constraints by an order of magnitude without a corresponding increase in annotation effort, leading to higher-quality concept embeddings at less cost.

From here, we extend this work in Chapter 3 for very large-scale settings. Unfortunately, with very large datasets, sparse human annotation is not enough, so we combine these expert labels with automatic similarity kernels. This way, deep-learned models can perform most of the "heavy lifting" by imputing the embedding with their *visual* knowledge while human experts fine-tune the result with their *intuitive* knowledge. The resulting embedding captures visual similarity and expert intuition.

Finally, we build a bridge back from abstract concept embeddings to explicit ontological semantics in Chapter 4. We study the task of assigning expert-assigned labels for objects, styles, and emotions featured in a large-scale collection of digital artwork. As before, the idea is to combine computer vision models with human raters within a "human-in-the-loop" setting. To do this, we incrementally train a deep-learned model to label the dataset while simultaneously guiding crowd annotator attention toward the difficult or unclear examples, similar to an active learning approach. Taking care of the "easy" examples automatically makes the best use of the crowd's effort.

All of the work in this thesis makes extensive use of the Mechanical Turk crowdsourcing platform. Many workers on Mechanical Turk typically perform this job professionally, many come from underprivileged backgrounds, and many rely on full-time Turk tasks as their only form of income. This means that the requester/worker power dynamic can have considerable impact on workers' quality of life. To respect this, we try to follow the "We Are Dynamo"

group's "Guidelines for Academic Requesters" as best we can.[2] In particular, we try to provide realistic time estimates and detailed descriptions, we pay the median worker a wage of at least $8 per hour (a number generally considered fair on this platform), and we never outright reject work from workers. Instead, we use quality control techniques to ignore problematic results, trusting workers to complete these subjective tasks to the best of their abilities. In rare cases where we need to block a worker, we do so nondestructively without harming their permanent record.

We leave several potential avenues for future work. For example, it is still not widely known which attributes could characterize artistic expression. In Chapter 4, we relied on an expert taxonomy, but it is conceivable that a model with strong intuition may understand which related concepts are meaningful to humans and which are not. Additionally, it could be worthwhile to study how to transfer strong intuition from a pre-trained machine back to a human learner in a machine-teaching setting.

Finally, it is worthwhile to investigate the many forms of bias in our results, both in terms of demographic worker selection and domain familiarity. For example, workers from different cultures and backgrounds may be familiar with different kinds of food, so they may be able to offer specialized expertise. A better system could potentially model and take advantage of workers' familiarity within the domain of interest for fine-grained refinement of the embedding results.

Bias creeps into our system in other ways. For example, our work only uses Mechanical Turk workers from the United States. The Food-100 and

[2]See the open letter to researchers here: http://wiki.wearedynamo.org/index.php/Guidelines_for_Academic_Requesters

Yummly10k dataset used in Chapter 2 and Chapter 3 are predominately skewed towards American and European cuisine. No attempt to correct for this bias was created, so it is unlikely that models learned from this data will be able to generalize to other kinds of dishes. Correcting bias is a necessary step for future work.

As one final parting comment, if we truly believe that machines should serve humans rather than the other way around, we should carefully consider how to give machines humanistic traits like strong intuition or compassion or kindness. Focusing on strong intelligence with the goal of merely surpassing humans' inference or problem-solving ability should not be the only academically interesting path.

BIBLIOGRAPHY

[1] Yaser S. Abu-Mostafa. Machines that learn from hints. *Scientific American*, 272:64–69, 1995. 24

[2] Sameer Agarwal, Josh Wills, Lawrence Cayton, Gert Lanckriet, David J Kriegman, and Serge Belongie. Generalized non-metric multidimensional scaling. In *International Conference on Artificial Intelligence and Statistics*, 2007. 4

[3] Ehsan Amid and Antti Ukkonen. Multiview Triplet Embedding: Learning Attributes in Multiple Maps. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1472–1480, 2015. 28

[4] Oscar Beijbom, Peter J Edmunds, David I Kline, B Greg Mitchell, and David Kriegman. Automated annotation of coral reef survey images. In *CVPR*. IEEE, 2012. 24

[5] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. *Label Propagation and Quadratic Criterion*, pages 193–216. MIT Press, 2006. 35

[6] Mikhail Bilenko, Sugato Basu, and Raymond J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *ICML*. ACM, 2004. 35

[7] A. Biswas and D. Parikh. Simultaneous Active Learning of Classifiers and Attributes via Relative Feedback. In *CVPR*, June 2013. 27

[8] Damian Borth, Rongrong Ji, Tao Chen, Thomas Breuel, and Shih-Fu Chang. Large-scale visual sentiment ontology and detectors using adjective noun pairs. In *Proc. MM*, 2013. 50, 51

[9] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–Mining Discriminative Components with Random Forests. In *ECCV*. Springer, 2014. 25, 41

[10] Steve Branson, Grant Van Horn, Serge Belongie, and Pietro Perona. Bird species categorization using pose normalized deep convolutional nets. In *BMVC*, Nottingham, 2014. 34

[11] Steve Branson, Grant Van Horn, Catherine Wah, Pietro Perona, and Serge Belongie. The Ignorant Led by the Blind: A Hybrid Human-Machine Vi-

sion System for Fine-Grained Categorization. *IJCV*, 108(1-2):3–29, February 2014. 22, 27

[12] E. J. Crowley and A. Zisserman. In search of art. In *Workshop on Computer Vision for Art Analysis, ECCV*, 2014. 51

[13] Yin Cui, Feng Zhou, Yuanqing Lin, and Serge J. Belongie. Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. In *Proc. CVPR*, 2016. 52

[14] C. Demiralp, C.E. Scheidegger, G.L. Kindlmann, D.H. Laidlaw, and J. Heer. Visual Embedding: A Model for Visualization. *IEEE Computer Graphics and Applications*, 34(1):10–15, January 2014. 22, 24

[15] C.D. Demiralp, M.S. Bernstein, and J. Heer. Learning Perceptual Kernels for Visualization Design. *IEEE Trans. on Visualization and Computer Graphics*, 20(12):1933–1942, December 2014. 22, 27

[16] Sagnik Dhar, Vicente Ordonez, and Tamara L. Berg. High level describable attributes for predicting aesthetics and interestingness. In *CVPR*, 2011. 51

[17] Chen Fang, Hailin Jin, Jianchao Yang, and Zhe L. Lin. Collaborative feature learning from social media. *CoRR*, abs/1502.01423, 2015. 50, 51, 52, 59, 66, 67, 68

[18] Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing objects by their attributes. In *CVPR Workshops 2009*, pages 1778–1785, 2009. 50

[19] Ryan Farrell, Om Oza, Ning Zhang, Vlad I. Morariu, Trevor Darrell, and Larry S. Davis. Birdlets: Subordinate categorization using volumetric primitives and pose-normalized appearance. In *ICCV*, 2011. 33

[20] M. Ferecatu and D. Geman. A statistical framework for image category search from a mental picture. *IEEE TPAMI*, June 2009. 7

[21] A. Frome, Y. Singer, F. Sha, and J. Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *IEEE ICCV*, 2007. 7

[22] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image Style Transfer Using Convolutional Neural Networks. In *Proc. CVPR*, 2016. 51

[23] Shiry Ginosar, Daniel Haas, Timothy Brown, and Jitendra Malik. Detecting people in cubist art. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8925, pages 101–116, 2015. 51

[24] Ryan Gomes, Peter Welinder, Andreas Krause, and Pietro Perona. Crowdclustering. *NIPS*, 2011. 7, 28

[25] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006. 28

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016. 59, 63

[27] Hannes Heikinheimo and Antti Ukkonen. The crowd-median algorithm. In *First AAAI Conference on Human Computation and Crowdsourcing*, 2013. 7

[28] Eric Heim, Matthew Berger, Lee M. Seversky, and Milos Hauskrecht. Efficient Online Relative Comparison Kernel Learning. *arXiv preprint arXiv:1501.01242*, 2015. 42

[29] Gary B. Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In *Workshop on Faces in'Real-Life'Images: Detection, Alignment, and Recognition*, 2008. 11

[30] Hamid Izadinia, Bryan C. Russell, Ali Farhadi, Matthew D. Hoffman, and Aaron Hertzmann. Deep Classifiers from Image Tags in the Wild. In *Proc. Multimedia COMMONS*, 2015. 54

[31] K.G. Jamieson and R.D. Nowak. Low-dimensional embedding using adaptively selected ordinal data. In *Allerton Conference on Communication, Control, and Computing*, 2011. 4, 7

[32] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 34

[33] Brendan Jou, Tao Chen, Nikolaos Pappas, Miriam Redi, Mercan Topkara, and Shih-Fu Chang. Visual Affect Around the World: A Large-scale Multi-

lingual Visual Sentiment Ontology. In *Proc. MM*, pages 159–168, 2015. 51, 54

[34] Sergey Karayev, Matthew Trentacoste, Helen Han, Aseem Agarwala, Trevor Darrell, Aaron Hertzmann, and Holger Winnemöller. Recognizing image style. In *Proc. BMVC*, 2014. 50, 51, 52, 67

[35] Maurice George Kendall. *Rank correlation methods.* Griffin, 1948. 27

[36] Matthus Kleindessner and Ulrike von Luxburg. Uniqueness of Ordinal Embedding. *JMLR*, 2014. 27

[37] Ivan Krasin, Tom Duerig, Neil Alldrin, Andreas Veit, Sami Abu-El-Haija, Serge Belongie, David Cai, Zheyun Feng, Vittorio Ferrari, Victor Gomes, Abhinav Gupta, Dhyanesh Narayanan, Chen Sun, Gal Chechik, and Kevin Murphy. Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://github.com/openimages*, 2016. 50, 51

[38] Jonathan Krause, Benjamin Sapp, Andrew Howard, Howard Zhou, Alexander Toshev, Tom Duerig, James Philbin, and Li Fei-Fei. The unreasonable effectiveness of noisy data for fine-grained recognition. In *Proc. ECCV*, 2016. 52

[39] Neeraj Kumar, Alexander C. Berg, Peter N. Belhumeur, and Shree K. Nayar. Describable visual attributes for face verification and image search. *IEEE Trans. PAMI*, 33:1962–1977, 2011. 11, 51

[40] Shrenik Lad and Devi Parikh. Interactively guiding semi-supervised clustering via attribute-based explanations. In *ECCV*. Springer, 2014. 28

[41] Yong Jae Lee and K. Grauman. Learning the easy things first: Self-paced visual category discovery. In *CVPR*, pages 1721–1728, June 2011. 33

[42] Min Lin, Qiang Chen, and Shuicheng Yan. Network In Network. *arXiv:1312.4400 [cs]*, December 2013. arXiv: 1312.4400. 34

[43] Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Proc. ECCV*, pages 740–755, 2014. 47

[44] Tsung-Yu Lin and Subhransu Maji. Visualizing and understanding deep

texture representations. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 51

[45] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, and Alex C. Berg. SSD: Single Shot MultiBox Detector. In *Proc. ECCV*, 2016. 63

[46] Zhengdong Lu and M.A. Carreira-Perpinan. Constrained spectral clustering through affinity propagation. In *CVPR*, June 2008. 28

[47] Brian McFee. *More like this: machine learning approaches to music similarity*. PhD thesis, University of California, San Diego, May 2012. 4, 7, 22

[48] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. 41

[49] George A Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956. 27

[50] Ishan Misra, C. Lawrence Zitnick, Margaret Mitchell, and Ross Girshick. Seeing through the Human Reporting Bias: Visual Classifiers from Noisy Human-Centric Labels. In *CVPR*, 2016. 55

[51] Naila Murray, Luca Marchesotti, and Florent Perronnin. AVA: A large-scale database for aesthetic visual analysis. In *Proc. CVPR*, 2012. 50, 51, 67

[52] Pere Obrador, Michele A. Saad, Poonam Suryanarayan, and Nuria Oliver. Towards category-based aesthetic models of photographs. In *Proc. MMM*, 2012. 51

[53] Genevieve Patterson and James Hays. Sun attribute database: Discovering, annotating, and recognizing scene attributes. In *Proc. CVPR*, 2012. 50, 51

[54] Kuan-Chuan Peng, Amir Sadovnik, Andrew Gallagher, and Tsuhan Chen. Where do emotions come from? predicting the emotion stimuli map. In *Proc. ICIP*, 2016. 51

[55] Robert Plutchik. The nature of emotions: Human emotions have deep evolutionary roots. *American Scientist*, 89(4):344–350, 2001. 54

[56] Nikhil Rasiwasia, Pedro J. Moreno, and Nuno Vasconcelos. Bridging the gap: Query by semantic example. *IEEE Trans. Multimedia*, 9:923–938, 2007. 50

[57] Miriam Redi and Bernard Mérialdo. Enhancing semantic features with compositional analysis for scene recognition. In *ECCV Workshops*, 2012. 51

[58] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *CVPR 2016*, pages 779–788, 2016. 63

[59] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014. 27

[60] Walter J. Scheirer, Neeraj Kumar, Peter N. Belhumeur, and Terrance E. Boult. Multi-attribute spaces: Calibration for attribute fusion and similarity search. In *CVPR*, 2012. 51

[61] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. *arXiv:1503.03832 [cs]*, March 2015. arXiv: 1503.03832. 22, 27, 28

[62] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. *arXiv:1409.4842 [cs]*, September 2014. arXiv: 1409.4842. 34, 59

[63] Omer Tamuz, Ce Liu, Serge Belongie, Ohad Shamir, and Adam Tauman Kalai. Adaptively learning the crowd kernel. In *ICML*, 2011. 4, 5, 7, 9, 18, 26, 30

[64] Wei Tang, Hui Xiong, Shi Zhong, and Jie Wu. Enhancing Semi-supervised Clustering: A Feature Projection Perspective. In *SIGKDD*, 2007. 28

[65] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *JMLR*, 9(2579-2605):85, 2008. 26, 29, 30

[66] Laurens Van der Maaten and K. Weinberger. Stochastic triplet embedding.

In *IEEE Int. Workshop on Machine Learning for Signal Processing*, 2012. 4, 7, 11, 12, 22, 24, 26, 29, 30, 42

[67] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 27, 33

[68] Catherine Wah, G. V. Horn, Steve Branson, Subhransu Maji, Pietro Perona, and Serge Belongie. Similarity Comparisons for Interactive Fine-Grained Categorization. *CVPR*, 2014. 5, 9, 22, 27, 71

[69] Jiang Wang, Yang song, Thomas Leung, Chuck Rosenberg, Jinbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning Fine-grained Image Similarity with Deep Ranking. *arXiv:1404.4661 [cs]*, April 2014. arXiv: 1404.4661. 28

[70] Xiang Wang, Buyue Qian, and Ian Davidson. On constrained spectral clustering and its applications. *Data Mining and Knowledge Discovery*, 28(1):1–30, September 2012. 35

[71] Michael J Wilber, Iljung S Kwak, and Serge J Belongie. Cost-effective hits for relative similarity comparisons. In *AAAI Conference on Human Computation and Crowdsourcing*, 2014. 24, 42

[72] Jeremy M. Wolfe. Guided search 2.0 a revised model of visual search. *Psychonomic Bulletin & Review*, 1, June 1994. 9

[73] Pengcheng Wu, Steven C.H. Hoi, Hao Xia, Peilin Zhao, Dayong Wang, and Chunyan Miao. Online Multimodal Deep Similarity Learning with Application to Image Retrieval. In *ACM International Conference on Multimedia*, 2013. 28

[74] Eric P. Xing, Michael I. Jordan, Stuart Russell, and Andrew Ng. Distance metric learning with application to clustering with side-information. In *NIPS*, 2002. 28

[75] Jinfeng Yi, Rong Jin, Shaili Jain, and Anil Jain. Inferring users preferences from crowdsourced pairwise comparisons: A matrix completion approach. In *HCOMP*, 2013. 7

[76] Jinfeng Yi, Rong Jin, Shaili Jain, Tianbao Yang, and Anil K. Jain. Semi-

crowdsourced clustering: Generalizing crowd labeling by robust distance metric learning. In *NIPS*, 2012. 28

[77] Jinfeng Yi, Lijun Zhang, Rong Jin, Qi Qian, and Anil Jain. Semi-supervised Clustering by Input Pattern Assisted Pairwise Similarity Matrix Completion. In *ICML*, 2013. 28

[78] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop, 2015. arXiv:1506.03365. 52, 55

[79] Liwen Zhang, Subhransu Maji, and Ryota Tomioka. Jointly Learning Multiple Perceptual Similarities. *arXiv:1503.01521 [cs, stat]*, March 2015. arXiv: 1503.01521. 28, 42