COMPUTER VISION FOR SPATIO-TEMPORAL ANALYSIS OF INTERNET PHOTO COLLECTIONS

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

Kevin Matzen

August 2016

© 2016 Kevin Matzen
ALL RIGHTS RESERVED

COMPUTER VISION FOR SPATIO-TEMPORAL ANALYSIS OF INTERNET PHOTO COLLECTIONS

Kevin Matzen, Ph.D.

Cornell University 2016

The advent of the digital camera and subsequently the smartphone has ushered in an unprecedented age of photography where a large fraction of the human population has an Internet-enabled camera in their pocket. What is perhaps more remarkable is that many of these people are willing to share their experiences publicly with the rest of the world by uploading them to social media platforms such as Flickr, Facebook, and Instagram; roughly billions of photos per day. However, making use of these photos in a shared setting that leverages the uniquely visual aspect of this medium is non-trivial. What does this photo contain? Where was this photo taken? What was the structure of the scene? What trends can we observe? What is the story behind this photo? These are the sorts of questions one might like to answer using *computer vision* applied to an immense corpus of imagery, effectively peering through billions of windows into the world with automation.

This dissertation presents three methods for doing such large-scale analyses to help a human analyst understand properties of the real world related to space and to time by automatically analyzing, cataloging, and visualizing Internet-scale photo collections, evaluated on millions or tens of millions of photos, but designed to scale horizontally for modern data processing platforms. First I present a method for aggregating millions of photographs of some physical space in the world (e.g. a city) and building a 3D reconstruction that has time-

varying appearance capturing the evolution of that 3D space over time. This method includes a segmentation algorithm to recover temporally consistent elements and these elements can in turn be detected in new imagery to predict when the photo was taken. Next I present a method for aggregating millions of photographs and using state-of-the-art convolutional neural networks to mine for small *discriminative patches* in the imagery. These patches are designed to be discriminative in the sense that if the goal is to classify an image into one of two categories, then these patches could be used in lieu of the full image. Experiments validate the method on a wide variety of datasets and tasks, but of most relevance is the application to informing an analyst what visual elements differentiate one city from another in terms of building architecture or geo-location and time from the observed fashion style. Finally, I present a method and in depth case study for using millions of photographs to identify and visualize fashion style trends across the world and across years.

BIOGRAPHICAL SKETCH

Kevin Matzen received his M.S in Computer Science from Cornell University in 2015, and his B.S.E. in Computer Science Engineering and B.S.E. in Computer Engineering from the University of Michigan in 2010.

To my parents.

ACKNOWLEDGEMENTS

I would like to thank my advisors Noah Snavely and Kavita Bala for their encouragement and guidance all throughout my graduate studies. Not only did Noah and Kavita help me develop as a researcher in computer vision, but they also helped me develop as a person. It's their constant encouragement of creativity that helped me learn how to craft a vision for the future.

The Cornell graphics and vision group is both a wonderfully creative and technically skilled set of people. In particular, I'd like to thank Sean Bell, Paul Upchurch, Daniel Hauagge, Pong Khungurn, Balázs Kovács, Scott Wehrwein, Kyle Wilson, Albert Liu, Ivo Boyadzhiev, and Chun-Po Wang for all the great discussions and collaborations over the years. I'd also like to thank members of the larger Cornell computer vision family, in particular Andy Gallagher, Adarsh Kowdle, Amir Sadovnik, and Suren Jayasuriya. Finally, I'd like to thank Chuck Yang for his help on several projects throughout the years.

A special thanks to Professor Peter Chen. Peter ignited my passion for computer science by teaching one of my very first classes at the University of Michigan. This formative experience is why I now ask myself, What cool things should I build today?

And finally, thanks to my parents who have been loving and supportive throughout the years.

TABLE OF CONTENTS

Biographical Sketch . . . iii
Dedication . . . iv
Acknowledgements . . . v
Table of Contents . . . vi
List of Tables . . . viii
List of Figures . . . ix

1 Introduction 1

2 Time-varying reconstruction of places from photos 5
2.1 Related Work . . . 8
2.2 Overview . . . 10
2.3 Method . . . 13
2.3.1 Dense 3D Reconstruction . . . 13
2.3.2 Computing the Visibility Graph . . . 14
2.3.3 Initial Time Interval Estimation . . . 16
2.3.4 Spatio-Temporal Graph . . . 18
2.3.5 Plane-Time Clustering . . . 20
2.4 Experiments and Results . . . 24
2.4.1 Datasets . . . 24
2.4.2 Viewing and Rendering . . . 25
2.4.3 Timestamp Prediction . . . 28
2.4.4 Bad Timestamp Detection . . . 30
2.5 Limitations . . . 32
2.6 Conclusions . . . 33

3 Discovering visual correlations in labeled photo collections 34
3.1 Background and Related Work . . . 37
3.2 Method . . . 41
3.2.1 Patch Discovery . . . 41
3.2.2 Patch Clustering . . . 44
3.3 Datasets . . . 46
3.4 Experiments . . . 49
3.4.1 Raw Classification Performance . . . 49
3.4.2 Qualitative Results . . . 53
3.4.3 Discriminability . . . 55
3.4.4 Localization . . . 57
3.4.5 Exploring Visual Data . . . 59
3.5 Limitations . . . 63
3.6 Conclusions . . . 63

4 Measuring human trends with social media images **65**
4.1 Related Work 69
4.2 Data 72
4.3 Methodology 78
4.3.1 Learning attributes 78
4.3.2 Measuring attributes at scale 81
4.3.3 Visually consistent style clusters 83
4.4 Exploratory Analysis and Applications 85
4.4.1 Analyzing Trends 85
4.4.2 Visualizing Styles 95
4.4.3 Artist Reference Material Application 105
4.5 Limitations 108
4.6 Future Work 109

5 Conclusions **111**

Bibliography **113**

LIST OF TABLES

2.1 Dataset statistics. 24

3.1 **Dataset synopsis.** I validate our method on a variety of datasets. Note that InstagramPeople does not come with explicit categories for classification, but instead an expressive set of metadata from which both classification and regression tasks can be constructed (e.g., by geolocation or timestamp). 46

3.2 **Raw classification performance.** A synopsis of classifier performance for different types of input (normal, foveal, and bubble) and each stage of fine-tuning. *Foveal random* indicates that at test time, a single, randomly-sampled focal location is used to process the input that the classifier sees. *Foveal best* and *bubble best* indicate that the network is allowed to search for the focal location that produces the lowest entropy prediction distribution. *Fine-tuned from [32] reference CaffeNet. †Fine-tuned from [32] reference GoogLeNet. ‡Original [32] reference CaffeNet. §Original [32] reference GoogLeNet. ¶Fine-tuned from [94] Hybrid-CNN. ‖Original [94] Places205-CNN. 51

4.1 Number of MTurk annotations collected for binary attributes indicating the presence of jackets, collars, scarves, necktie, hat, glasses, and multiple layers of clothing. 76

4.2 Attributes with multiple values include the type of clothing, sleeve length, neckline shape, clothing pattern, and major color. In parentheses are the number of MTurk annotations I collected for each attribute. 76

4.3 Test set performance for the trained CNN. Since each attribute has an unbalanced set of test examples, I report both accuracy and mean classification accuracy. Included are baseline performance scores for random guessing and majority class guessing for reference. 81

LIST OF FIGURES

2.1 A montage of a planar region of the 5 Pointz dataset, showing how this region has changed over time as different pieces of art have been overlaid atop one another. 7

2.2 System pipeline. Given a dense 3D reconstruction (1), I expand a visibility graph (2), and then use that graph to estimate per-patch time intervals (3). Using spatial and temporal cues, I construct a spatio-temporal graph and perform an under-segmentation (4). Then I fit plane-time cuboids to this graph (5) and generate texture maps for visualization (6). This example shows an exploded view of several billboards at the same location over time. 12

2.3 An example of the same physical space in Times Square, New York undergoing change in appearance over time. Each billboard forms a single unit that I wish to recover automatically. . . 13

2.4 Time profiles, shown at right, for two different appearances of the same wall, shown at left. Green marks on the timeline indicate positive observations and red marks indicate negative observations. Dark gray intervals indicate the shortest interval that has the maximal F_1 score. Since I use a discrete set of observations, any interval with endpoints contained within the light gray regions will have the same F_1 score. Note the false positive red marks inside the gray intervals and the false negative green marks outside the gray intervals, as well as the very non-uniform distribution of observations. The use of the F_1 score based on classification accuracy is motivated by the need to robustly handle noisy, non-uniform observations in a more effective way than simple percentile-based methods. 18

2.5 A texture mapped rendering of `5Pointz` at two different points in time. 25

2.6 An assortment of plane-time cuboids from `TimesSquare` (blue), `Akihabara` (green), and `5Pointz` (red) with texture maps and estimated time intervals produced by the system. 26

2.7 `5Pointz` timestamp prediction results. Positive and negative observations provide clues as to when a photo was taken. Positive observations for a sample photo are shown on the right in green. Several components of this scene are well-localized in time and can be used to build a timestamp scoring function. The timestamp with the highest score is marked on the timeline. I predict this photo was taken on July 25, 2013 and the photo metadata indicates it was taken on August 10, 2013. 28

2.8 Retimestamped photos. The timeline indicates the timestamp scoring function with the given timestamp indicated in blue and the predicted timestamp indicated in pink. The originally recorded date and estimated date range as determined by manual inspection are provided above each image. 31

3.1 **Foveated imaging.** This method analyzes the contribution of localized content in images by modifying the data shown to a CNN during training and at test time. Given an input image (top left), I apply an alpha mask (center) to produce two representations: a *foveal image* (top right) and a *bubble image* (bottom right). While a *bubble image* preserves a small amount of spatialized content, a *foveal image* includes a blurred, grayscale background to provide additional context. 36

3.2 **StreetViewCities dataset.** Several tiles shown from various cities. The number of images per city varies from 102,192 (Venice) to 964,703 (Tokyo). Each 360° spherical panorama has been cropped to produce 7 tiles. Can you guess each city?[1] 47

3.3 **InstagramPeople dataset.** Several examples of cropped people from the InstagramPeople dataset. Networks were trained on crops (shown resized) to network input dimensions. 48

3.4 **StreetViewCities confusion matrix.** StreetViewCities is a new dataset built from millions of StreetView photos. The GoogLeNet model obtains 69% accuracy on the test set. 50

3.5 **Extracted patch clusters.** Patches extracted from clusters with low entropy. I show one example in the lower left of each cluster with a mask indicating the size of the Gaussian bubble used during discovery. The majority class label is shown below each cluster. 54

3.6 **Indoor67 evaluation.** *Top*: The patch selection method allows us to select a very small set of semantic elements as a representation, yet still retain good classification performance.*Bottom*: I validate two alternate BubbleBank design choices. (1) I take the BubbleBank feature vector (a) or PCA applied to the penultimate layer of the CNN (b) and preserve the top N features of each descriptor while setting the remaining to zero. For example, $N = 1$ would produce a feature vector with exactly 1 non-zero located in the position of the component with largest magnitude. (2) I use only the top N ranked bubble features given by (a) a random ranking or (b) the method's ranking. 55

3.7 **Part localization.** Normalized distance to closed part for the bubble location that maximizes correct class label on CUB-200-2011. The bubble method chooses locations closer to part annotations more often than random guessing. 57

3.8 **Occlusion analysis.** Given a AlexNet/GoogLeNet classifier, I analyze how much the classification score is impacted by occluding parts at random and as guided by our discriminative bubbles on three datasets. First a discriminative bubble is mined for each photo and then an occluding box is placed at its position. The photo's classification is reevaluated. 58
3.9 Three cities with discriminative bubble cluster activations. Left to right: motorbikes in Jakarta, houses with siding in Boston, and windows with metal railings in Paris. 59
3.10 **Finding temporal correlations.** *Top*: NYC one-versus-one month classification accuracies. *Bottom*: I found sunglasses to be a discriminative element that makes April in NYC visually distinct from March and correlated them to weather reports. 60
3.11 **NYC vs. London patches.** I used my method to find what makes people in NYC visually distinct from London. I show clusters correlated with NYC (top) and London (bottom). 62

4.1 t-SNE visualization of learned feature space. The style embedding is 1024 dimensions and is visualized in 2 dimensions here. Several combinations of attributes are highlighted. For example, white shirts with text, plaid shirts, black v-neck shirts, etc. are all styles discoverable in this embedding. 68
4.2 Given an approximate person detection and a precise face localization, I generate a crop with canonical position and scaling to capture the head and torso of the person, shown left. Additional examples from our dataset are shown on the right. 74
4.3 Geo-spatial distribution of images collected from Instagram. Images were collected according to two distributions: a set of the world cities listed (yellow dots), and distributed over the entire Earth (blue). 75
4.4 Confusion matrices for each of the clothing attributes. Given class labels are vertical and predicted class labels are horizontal. Each row has been normalized such that a cell indicates the probability of a classification given some true label. 80
4.5 Reliability curves for a few of the clothing attributes. A well-calibrated reliability curve has fraction of positives equal to mean predicted value. Neural networks tend to produce well-calibrated probabilities, but some minor improvement is made using Isotonic regression for additional calibration. 82

4.6 Mean major color score for several different colors. The means are computed by aggregating measurements per 1 week intervals and error bars indicate 95% confidence interval. Colors such as brown and black exhibit unified seasonal patterns whereas colors like blue and white are the opposite. Colors such as gray are trending up year over year whereas red, orange and purple are trending down. 86
4.7 Mean major color score for the color white for two cities highlighting flipped behavior between Northern Hemisphere and Southern Hemisphere. Mean is computed by aggregating measurements per 1 week intervals and error bars indicate 95% confidence interval. 87
4.8 Pearson's correlation coefficients for the mean prediction score of the color white across all pairs of major world cities. Labels have been permuted by latitude to highlight the relationship between the northern and southern hemispheres and their relationship to the seasonal trend of the color white. 88
4.9 Distribution of Instagram filters used in the dataset. More than half of the dataset has no filter applied. Popular filters near the end of the dataset include the filter "Clarendon". Filters popular several years earlier such as "Valencia" have declined significantly over the years. 89
4.10 Average image for each month in the dataset with per-pixel mean subtracted and scaled by per-pixel standard deviation. Top row: 2013, Bottom row: 2016. Over time photos uploaded to Instagram have been progressively changing in mean color from reddish to greenish. 90
4.11 Average image for each month in the "filter=Normal" subset with per-pixel mean subtracted and scaled by per-pixel standard deviation. Top row: 2013, Bottom row: 2016. Over time photos uploaded to Instagram have been progressively changing in mean color from reddish to greenish despite considering only those with no filter applied. 91
4.12 Scarf trends in the USA. The Instagram-based measurements are compared to Google Trends. Instagram measurements are derived from mean scarf score aggregated by week with error bars indicating 95% confidence interval. Google Trends signal is per-week "q=scarf, geo=US" query volume normalized by total query volume. 92
4.13 Mean jacket score per country. Countries with $\geq$ 1000 photos included. Countries with < 1000 photos shown in gray. 93
4.14 Mean scarf score per country. Countries with $\geq$ 1000 photos included. Countries with < 1000 photos shown in gray. 94

4.15 Mean hat score per country. Countries with ≥ 1000 photos included. Countries with < 1000 photos shown in gray. 95
4.16 Several top-ranked examples of hats in Oman. While not strictly hats, it's clear that the classifier has learned to identify more generally, head coverings. 95
4.17 Images closest to the centers of several mixture components in our GMM. Clusters are presented in no particular order. 96
4.18 Top ranked GMM clusters according to entropy across cities. . . 98
4.19 High-ranking spatio-temporal cluster illustrating that hijabs are a popular garment in Cairo and Jakarta. 99
4.20 High-ranking spatio-temporal cluster illustrating that dark jackets and glasses are a popular combination during the winter in several cities that typically experience cold winters. 100
4.21 High-ranking spatio-temporal cluster illustrating that white t-shirts with graphic designs and glasses are a popular combination during summer months, but popular in Singapore throughout the entire year. 101
4.22 High-ranking spatio-temporal cluster illustrating that yellow t-shirts with graphic patterns were very popular for a very short period of time, June-July 2014, in specifically Bogotá, Karachi, Rio de Janeiro, and São Paulo. 102
4.23 High-ranking spatio-temporal cluster illustrating that blue polo shirts are popular in Cairo and throughout the Indian subcontinent. 103
4.24 High-ranking spatio-temporal cluster illustrating that black polo shirts with glasses are popular in Singapore and throughout the Indian subcontinent. 104
4.25 Fashion sketches of Mumbai used in study. Row 1: Mumbai, Artist 1, random reference material, Row 2: Mumbai, Artist 1, using style clusters, Row 3: Mumbai, Artist 2, random reference material, Row 4: Mumbai, Artist 2, using style clusters. 107
4.26 Fashion sketches of Jakarta used in study. Row 1 left: Jakarta, Artist 1, random reference material, Row 1 right: Jakarta, Artist 1, using style clusters, Row 2 left: Jakarta, Artist 2, random reference material, Row 2 right: Jakarta, Artist 2, using style clusters. 108

CHAPTER 1

INTRODUCTION

The advent of the digital camera and subsequently the smartphone has ushered in an unprecedented age of photography where a large fraction of the human population has an Internet-enabled camera in their pocket. What is perhaps more remarkable is that many of these people are willing to share their experiences publicly with the rest of the world by uploading them to social media platforms such as Flickr, Facebook, and Instagram; roughly billions of photos per day. However, making use of these photos in a shared setting that leverages the uniquely visual aspect of this medium is non-trivial. What does this photo contain? Where was this photo taken? What was the structure of the scene? What trends can we observe? What is the story behind this photo? These are the sorts of questions one might like to answer using *computer vision* applied to an immense corpus of imagery, effectively peering through billions of windows into the world with automation.

Before we can quantify the contents of a collection of photos, we need to determine what properties should be quantified in the first place. How we can begin to talk about the rich structure in any given photo? Text-based analysis has a natural set of atoms—a natural language vocabulary. A document can be described by the frequencies of certain sets of words. This gives us a way to measure trends in documents using text. What can be said of images? Is the enumeration of objects in the photo the most informative? The physical 3D structure of the scene? What about patterns and trends that only emerge in the collection as a whole? I claim that there is no single universal representation and that the correct representation is highly task-dependent. Whether we wish

to identify the species of bird we've photographed or we wish to find clothing trends in social media, the relevant *visual vocabulary* will be significantly different. Furthermore, to scale to the plethora of potential visual questions, this vocabulary needs to be learned from data. Throughout this dissertation I will describe very different problems and with each their own visual vocabulary.

This dissertation focuses on automatically building these visual vocabularies from large-scale visual corpora for questions related to two pieces of metadata commonly included with photos on social media platforms—geotags and timestamps. Each photo in a collection is an observation of the real world, localized in space and time. By reasoning about how different periods of time or locations visually differ, we can derive relevant atoms and measure their frequency. Imagine a future where trillions of photos across a century of human history can automatically be mined for the visual elements related to the transformation and geographic distribution of human stylistic concepts such as architecture, art, fashion, and design. In the context of understanding the evolution of physical places, this is the basis for an *Automatic Archaeologist*, a system for automatically reconstructing and recovering the past of our ever-changing world. In the context of understanding people, this is the basis for an *Automatic Anthropologist*, a system for automatically measuring human trends in how we present ourselves throughout the world.

Chapter 2 presents a method that uses massive numbers of timestamped photos to catalog the visual evolution of a 3D scene over time. This is challenging because a standard assumption in multi-view stereo and structure from motion work is that the scene to be reconstructed must be static. This problem of *4D reconstruction* from unconstrained Internet photo collections is even more

difficult as metadata can be quite noisy and modern vision methods can make errors. This motivates robust methods for reasoning about temporal observations. I propose an algorithm that relaxes the static world assumption. Briefly, the method takes as input a Flickr keyword query such as "times square", collects photos from the search results, reasons about camera geometry, scene geometry, and scene appearance, and produces a point cloud that can be explored in 3D or across time. Furthermore, I segment these 4D spatio-temporal point clouds to produce *4D space-time segments*, a physical 4D vocabulary. These vocabulary elements can be visualized and detected in novel photography to enable tasks such as timestamp prediction.

Chapter 3 presents a method for analyzing distinctive visual elements of categorized photography using convolutional neural networks. Briefly, given a massive corpus of imagery with some set of discrete labels, one per image, this method seeks to find small "bubbles", localized patches in the input imagery, that provide the best or most certain classifications. These patches are an automatic visual vocabulary for any discriminative task. This work is the combination of state-of-the-art Convolutional Neural Networks (CNNs) to build a robust, discriminative functions and an experimental methodology derived from human visual perception literature to extract insight from the structure of these CNNs. These bubbles are further analyzed to find sets of concepts across the corpus that can be measured and tracked to find geospatial and temporal patterns.

Finally, Chapter 4 presents a world-scale case study analyzing the social photography platform, Instagram, to find trends in millions of photos of people around the world and across several years. In particular, I look at patterns in

the types and styles of clothing worn by these people. A visual vocabulary is built By annotating a relatively small subset of examples and making large-scale measurements with Convolutional Neural Networks. I use this wealth of data and associated vocabulary to find trends related to season, events, and cultural significance.

I conclude with my future vision for this line of work in Chapter 5. I believe that in order to make the best use of large-scale visual data, visual vocabularies will be a key component for describing and understanding the people, places, and things of the world automatically from the web.

CHAPTER 2

TIME-VARYING RECONSTRUCTION OF PLACES FROM PHOTOS

The 3D reconstruction problem takes as input a set of photographs and produces some 3D representation of the scene such as a point cloud or a polygonal mesh. I will start by introducing a related problem, 4D reconstruction, where the world is allowed to change in appearance over time. The goal at hand is to build a representation, a visual vocabulary, for visually changing scenes. The work has been presented in the European Conference on Computer Vision 2014 [43] and received a best paper award.

Over the last few years, our ability to build 3D models of the world around us has exploded due to the availability of vast numbers of photographs from sources such as Flickr, Facebook, and Picasa, as well as new reconstruction algorithms that can run on Internet-scale collections [1, 20]. Indeed, the concept of "Internet-scale" has been stretched to match the scale of the world, as planet-sized reconstructions are now being built [36]. Once every stone has been turned to a point in a point cloud, does that mean the problem of 3D reconstruction is complete? While we often refer to scenes reconstructed via methods such as structure from motion (SfM) as being "static," in reality many urban scenes are quite dynamic over time scales of even a few years. Signs, lights, decorations, and street art are examples of such dynamism—they are often reconstructable with current methods, but transient in a longer-term view. Such visual elements provide opportunities for us to reconstruct and visualize our ever-changing world.

However, state-of-the-art 3D reconstruction methods largely remain agnostic to the time domain. While some systems work out of the box when ap-

plied to dynamic scenes, they usually provide an output model devoid of any temporal information, where points from different time spans simply co-exist. Other systems actively prune multiple points that occupy the same spatial location, which for time-varying scenes leads to broken reconstructions, veritable chimeras of different objects or appearances conflated in time.

My aim is to bring temporal meaning back to these reconstructions. I propose a simple approach by which output from an existing 3D reconstruction method can be taken as input and segmented into distinct space-time objects, each with a specific extent in time. This takes the web of observations and untangles it into temporally consistent components. My approach not only augments a reconstruction with temporal information, but also leverages the time dimension as a unique signal for segmenting scenes into meaningful objects; just as object motion is an important cue for video segmentation, the temporal structure of a model is a useful cue for segmenting scenes.

The underlying application of segmentation here is to build a visual vocabulary for the evolution of physical places. A straightforward application is to visualize the reconstructed scene. That is, for each snapshot in time, we can render the segments, the vocabulary, of the scene to produce an interactive visualization. Additionally, by reasoning about which segments are visible in a photo, we can estimate when that photo must have been taken. Finally, the long term vision is to build a system that can reason about spatial, temporal, and appearance-based relations between these elements in the visual vocabulary. Just as an archaeologist would both recover some physical structure of the scene as well as understand it in the larger historical context, such a visual vocabulary could be analyzed by an *Automatic Archaeologist*.

Figure 2.1: A montage of a planar region of the 5 Pointz dataset, showing how this region has changed over time as different pieces of art have been overlaid atop one another.

My method works by analyzing two key pieces of information: (1) the visibility relationship between images and points in a reconstruction, and (2) the timestamps on the images themselves. However, due to noise, it is difficult to use this raw information to directly reason about time. Incorrect timestamps, registration failures, uncontrolled illumination, photographer bias, sporadic temporal sampling, and photo manipulation all make temporal reasoning a challenging task. This work is inspired by previous methods on temporal inference on 3D scenes [61, 60], but my approach differs especially in *scale* and *granularity*—my method is designed for handling tens to hundreds of thousands of images sampled relatively densely in time. When reasoning about time in image collections, every observation has something to say, and so I focus on methods that are highly scalable to the billions of individual observations in our datasets.

I demonstrate our method on several large-scale datasets, including dynamic city scenes, such as Times Square, as well as 5 Pointz, an outdoor street art exhibit space in Queens, NYC. Figure 2.1 shows a region of our 5 Pointz dataset,

demonstrating the kinds of outputs we produce—extracted regions, sometimes overlapping in space, each with a temporal span. I use the results of my temporally segmented scenes in two applications: a system for visualizing scenes over time, and a method for timestamping a new photo based on its contents. More broadly, I believe that this system can be used as the basis for *temporal visual discovery*—allowing one to automatically chronicle the rich visual appearance of a scene over time leveraging the wealth of online imagery, and to explore and visualize the results.

2.1 Related Work

Dynamic scene analysis. Scenes can be dynamic at many different time scales—seconds, minutes, years, decades—and some time scales have received more attention in the literature than others. At video-rate time scales, there has been significant work on reconstruction of moving objects, spanning problem domains including motion capture [62], non-rigid SfM [8], scene flow methods [79, 78], and video browsing [4]. Recent work has also sought to automatically *sequence* unstructured images captured over short time scales, based on motion constraints [10]. Somewhat longer time scales can be captured with time-lapse video, which only captures a single viewpoint but is especially useful for studying changes due to illumination [70, 31] or long-term motions of objects [56]. This domain of dynamic reconstruction methods from unstructured imagery at the scale of several years has received relatively little attention in the community, but is becoming more and more feasible as the number of available images of scenes grows ever larger (imagine a future scenario where 100 years' worth of densely sampled imagery is available for any scene!)

There are a few systems that operate on unstructured imagery over large time scales. Probably most related to my work is that of Schindler and Dellaert [60] on probabilistic temporal reconstructions of scenes from historical photo collections. By analyzing occluder-occludee relationships of buildings and considering timestamps of varying precision, they formulate a graphical model and perform joint inference to determine the most likely temporal ordering for a set of photographs, a set of improved timestamps for these photos, and temporal extents for the observed buildings in the scene. My work differs from Schindler and Dellaert in several key respects. First, their analysis was conducted on photo collections containing hundreds of photos, and performs an inference involving potentially expensive steps, such as an occlusion-test subroutine. In contrast, a key goal of mine is to scale to very large collections of images (my largest input set includes over 1 million photos). Second, I focus on more fine-grained temporal reasoning, involving photos sampled densely in time over the last ten years. I also focus primarily on changes in *appearance* (e.g., one billboard changing to another), rather than changes in geometry, resulting in a different space of design decisions. Finally, their work performs an initial geometric object segmentation based on proximity and a simple camera co-visibility metric. In my work, I integrate time-based reasoning directly into the scene segmentation, avoiding hard segmentation decisions early on.

Change detection. A related field is that of change detection, where the goal is to identify differences between two or more 2D images or 3D scenes. Pollard and Mundy proposed a change detection method for 3D scenes observed from an aerial vehicle [54]. By building a model of occupancy and appearance over a voxel grid, they could determine if a new view was generated either by the background model or by some change. Other related work proposes geomet-

ric change detection by comparing probabilistic 3D reconstructions built from videos from ground vehicles [74, 75]. However, these formulations only produce differences between two points in time. My work provides a full temporal model with which the scene's appearance at an arbitrary point in time could potentially be predicted, given enough data relevant to that period in time. Fard et al. proposed a method for using unstructured photo collections to monitor the progress of a building under construction by comparing 3D reconstructions to 3D building specifications [18]. In our case, however, I have no a priori scene model.

A fundamental principle behind my system design is to scale to millions of images in order to robustly handle temporal outliers as well as to provide high temporal resolution. Towards this goal, I have devised a graph-based temporal segmentation algorithm that is simple and scalable in its design.

2.2 Overview

My overall goal is to build a dense 3D point cloud with appearance information, e.g., using methods such as PMVS [22], and then to segment this reconstruction into a set of temporally coherent geometric components (corresponding to signs, billboards, storefronts, etc.), as well as estimate a time range $[t_{\min}, t_{\max}]$ for each component defining its temporal extent. I use the timestamps on images for this task. However, due to noise in timestamps and in the estimates of point visibility, this task is difficult to perform robustly. In addition, due to the scale of our data, I require very efficient algorithms.

Assumptions and Scene Representation My method makes several assump-

tions about the input scene. First, I apply a standard SfM [67] and multi-view stereo (MVS) [22] pipeline (with a few modifications) to derive an initial reconstruction from a set of photos. Hence, even for a dynamic scene, I assume we can reconstruct accurate cameras and points using SfM. This assumption relies on there being sufficient scene content that is constant over time, or at least there being sufficient continuity in a scene from one time to the next. Second, I assume that the geometry of the scene is largely static, but that the appearance can change dramatically from one time to the next. Such appearance changes are a key source of dynamism in scenes like Times Square, where storefronts, billboards, and other ads are changing, often from one week to the next. Finally, because I focus on urban scenes, I assume that the scene geometry can be well-approximated with a set of planes, as in other work on urban reconstruction [66, 21].

This last assumption leads to my choice of output scene representation. I assume that the components I segment the scene into can be approximated as (not necessarily axis-aligned) planar facets—i.e., at any given time, the 3D scene is approximated by 2D planes. Further, as in Schindler et al. [60], I assume that unique objects come into existence at some time, exist for a while, then are removed or replaced permanently. Hence, each object exists for a single, contiguous span of time. If one considers a dynamic scene as a 4D space-time volume, then these planar objects form 3D space-time "cuboids" in this volume—2D in space, 1D intervals in time. Hence, my output representation consists of a set of such objects, which I refer to as "plane-time cuboids."

Space-time Reconstruction System My overall pipeline is illustrated in Figure 2.2. I take in a large collection of Internet photos of a scene, and generate

Figure 2.2: System pipeline. Given a dense 3D reconstruction (1), I expand a visibility graph (2), and then use that graph to estimate per-patch time intervals (3). Using spatial and temporal cues, I construct a spatio-temporal graph and perform an under-segmentation (4). Then I fit plane-time cuboids to this graph (5) and generate texture maps for visualization (6). This example shows an exploded view of several billboards at the same location over time.

a complete dense 3D point cloud reconstruction (Fig. 2.2(1)), which initially is the union of all points across time. Next, I reason about which cameras observe which points (2), and use the observation profile of each point to reason about the time interval in which it existed (3). I then segment the scene into sets of points that are spatio-temporally coherent ((4) and (5)), from which I can generate visualizations of the appearance of each part of the scene over time (6). Figure 2.3 shows an example of the same physical space undergoing change in

Figure 2.3: An example of the same physical space in Times Square, New York undergoing change in appearance over time. Each billboard forms a single unit that I wish to recover automatically.

appearance over time. I now describe this space-time reconstruction and segmentation algorithm in detail.

2.3 Method

2.3.1 Dense 3D Reconstruction

The input is a large set—tens of thousands to millions—of photographs downloaded from Flickr using targeted keyword searches such as `times square` or `abc studios`. I restrict the download to the set of photos where the `datetaken` timestamp is provided by Flickr. Then, using a random subset of approximately 20,000 of these photos, I build an initial SfM reconstruction using Bundler to obtain camera parameters and a sparse 3D point cloud [68]. I use 2D-to-3D matching techniques to register the remaining photos to the reconstruction [40]. This process results in a set of reconstructed images $\mathcal{I}$, camera parameters for each image, and a sparse set of 3D points. Finally, I use CMVS to factor the reconstruction into smaller components [22], and then use a mod-

ified version of PMVS to generate a dense 3D point cloud [23]. Briefly, PMVS contains a number of filters on the geometry it produces that disallow two 3D points from occupying the same (or very similar) spatial location. I modify their implementation to remove this restriction on output geometry. The result of CMVS+PMVS is a dense set of 3D patches $\mathcal{P}$. Each patch $p \in \mathcal{P}$ is represented with a 3D position, a surface normal, and a set of images $V_{\text{PMVS}}(p) \subseteq \mathcal{I}$ in which p is deemed to be visible (as determined by high photoconsistency with a reference patch for p). One of these views is selected as a reference view.

2.3.2 Computing the Visibility Graph

My system relies on a good initial estimate of the visibility of each patch p in the set of cameras, as visibility (along with timestamps on images) provides strong information about when in time a patch existed. The visibility sets V_{PMVS} produced by PMVS are very approximate, and so I expand these sets into a richer representation of visibility. I distinguish between two notions of visibility in this work, which I call *photoconsistent-visibility* and *projection-visibility*:

1. A patch p is *projection-visible* in view $v \in \mathcal{I}$ if p is in front of v, p projects inside the field of view of v, and p passes an occlusion test with respect to other patches.
2. A patch p is *photoconsistent-visible* in view $v \in \mathcal{I}$ if p is projection-visible in view v and further p satisfies a photoconsistency criterion in view v.

I use these definitions to further define notions of "positive" and "negative" observations of a patch p in an image v, encoded as a *visibility graph*. The visi-

bility graph $\mathcal{V}$ is a bipartite graph $(\mathcal{I}, \mathcal{P}, \mathcal{E})$ with edges $e \in \mathcal{E}$ connecting views to patches. $\mathcal{V}$ has two types of edges. A *positive observation* edge e_{ij+} encodes a temporal event where p_j is photoconsistent-visible view v_i. A *negative observation* edge e_{ij-} encodes a temporal event where view v_i *should* have observed p_j under projection-visibility (i.e., p_j is projection-visible to v_i), but p_j is not photoconsistent-visible to v_i. Negative observations can occur for several reasons including dynamic occlusion and misregistration, but more importantly, they occur when a patch does not exist at a particular point in time. These observations help us narrow down exactly when a patch does exist. Note that observations that are neither projection-visible nor photoconsistent-visible are treated as providing no information.

The CMVS+PMVS algorithm typically produces only a small subset of possible positive observations $V_{\text{PMVS}}(p)$ for each patch p, for reasons of efficiency—only a few views are necessary for computing a well-conditioned estimate of a point's location, not the hundreds or thousands of potential views available for that patch. In this work, however, I want as much information as possible (as well as both positive and negative observations) to precisely estimate t_{min} and t_{max} for each patch. Therefore, I perform a visibility graph expansion step.

To do so, for each image $v \in \mathcal{I}$, I generate a depth map by projecting all reconstructed patches into v's image frame using a z-buffering algorithm. Each patch adds a 3-pixel radius contribution to the depth map. After generating this depth map, I project each patch p once more into the image; if p projects inside v's field of view and satisfies a soft depth test (i.e. a depth test against a z-buffer that has been pushed away from the camera by a small amount) against this depth map, I deem p projection-consistent. I then compute a photoconsistency

score by comparing a reference patch for p extracted from p's reference view (computed according to PMVS' patch extraction method) to the projection of the same oriented planar patch in v. If the normalized cross-correlation of both windows is above a threshold τ_{pos}, then a positive observation is recorded. Similarly, if the normalized cross-correlation is below a threshold τ_{neg}, then a negative observation is recorded. For all experiments, I set $\tau_{\text{pos}} = 0.8$ and $\tau_{\text{neg}} = -0.2$.

2.3.3 Initial Time Interval Estimation

The set of observations for a given patch p gives us useful information about when in time p existed, via the timestamps on images that observe p. For the segmentation step of our pipeline, I require an approximate initial time span $[t_{\min}(p), t_{\max}(p)]$, which I compute from the associated image timestamps—answering the question "when did this point exist?" However, I found that it was difficult to robustly estimate a good time span from these raw observations. A natural algorithm is to simply take the minimum (earliest) and maximum (latest) positive observation timestamps as the time interval. This algorithm doesn't work well due to false positive observations and incorrect timestamps (surprisingly, even in a time when many cameras are networked, there appear to be many reasons why timestamps can be wrong). I also evaluated ordering the positive observations by time and trimming off the bottom and top k-percentile observations. But because the observations are very non-uniform samples in time (i.e., some times are much more densely sampled than others), I found that this approach also performed poorly. In general, I found that considering only positive observations was insufficient to get good time bounds for a patch. Hence, I also use the negative observations.

I found it effective to phrase this problem is in terms of classification, where I wish to select a single time interval that classifies observations as being positive (inside the interval) or negative (outside the interval), with the goal of achieving both good precision and recall. Here, "precision" is the ratio of positive observations inside the interval to the total number of observations in the interval. Similarly, "recall" is the ratio of the number of positive observations in the interval to the total number of positive observations across all of time. One common way to balance these measures is to use the F_1 score, which is the harmonic mean of these two terms with equal weight:

$$\begin{aligned} \text{Precision} &= \frac{\#\text{TP}}{\#\text{TP} + \#\text{FP}} \\ \text{Recall} &= \frac{\#\text{TP}}{\#\text{TP} + \#\text{FN}} \\ F_1 &= \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \end{aligned}$$

For each patch p, I search over all possible start and end times $t_{\min}(p)$ and $t_{\max}(p)$ (where I can limit the search to only consider the discrete set of timestamps corresponding to our observations), and select the pair defining the time interval with the maximum F_1 score. This can be computed very quickly using dynamic programming. Note that in general there is a family of intervals with the same F_1 score for a set of discrete observations, as we can vary the endpoints continuously between two observations without changing the score. Therefore, I record four scalar values per patch, the range of best start and end times for the beginning of the interval, $t_{\min}(p)$, and the start and end times for the end of the interval, $t_{\max}(p)$. Figure 2.4 shows several patches with their estimated time intervals.

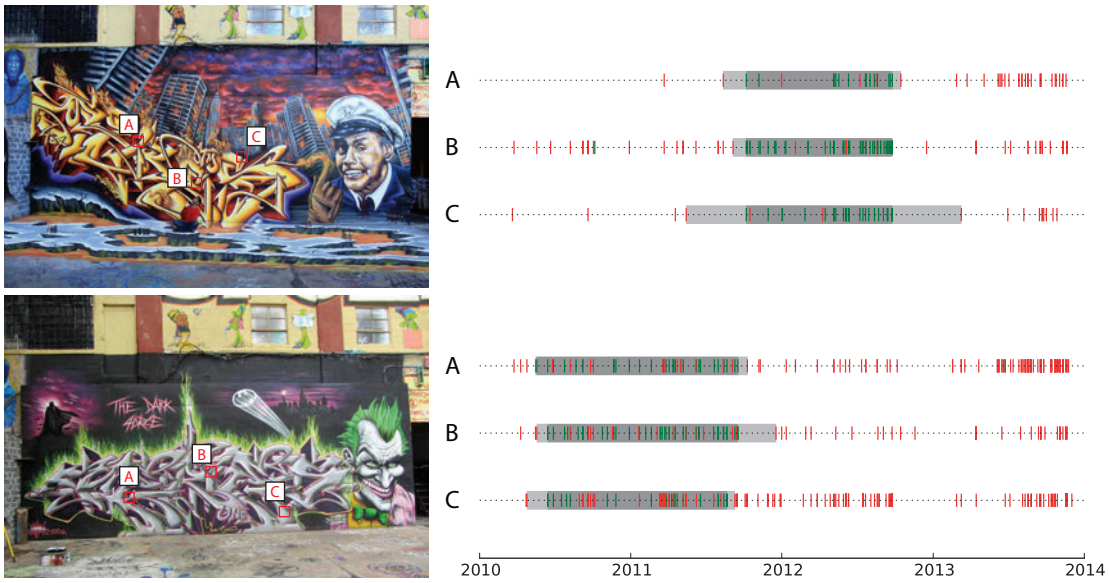


Figure 2.4: Time profiles, shown at right, for two different appearances of the same wall, shown at left. Green marks on the timeline indicate positive observations and red marks indicate negative observations. Dark gray intervals indicate the shortest interval that has the maximal F_1 score. Since I use a discrete set of observations, any interval with endpoints contained within the light gray regions will have the same F_1 score. Note the false positive red marks inside the gray intervals and the false negative green marks outside the gray intervals, as well as the very non-uniform distribution of observations. The use of the F_1 score based on classification accuracy is motivated by the need to robustly handle noisy, non-uniform observations in a more effective way than simple percentile-based methods.

2.3.4 Spatio-Temporal Graph

Now that I have a set of patches with initial time intervals, I have the information I need to segment the scene into a set of plane-time cuboids. Section 2.3.5 describes the repeated RANSAC procedure for segmenting the scene based on geometric and temporal cues. First, however, these scenes are quite large with millions of patches, and for such scenes a segmentation algorithm built on a ran-

dom sampling algorithm like RANSAC can be difficult to scale. To address this problem, I take an approach inspired by GroupSAC and related RANSAC variants [51], and preprocess our data to under-segment it into more manageable chunks where RANSAC is more likely to quickly find good hypotheses. To do so, I build a graph on the set of patches where pairs of patches are connected if they have high *spatio-temporal affinity*—if they are proximate in space and time—and then compute connected components of this graph. This method is related to the object segmentation in [60], but considers time as well, and serves as a preprocess to our final segmentation algorithm.

To build this spatio-temporal affinity graph, I connect two patches p and q if they satisfy the following criteria:

1. **Spatial proximity.** p and q are within a 3D distance δ_d, and have normals that differ by no more than an angle δ_θ.
2. **Temporal overlap.** p and q have time observations with a high degree of overlap.
3. **Common view.** p and q are seen by at least one common camera from the original PMVS reconstruction ($|V_{\text{PMVS}}(p) \cap V_{\text{PMVS}}(q)| > 0$).

For the temporal overlap criterion (item 2 above), I could use the degree of overlap in the time intervals for p and q estimated in Section 2.3.3 above. Instead, I found a somewhat stricter notion of temporal overlap to be useful, in particular, I use the positive observations for each patch to form an *observation profile*. For each positive observation for a patch p, I pad it by ±1 week to form a time interval. Overlapping time intervals for a single patch are aggregated into larger time intervals. The final set of (not necessarily contiguous) time intervals

forms the observation profile for p. To measure the temporal overlap of a pair of patches (p, q), I compute the size of the intersection divided by the size of the union of the two observation profiles (where here "size" means the total time mass in the intersection or union of two observation profiles). This gives us a single scalar temporal affinity score. If this score is greater than a threshold δ_t, then the pair (p, q) satisfies the temporal overlap criterion.

To efficiently evaluate the three criteria above, for each patch p, I use FLANN [48] to find spatial neighbors within a 3D distance δ_d, then prune these potential candidate neighbors by the surface normal, temporal overlap, and common view criteria. The surviving edges define a graph G_{st} whose connected components form a rough initial under-segmentation into objects, as illustrated in Figure 2.2(4). For all experiments, I use $\delta_\theta = 60°$, $\delta_t = 0.75$, and I manually select a per-reconstruction δ_d equal to roughly 1-2 meters. Note, an alternative to manually selecting δ_d is to geo-register the reconstruction so that it is to metric scale.

2.3.5 Plane-Time Clustering

Finally, I further segment each component of G_{st} from the initial spatio-temporal segmentation into a set of plane-time cuboids that represent coherent objects in the scene. I perform the segmentation by greedily and repeatedly running a special "Plane-Time RANSAC," on each component, after each round removing the induced subgraph given by the RANSAC inliers from G_{st} to form G'_{st}, and recursively applying the same procedure to the connected components of G'_{st}. I terminate the recursion if a connected component does not contain some mini-

```
1:  procedure PLANETIMECLUSTER(E)
2:      V ← VERTICES(E)
3:      inliers ← PLANETIMERANSAC(V)
4:      p ← ∅
5:      n ← ∅
6:      for (v1, v2) ∈ E do
7:          if v1 ∈ inliers & v2 ∈ inliers then
8:              p ← p ∪ {(v1, v2)}
9:          else if v1 ∉ inliers & v2 ∉ inliers then
10:             n ← n ∪ {(v1, v2)}
11:         end if
12:     end for
13:     C ← CONNECTEDCOMPONENTS(p)
14:     for cc ∈ CONNECTEDCOMPONENTS(n) do
15:         C ← C ∪ PLANETIMECLUSTER(cc)
16:     end for
17:     return C
18: end procedure
```

Algorithm 1: **Plane-Time Clustering.** The set of points in the spatio-temporal graph are segmented by finding a set of inlier points, removing their induced subgraph, and recursively segmenting the connected components of the remaining subgraph.

mum number of vertices or if the algorithm is unable to find any valid RANSAC hypotheses after some fixed number of iterations. This process is run in parallel on each connected component. Algorithm 1 summarizes this process.

Plane-Time RANSAC combines a standard plane-fitting inlier criterion with a new time interval hypothesis criterion. In particular, the algorithm samples three patches at random from the component and fits a plane to the three samples. It also computes a time interval hypothesis by taking the union of the patches' time intervals as estimated in Section 2.3.3. Thus a hypothesis is a combination of a plane and a time interval. A patch is considered an inlier to this hypothesis if (1) it is within some spatial distance of the hypothesis plane, (2) its

procedure PLANETIMERANSAC(V, iters, δ_d, δ_n, δ_t)
    best $\leftarrow \emptyset$
    $t' \leftarrow (-\infty, \infty)$
    for $i \leftarrow 1$, iters **do**
        $v_1, v_2, v_3 \leftarrow$ SAMPLE(V)
        $p \leftarrow$ PLANEFIT($\{v_1, v_2, v_3\}$)
        $t \leftarrow (\min(v_{1t} \cup v_{2t} \cup v_{3t}), \max(v_{1t} \cup v_{2t} \cup v_{3t}))$
        inliers $\leftarrow \emptyset$
        for $v \in V$ **do**
            if $\|p_n^T v_p - p_d\| < \delta_p$ & $p_n^T v_n > \delta_n$ & $\frac{\|t \cap v_t\|}{\|t \cup v_t\|} > \delta_t$ **then**
                inliers $\leftarrow$ inliers $\cup \{v\}$
            end if
        end for
        if ||inliers|| > ||best|| **then**
            best $\leftarrow$ inliers
            $t' \leftarrow t$
        end if
    end for
    $p \leftarrow$ PLANEFIT(best)
    inliers $\leftarrow \emptyset$
    for $v \in V$ **do**
        if $\|p_n^T v_p - p_d\| < \delta_p$ & $p_n^T v_n > \delta_n$ & $\frac{\|t' \cap v_t\|}{\|t' \cup v_t\|} > \delta_t$ **then**
            inliers $\leftarrow$ inliers $\cup \{v\}$
        end if
    end for
    return inliers
end procedure

Algorithm 2: **Plane-Time RANSAC.** Each iteration of RANSAC samples three vertices at random, computes their best-fit plane, and takes the union of their time intervals. A vertex is considered an inlier if it satisfies a point-to-plane distance constraint, a normal-to-plane constraint, and a time interval overlap constraint.

normal is within some cosine distance of the hypothesis plane, and (3) its time interval has a high degree of overlap with the hypothesis time interval. Once again, I define overlap of two time intervals to be the size of their intersection divided by the size of their union. Because the estimated patch time interval is really a family of intervals with endpoints that are allowed to vary slightly, I select the unique patch time interval that maximizes the overlap between the patch and the hypothesis. Note that when generating a hypothesis, three non-collinear points uniquely define a plane, but the normals and time intervals of the sample points themselves may not satisfy the inlier criterion for the hypothesis. I check for such cases and discard the hypothesis if any sample is not an inlier to its own hypothesis. In our experiments, RANSAC was allotted 1000 attempts to generate a hypothesis, but was limited to 100 valid hypotheses. This process is summarized in Algorithm 2.

At termination, I have a set of plane-time cuboids (associated with each cluster), and each patch has either been assigned to a cluster or has been discarded (as not belonging to a large enough cluster). The time interval associated with each plane-time cuboid can then be transferred back to the individual patches in the cluster to obtain improved per-patch time intervals, since I assume the entire object represented by the cluster appears and disappears as one unit. The distance thresholds, normal thresholds, and time overlap thresholds used in all RANSAC runs are identical to their analogous counterparts in Section 2.3.4.

In order to visualize these plane-time cuboids I wish to use the input photos to synthesize textures. Since camera technology has undergone a great deal of change in terms of resolution over the past decade and I aim to make use of all available data, rather than select a single world-dependent resolution for a

Dataset	Photos Retrieved	Photos Registered	Patches	Positive Obs.	Negative Obs.
Times Square	1,222,792	246,218	13,590,341	5,266,855,599	12,347,622,570
5 Pointz	48,375	12,777	6,306,013	820,203,882	634,649,156
Akihabara	171,469	13,946	1,732,784	132,520,779	254,624,878

Table 2.1: Dataset statistics.

particular proxy, I sample each texture according to the resolution of the image. The resolution at which I generate a texture is the same as used in PMVS. This produces a texture stack for each proxy where each texture has the same aspect ratio, but different resolutions. Associated with each texture is an alpha matte as each camera may not have observed the entire extent of the proxy.

In order to merge this texture stack into a single representation suitable for interactive rendering, I compute a frequency-space decomposition for each texture by building a Difference of Gaussians pyramid. Like-resolution levels of these pyramids are averaged and then collapsed to produce the final texture map. This has the benefit that areas observed with high resolution may maintain their fine detail while allowing areas observed with low resolution to have some contribution.

2.4 Experiments and Results

2.4.1 Datasets

I ran the pipeline on three highly dynamic datasets of urban environments from photos taken over a span of several years. `TimesSquare` in Manhattan, NYC contains a quarter million registered photos exhibiting a wide variety of change

Figure 2.5: A texture mapped rendering of `5Pointz` at two different points in time.

in appearance in terms of billboards, signs, and building facades. `Akihabara` in Tokyo offers a related setting where most of the dynamic content is localized to billboards and signs. However, `Akihabara` also contains some dramatic instances where entire buildings change appearance. Finally, `5Pointz` in Queens, NYC, served for several years as an outdoor art exhibit space for graffiti artists, and offers a visually rich experience involving many pieces of art over time. In this last case, the algorithm can be seen as a step towards automatic, distributed, art documentation. Table 2.1 shows the scale of these datasets.

2.4.2 Viewing and Rendering

My plane-time cuboid representation provides a simple yet effective way to explore the evolution of a scene through space and time. I have created an interface where a user can explore the scene using standard CAD controls such as drag to translate, scale, and rotate, but can also scrub back and forth in time using a slider control. At any given time, I can render a snapshot of the scene as a point cloud using the patches that existed during that time, or can additionally

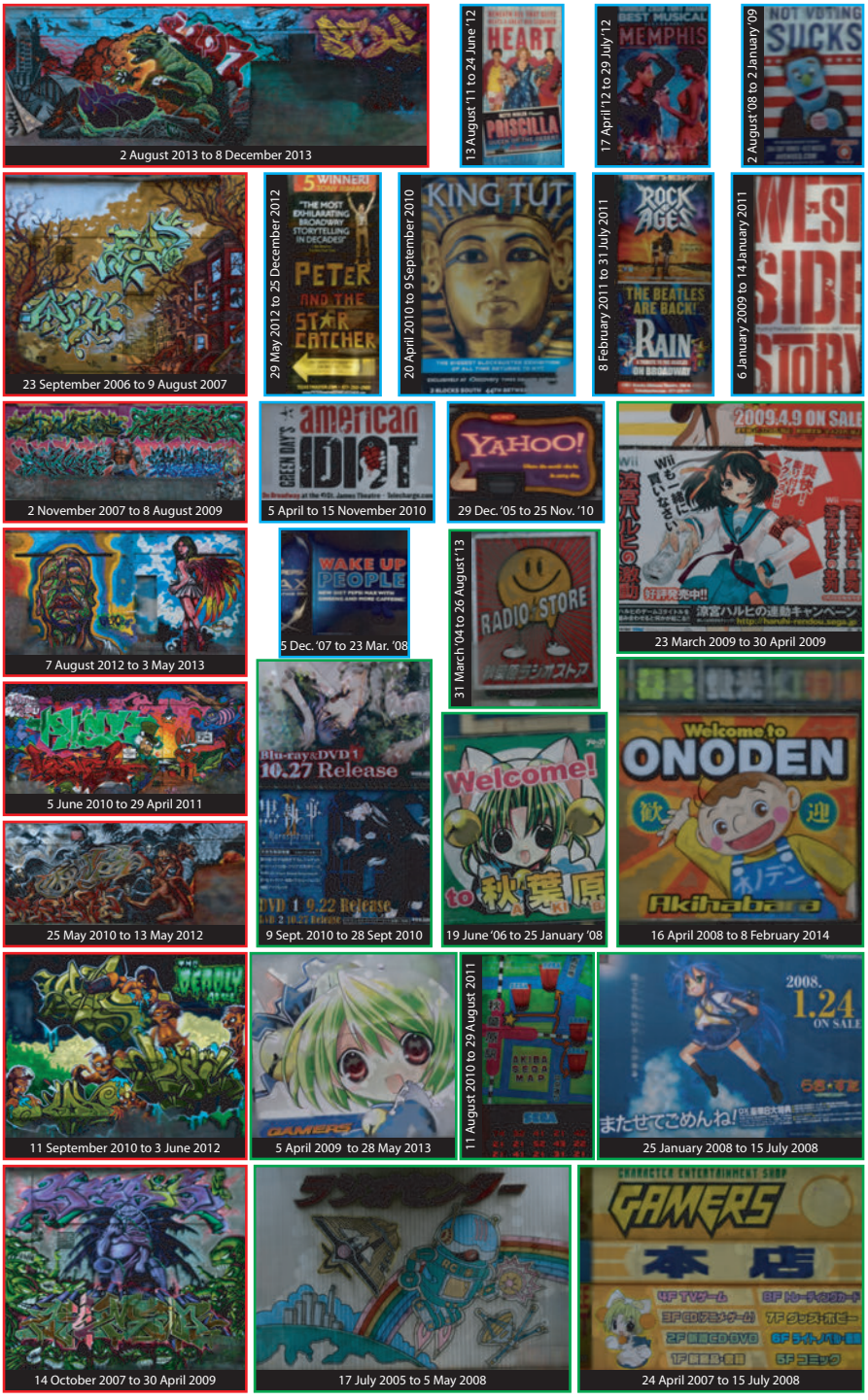
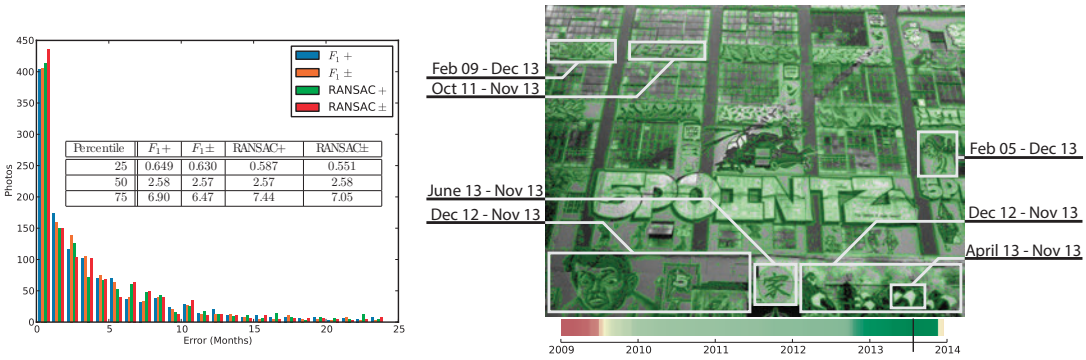


Figure 2.6: An assortment of plane-time cuboids from `TimesSquare` (blue), `Akihabara` (green), and `5Pointz` (red) with texture maps and estimated time intervals produced by the system.

render each cuboid as a texture mapped "quad." To do so, for each plane-time cuboid proxy, I generate a bounding 3D rectangle with one axis perpendicular to the scene's estimated up vector. For each camera that observes at least six patches associated with the cuboid, I apply the homography mapping image space to the quad's texture space. To enable clean blending between multiple cameras despite transient occlusions and geometric misregistration, I generate an alpha matte using Gaussian splats of uniform size centered at the projections of the positive patch observations. Examples of these quads are shown in Figures 2.2, 2.4.2, and 2.6.

These plane-time cuboids qualitatively represent meaningful semantic units. Figure 2.4.2 shows a birds-eye view of `5Pointz` at two distant and visually distinct points in time. Throughout time, segments come in and out of existence, sometimes in synchrony, sometimes independently. It is not until the whole collection is analyzed that the behavior of the scene becomes clear. My claim is that for these sorts of scenes, it would be exceedingly difficult for an expert to manually analyze each photo to determine the temporal structure of the scene over the span of several years. However, I can achieve these results fully automatically. Figure 2.6 shows a number of interesting examples of the automatically recovered space-time cuboids. Of particular interest are examples with temporal cues contained within the image. For example, a poster advertising the 2008 US presidential election is estimated to exist during the election period. Anime ads found in Akihabara advertise products with recent release dates. Graffiti in 5Pointz can be matched with coarse-grained databases found online. Each of these examples gives us clues about the appearance of the world at a given point in time. While it is difficult to evaluate results quantitatively, subjectively I achieve segmentations with close correspondence to temporal objects in each



Percentile	F_1+	$F_1\pm$	RANSAC+	RANSAC±
25	0.649	0.630	0.587	0.551
50	2.58	2.57	2.57	2.58
75	6.90	6.47	7.44	7.05

Figure 2.7: `5Pointz` timestamp prediction results. Positive and negative observations provide clues as to when a photo was taken. Positive observations for a sample photo are shown on the right in green. Several components of this scene are well-localized in time and can be used to build a timestamp scoring function. The timestamp with the highest score is marked on the timeline. I predict this photo was taken on July 25, 2013 and the photo metadata indicates it was taken on August 10, 2013.

scene.

2.4.3 Timestamp Prediction

One application of this system is for the task of predicting a timestamp date for a photo, in case a date is missing or incorrect. This time-stamping algorithm analyzes what portions of our time-varying reconstruction are observed in a query photo, what portions are not observed, and determines a date that best fits that information.

Given a new query photo registered to the model (e.g., using 2D-to-3D matching), I use the same observation generation procedure as described in Section 2.3. However, I select a much more conservative $\tau_{\text{neg}} = -0.5$ to obtain more

confident observations. Given the photo and its positive and negative observations, I use a voting scheme to determine a region of time where the positive time intervals overlap, but where the negative observation time intervals do not. For each observation, I generate two events, a start event and an end event; each event carries a vote. For a positive observation, the start event carries a vote of $+w$ and the end event carries a weight of $-w$ where w is some tunable weight, $w = 10$ in our experiments. For a negative observation, the start event carries a vote of -1 and the end event carries a weight of $+1$. The score for each time subinterval is computed by sorting all events by timestamp and then stepping along the timeline, accumulating the vote carried by each event. Then the subinterval with the maximal score is found. To generate a single predicted timestamp, I select the midpoint of this interval.

In practice, I found that I had to consider the effect of unmodeled occlusions and its impact on negative observations. By unmodeled occlusion, I mean the effect of holes in the depth buffer used to determine projection-visibility of a point due to parts of the scene geometry not being reconstructed. While this is not as much of a concern when aggregating observations at each patch since this effect is more or less random for a given patch, it is very consistent for aggregating observations for each camera. Therefore, I only include a negative observation if at least one positive observation is in close 3D spatial proximity. This implies that I could see the surface geometry at one point in time, but I do not currently see a certain appearance in the query photo.

To evaluate this method, I conducted a timestamping experiment on `5Pointz`. 10% of the registered photos were selected as a test set and my pipeline was carried out as a training stage on the remaining 90%. The test

set was selected such that photographers do not overlap between training and testing sets to prevent any bias a single photographer might introduce. Note that while I measure distance to the original timestamps as ground truth, these contain noise and so the results should be interpreted with this in mind.

I present four variations on the algorithm described above. Methods denoted F_1 use the per-patch time estimates obtained by our F_1 interval procedure before Plane-Time RANSAC segmentation is performed. Methods denoted RANSAC use per-cluster time estimates obtained via Plane-Time RANSAC. Methods denoted + use only positive observations while those denoted ± use both positive and negative observations. Figure 2.7 shows the results as a histogram of distances between predicted and original timestamps. I observe improvement with both the F_1 and RANSAC methods after I introduce negative observations, achieving a median distance of around 2.5 months, with 25% of images having about 0.5 month distance (quite low given that all the only cue we are using is changing scene appearance). However, there is an increase in error for the 75^{th} percentile when we move to the RANSAC methods from the F_1 methods. One explanation is that by clustering the points, I might be losing some precision in our time estimates; another is that I am seeing effects from noise in the original timestamps.

2.4.4 Bad Timestamp Detection

Since the timestamp prediction algorithm generates a score for each point in time, one additional application is to evaluate that score at the time recorded for the photo. If that score is low enough, then that gives us a hint as to whether

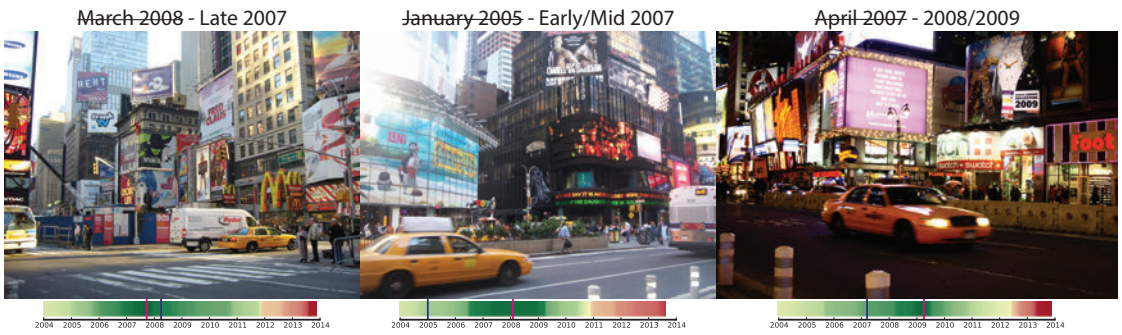


Figure 2.8: Retimestamped photos. The timeline indicates the timestamp scoring function with the given timestamp indicated in blue and the predicted timestamp indicated in pink. The originally recorded date and estimated date range as determined by manual inspection are provided above each image.

or not a photo might not agree with this model of the world. This could be due to a number of reasons, but in practice I found it was most often due to misregistration and bad timestamps. In the case of misregistration, the positive observations often correspond to the content in the photo that caused the misregistration; e.g. a logo found on a storefront might also appear on a product, in which case the timestamp of the storefront photo is irrelevant when evaluating the timestamp of the product photo. Another scenario is when the original timestamp of the photo itself is incorrect.

To evaluate the effectiveness of this score for identifying incorrectly timestamped photos, I randomly selected a subset of photos from the Times Square reconstruction not seen by my algorithm and labeled them indicating whether I believed the timestamp to be correct to within a couple of months, believed the timestamp to not be correct to within a couple of months, or could not confidently identify any discriminative content in the photo to pinpoint a date. In practice, this meant identifying movie, television, and sporting event billboards.

407 photos were labeled as unknown, 61 were labeled as correct, and 9 where labeled as incorrect. I applied the timestamping method to known photos from `TimesSquare` that I believed to have incorrect timestamps based on the content in the scene (i.e., dated ads, weather, clothing). Figure 2.8 shows that the algorithm was able to predict timestamps that more closely matched my belief of when these photos were taken.

2.5 Limitations

There are several limitations associated with this segment representation. The most important limitation is that segments with a single start and single end point cannot accurately model periodic phenomenon such as surfaces that are illuminated by the sun during the day and by artificial lighting at night. Some sort of periodic day/night representation is necessary to capture these effects. In some existing situations, both the day and night appearances are merged leading to poorly merged texture maps for the visualized quads. Electronic displays require a representation completely different from that presented in this chapter since they can change very rapidly and are unlikely to be reconstructed by the 3D reconstruction phase. Another limitation is that our segmentation algorithm does not enforce non-intersection constraints between the space-time cuboids nor does it require that the spatio-temporal domain be entirely filled with cuboids. Finally, there's a limitation in that since this pipeline uses 3D multi-view stereo results as input to a process that extracts the additional temporal information. Any mistakes made in the 3D stage are propagated to the temporal stage. A reasonable alternative would be to build a holistic method that reasons about space and time jointly during the dense patch-based recon-

struction.

2.6 Conclusions

This chapter presented a method by which a physical space with changing appearance in the real world could automatically be reconstructed using photos retrieved from the internet with a keyword search. The bigger picture behind this piece of work is that it offers a way to extract elements in these scenes that have some larger story to tell. For example, given two of these segments located on opposite ends of the Earth, can we measure how similar they are? Can we figure out which places on Earth are connected in terms of style? How can we make such comparisons across disparate cities and across time? To do this reasoning, we need a few more tools to help us identify visual similarities and differences between categories of interest. Chapter 3 will present a method that can take labeled photo collections and extract a representation useful for visualizing patches of localized content that are salient for image classification tasks.

CHAPTER 3

DISCOVERING VISUAL CORRELATIONS IN LABELED PHOTO COLLECTIONS

If we wish to study space and time in online photography, we need some idea of what is relevant to compare and contrast between some subsets of photos. In Chapter 2, I presented a method that assumed that the atomic units, or *visual vocabulary* consisted of 3D planar elements that existed for some finite, contiguous period of time. This is strongly tied to a geometric notion of the world. But what if we have some arbitrary imagery that we wish to analyze? Perhaps these images come with metadata such as geotags or timestamps. Can we make a visual comparison between architecture in two cities? Can we find differences in the types of clothing people wear in the summer versus winter? These types of comparisons require a much more general type of visual vocabulary, one that should be learned for arbitrary classification tasks with few assumptions about the types of classifications. (i.e., The method should be applicable whether we are classifying species of birds or types of scenes.) This chapter will present a method for building a visual vocabulary for arbitrary discriminative tasks and it has been presented at the International Conference on Computer Vision 2015 [44].

Fortunately, the rapid growth of online imagery and the emergence of powerful machine learning techniques have led to exciting developments in computer vision. While many of these developments have been in visual recognition, these same trends also enable new approaches to *visual discovery*. By visual discovery, I mean a form of visual data mining—identifying patterns or correlations in visual data that can be used to propose or validate hypotheses about

the world.

Towards this goal, a few recent papers have explored visual discovery in particular domains. I am especially inspired by the work of Doersch et al. [15] where visual elements (i.e., patches) that give cities such as Paris their distinctive look are automatically mined from a corpus of Street View photos. This task is an application of a broader set of methods for mining imagery for distinctive visual elements [65, 13, 33, 2]. This work also seeks to discover visual elements in the form of patches. However, while most prior work uses handcrafted features such as HoG, I seek to harness the power of convolutional neural networks (CNNs) for visual discovery. I believe the time is ripe for studying this problem—just as text analysis can reveal trends on social networking sites like Twitter, visual discovery driven by powerful image understanding methods could enable the automatic identification of visual trends in fashion, design, art, and many other areas. These *discriminative patches* are a promising representation for such visual vocabularies.

CNNs have shown astonishingly good performance for visual recognition [37, 26]. However, in order to be useful for visual discovery, we need to be able to extract insight about what a CNN has learned, and communicate these insights to humans. Extracting such insights at scale by analyzing learned networks has proved challenging, because CNNs tend to be very opaque. For instance, a given learned unit (neuron) deep in a network can depend on almost all of the input pixels in a highly non-linear way, making it very difficult to associate features in the network with visual elements that are meaningful to humans.

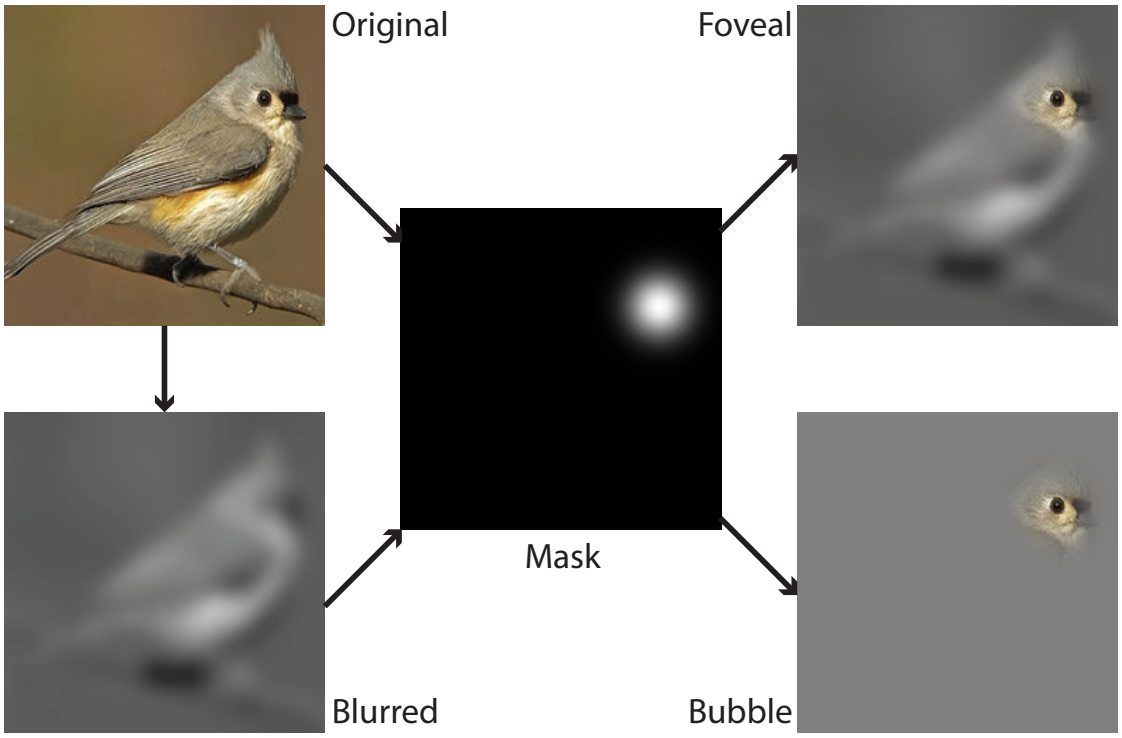


Figure 3.1: **Foveated imaging.** This method analyzes the contribution of localized content in images by modifying the data shown to a CNN during training and at test time. Given an input image (top left), I apply an alpha mask (center) to produce two representations: a *foveal image* (top right) and a *bubble image* (bottom right). While a *bubble image* preserves a small amount of spatialized content, a *foveal image* includes a blurred, grayscale background to provide additional context.

In contrast, rather than trying to interpret a network after the fact, my key contribution is a new type of training process that forces the network to learn much more localized, interpretable properties of the input imagery. Namely, I intentionally restrict how much of the image the network gets to see at training and test time. Inspired by the BubbleBank work of Deng et al. [11], the network is only allowed to look at a small portion of the image in high resolution, via a *foveal image*, where the rest of the image is blurred out, or a *bubble image*, where

the rest of the image is blank, as illustrated in Figure 3.1. However, the network can optimize the spatial position of the bubble in order to maximize classification confidence. In this manner, my approach efficiently learns both network weights as well as the most salient parts of each training image. I then cluster these salient image regions to derive a set of visual elements for a given dataset.

This new approach to finding discriminative visual elements has several key attributes: it is a simple modification of existing CNNs methods, and it scales to large amounts of data, which is critical for many applications. I show a wide range of results on existing datasets, as well as two new large-scale datasets of people and places. I also perform quantitative comparisons of the descriptive power of our visual elements at classification tasks, showing they outperform state-of-the-art algorithms. Finally, I show applications of our method to visual discovery in analyzing people over space and time.

3.1 Background and Related Work

This work is in the vein of discriminative patch mining methods [65, 13, 33, 39]. Such methods typically take in a corpus of images and try to identify clusters of "mid-level patches" where each cluster represents an element that appears often in a single category, and rarely in other categories. Each cluster represents an identifiable "visual theme" that is **discriminative** for a class. The scope of a visual theme is left unspecified and is something to be discovered by the algorithm, but examples of visual themes one might expect to emerge include:

- Specific instances of objects for object or scene classification

- Architectural elements for street-level city classification [15]
- Coloring and other markings for animal species classification

What makes these patch clusters useful is that they are visually homogeneous to the extent that a human analyst can inspect them to understand what has been encoded by the classifier, but they are simultaneously heterogeneous to the extent that a few clusters can succinctly encode the variety of concepts necessary to perform the original classification task without the original image content, that is, act as proxies to the original photo.

Low-level vs. deep-learned features. Prior discriminative patch methods often build mid-level representations on top of low-level features (e.g., HoG) [65, 13, 33]. However, recent methods that learn features straight from pixels (e.g. CNNs) have enjoyed great success on recognition problems [37, 26]. Can we also use CNNs to learn distinctive visual elements? Raw CNN features (without fine-tuning) have been shown to have useful properties for this task [39]. On the one hand, the representative power of CNNs suggest that they can learn visual elements that are more subtle and with greater invariance to image appearance compared to low-level features. But on the other hand, standard CNN architectures, such as AlexNet [37], are opaque—their features and inner workings are such that can be very challenging for humans to understand and interpret.

Understanding deep networks. Most prior network visualization methods have sought ways to analyze and visualize existing networks. Given enough visual inputs, one can compute and examine distributions over neural activations (i.e., the scalar outputs of individual units of the network). For example, a particular neuron may learn a human face. Another may respond to text. By

measuring which neurons are highly correlated with certain types of inputs, we can gain a low-level understanding of the organization taking place [26].

However, other recent results show that reasoning about the semantics of individual high-level units can be misleading, because linear combinations of units are just as meaningful as individual units [73]. Further, evidence suggests that neurons undergo co-adaptation in order to robustly and compactly encode concepts [89]. Indeed, my own experience has shown that it is challenging to reliably identify human-recognizable concepts from the activations of individual neurons. Units at the end of a large network often have very large receptive fields—i.e., regions of input pixels that pool together through the network to effect that unit—perhaps even the entire input image. Many useful visual elements are highly localized (e.g., the beak shape of a particular bird species), and it can be challenging to discern how these local features are represented in the network.

An alternative to direct analysis of network activations is to present controlled image stimuli to the network, and measure the difference in classification performance. For instance, one can slide an occluder (e.g., a gray box) over the input image and see where the performance decreases the most. These regions are identified as salient for a given category [90, 95]. However, these subtractive methods present CNNs with a new kind of image that they have never seen, and there are no guarantees on how the network will perform. I find that it is much better to fine-tune networks using these kinds of modified images as training data. Further, for many types of scenes (e.g., a bookstore full of bookshelves), any occluder location may yield similar classification performance; an additive method like the one presented is more suitable for identifying the most

salient regions.

This work is also related to the scenes and objects work of Zhou et al. [95], which uses a subtractive method to analyze saliency of scenes, and shows that objects emerge as key element of the learned representation. My goal is a general technique that we apply to a range of datasets.

Unlike these prior methods, I change the way the network itself is trained to make the results more interpretable. This method is in some sense the inverse of the occlusion methods—I obscure or hide all of the image except for a limited region (a "bubble"), and train the network to recognize these obscured images resulting in a new network that we call a *BubbLeNet*. This idea is inspired by work in psychology on using bubbles for human studies, e.g., determining which parts of a human face are most important for gender and expression recognition in people [28]. This idea has been used in the vision community to use humans to identify discriminative elements in fine-grained categories [11]. In our case, I flip this around—rather than learn what humans see, we use bubbles to encourage the network to automatically learn localized, human-interpretable visual elements.

Visual attention. This work is also related to work on saliency and visual attention. Recent work has used deep learning to attend to salient regions of an image [76, 47], and applied this to applications such as object recognition [3] and automatic caption generation [84]. This bubble model also uses a form of visual attention to capture more local properties of scenes, but in a much simpler way that can be implemented by modifying the training data.

3.2 Method

This method has two main parts: (1) **patch discovery**, where I train a CNN that can localize discriminative visual content and (2) **patch clustering**, where I aggregate the discovered visual elements in a way suited for visualization and for detecting visual elements in new images.

3.2.1 Patch Discovery

The main contribution of this method is a simple modification to the images shown to the classifier during both training and test time. This modification intentionally restricts the amount of information presented to the classifier in a spatially-localized way. In particular, the classifier is shown a blank background with a single "bubble" of exposed image, as shown in Figure 3.1 (bottom right). By making this modification, I can analyze the relative performance of the classifier as the focal location of this bubble is moved. Effectively, this modification allows me to generate a saliency map, by forcing the network to fixate on regions of the image at a time. Once I have a saliency map, discovering discriminative elements is much easier since I can use that map as a distribution for sampling candidate visual elements.

The primary motivation for this method comes from the line of work in psychology [28] and vision [11] where small, circular regions of images are presented to humans in order to study the importance of different visual features to human perception. These human-derived insights could then be transferred to an algorithm, but the problem at hand is exactly the opposite. In this case,

state-of-the-art classifiers built from convolutional neural networks perform so astonishingly well that they outperform well-trained humans for certain difficult tasks [58]. What makes matters worse is that despite direct access to the inner workings of these models, any statistics potentially encoded in the learned parameters are extremely opaque. I would like the model to teach us what it has learned and I do so by conducting experiments based on this established human-centric protocol.

Figure 3.1 illustrates this data modification. Given an input image (which I call a "normal" image), I generate an alpha mask with a single Gaussian blob (centered, for now, on a random location). Similar to [28], I use this mask to create a *bubble image* containing only a single colored patch. By modifying a normal image to create several bubble images with different focal locations, I can examine the CNN's prediction scores to determine which parts of the image provide the most evidence for the true class label.

I found that in practice if you take a CNN that has been trained on normal images for a particular task and then test it with bubble images, it will perform no better than random guessing. I found that the network can work much better if it is fine-tuned on **both** normal and bubble images.

An alternative to a bubble image is a *foveal image* which sits somewhere in-between a normal image and a bubble image. A foveal image is created by first generating a Gaussian-blurred, gray-scale version of the normal image. This blurred image is then composited with the normal image using the same alpha mask used to generate the bubble image (Figure 3.1, top right). The result is an image which is very similar to the images used in the user study of [11]. Foveal images preserve some contextual information beyond the bubble which

potentially makes training easier. However, when I do patch discovery using foveal images directly, the results tend to be misleading since the silhouettes of objects can still be discerned even if the focal point is placed far away. After this *FoveaNet* has converged, these foveal images can be replaced with bubble images and fine-tuned once again to produce a *BubbLeNet*.

To recap, starting from a CNN trained on ILSVRC12 [58], I use a three stage fine-tuning regimen to train a network better suited for spatially localized predictions:

AlexNet/GoogLeNet: Fine-tune the network on 100% normal images.

FoveaNet: Fine-tune the AlexNet/GoogLeNet on 50% normal images and 50% foveal images.

BubbLeNet: Fine-tune the FoveaNet on 50% normal images and 50% bubble images.

I found that in practice, it is sufficient to sample focal locations uniformly at random when generating the foveal and bubble images for training. Hence, generating this extra type of training data is very fast.

Now that I have a CNN trained on bubble images, I can use it to discover salient patches. A naive approach to exploring the saliency of a normal image is to generate a large number of bubble images and pass each through the CNN, recording the score for the true class label. However, this is prohibitively expensive. Fortunately, the choice of alpha mask makes local optimization of focal location quite easy. I can easily compute the gradient of the bubble image with respect to the Gaussian blob's center, and use stochastic gradient descent with the back-propagation algorithm to compute the gradient with respect to the bubble location, moving the bubble along that gradient until we reach a lo-

cal optimum. In the setting where I wish to mine for discriminative patches, I know the class label for the image, so I can minimize cross entropy between the true label and the prediction distribution. In the setting where I wish to predict the class label for a test image given only one bubble to view at a time, I minimize the entropy of the prediction distribution. I present a more rigorous justification of our algorithm in Section 3.4.

This optimization has many local minima, so I sample 128 initial bubble locations per image at random and run SGD for 10 iterations to optimize the position of each bubble according to the gradient of the loss function. The bubble with the minimum loss at the end is selected as the most discriminative bubble for that image.

3.2.2 Patch Clustering

Having discovered discriminative patches in each photo, I now cluster them to produce better visualizations and to build detectors for finding these patches in new photos.

I apply our BubbLeNet to a validation set of photos and find bubble locations that maximize the correct class label. I then take the top scoring bubble per photo. Each photo in the validation set now has a *normal* image and a *bubble* image as well as corresponding feature vectors extracted from the penultimate layer of our CNN (e.g., fc7 for AlexNet) for both normal and bubble images. I split this set of CNN features into two groups, denoted S_{normal} and T_{normal} with associated bubble image feature sets S_{bubble} and T_{bubble}. The algorithm then (1) clusters the bubble images into groups of similar bubbles, and (2) builds a clas-

sifier for each such cluster that predicts whether a normal image contains that visual element.

This algorithm is based on [65]. I begin by running k-means clustering on the elements of S_{bubble} and use the cluster assignments as labels to train a linear multiclass SVM on S_{bubble}. The resulting SVM is applied to the holdout set T_{bubble}. The class prediction for each element in T_{bubble} is used as its new label and I train a linear multiclass SVM on T_{bubble}. This second SVM is applied to the new holdout set S_{bubble}. These predictions are propagated from the elements of S_{bubble} to their corresponding elements of S_{normal} and a third linear multiclass SVM is trained. This produces an SVM that can take a normal image and predict whether a patch class is contained within.

Unless I have prior knowledge of the number of clusters for a given dataset, I opt to over-provision the clusters, then rank them to determine which ones to keep. Like [65], I build a ranking function based on two measures of cluster quality. The first is entropy: I want a cluster to cover a small number of classes, i.e., have low entropy. The second is confidence: I want the scores of the top activations in the cluster to be high. Therefore, for each cluster I compute entropy minus λ times the sum of the top N SVM scores. I used $N = 5, \lambda = 0.01$ in the experiments. I apply this function to each cluster and rank them accordingly.

The results of this algorithm are a ranked set of bubble image clusters and a detector for each cluster. Because the bubbles in each image have a small footprint, I can also extract a patch from each bubble image as a square region containing the bubble. These can be used for visualization. Figure 3.5 shows examples of some of these clusters for a variety of datasets.

Name	Source	Examples	Classes
CUB-200-2011	[81]	11,788	200
Food-101	[6]	101,000	101
ILSVRC2012	[58]	1,381,167	1,000
MIT Indoor67	[55]	15,620	67
MIT Places205	[94]	2,469,371	205
StreetViewCities	Ours	15,392,883	44
InstagramPeople	Ours	16,259,028	Varies by task

Table 3.1: **Dataset synopsis.** I validate our method on a variety of datasets. Note that InstagramPeople does not come with explicit categories for classification, but instead an expressive set of metadata from which both classification and regression tasks can be constructed (e.g., by geolocation or timestamp).

Discussion. This patch discovery and clustering algorithm has several key features worth noting. It learns a network that can *jointly* classify both normal and bubble images. This property is very useful for building classifiers that predict the presence of a visual element from a normal image. In addition, because the network can model spatial location, the patch clusters I discover not only can encode appearance information, but also spatial distributions (e.g., if a feature usually appears towards the bottom of an image).

3.3 Datasets

I evaluate on a range of existing datasets, plus two large, new datasets: StreetViewCities and InstagramPeople. A summary of these datasets is presented in Table 3.1.

Existing datasets. This method is designed to highlight discriminatively informative content between classes in a dataset, so large-scale, fine-grained datasets

Figure 3.2: **StreetViewCities dataset.** Several tiles shown from various cities. The number of images per city varies from 102,192 (Venice) to 964,703 (Tokyo). Each 360° spherical panorama has been cropped to produce 7 tiles. Can you guess each city?[2]

are ideal. **CUB-200-2011** [81] is a standard bird dataset for fine-grained classification. I crop the images with a padded bounding box and we do not use the part annotations. **Food-101** [6] is a fine-grained food classification set. **Indoor67** [55] and **Places205** [94] are datasets of scene images. Places205 is much larger, while Indoor67 has been used in much prior work. Finally, I apply our method to **ILSVRC12** [58].

StreetViewCities. To mine visual elements from cities, in the spirit of [15] (but on a much larger scale), I built a very large-scale database of Street View images of 44 cities. Google Street View photos were acquired by randomly sampling pre-defined regions of the world. First, I randomly sampled a city from a short list of populous cities, and then sampled a geodetic coordinate uniformly at random from that city's bounding box. If Street View found a panorama within 50 meters of our query, I retained the preprocessed tiles. Figure 3.2 highlights tiles from several cities. Each tile was then resized to 256x256. I performed addi-

Figure 3.3: **InstagramPeople dataset.** Several examples of cropped people from the InstagramPeople dataset. Networks were trained on crops (shown resized) to network input dimensions.

tional filtering for the experiments in this paper. The dataset was subsampled such that no two panoramas are within 30 meters of each other to produce 15.4 million photos. Next, the dataset was further subsampled to provide balanced classes. Finally, a 99-1 training-test split was made with the constraint that all tiles from a single panorama were placed entirely in the same part of the split. This final training set contains about 4.5 million photos.

InstagramPeople. I am also interested in exploring elements, such as fashion, that distinguish *people* in different cities, or at different times, from one another. These experiments act as a precursor to those expounded upon in Chapter 4. To that end, I built a dataset of Instagram photos taken in cities around the world, along with people detections in each photo. To build this dataset, I created a list of populous cities and for each city queried the Instagram API for public photos taken within 5km of the city center. I removed exact duplicates, and then each photo was passed through a deformable part model detector [25] trained on images of people.

To provide precise face localization and to filter out photos where a person is

[2]Top row: Vancouver, Tel-Aviv, Hong Kong, Amsterdam, Singapore. Bottom row: San Francisco, Tokyo, Taipei, Barcelona, Seoul.

facing away from the camera, I ran all of the people detections through the face-plusplus.com face detection API. Each image additionally contains time, place, and weather annotations. Figure 3.3 shows several example detections used in this dataset. Each detection has been pose normalized such that that face appears in the same location and at the same scale.

InstagramPeople can be used in many different ways. I show results in Table 3.2 for classifying people in NYC during summer vs. winter, labeled `InstagramSeasons`. The full size of the dataset is 16.2 million people and each image contains on average 2.36 people.

3.4 Experiments

I now evaluate the mining algorithm from Section 3.2. In particular, I validate that the three stage fine-tuning protocol makes training a network to recognize entire images from isolated bubbles feasible, that I can mine for the best regions in a dataset automatically, and that those regions are in fact discriminative both qualitatively and quantitatively.

3.4.1 Raw Classification Performance

Since StreetViewCities is a new dataset, I briefly present a confusion matrix generated by classifying the test set in Figure 3.4. Accuracy numbers for the other datasets can be found in Table 3.2.

I first evaluate the performance of each stage of the classifier on differ-

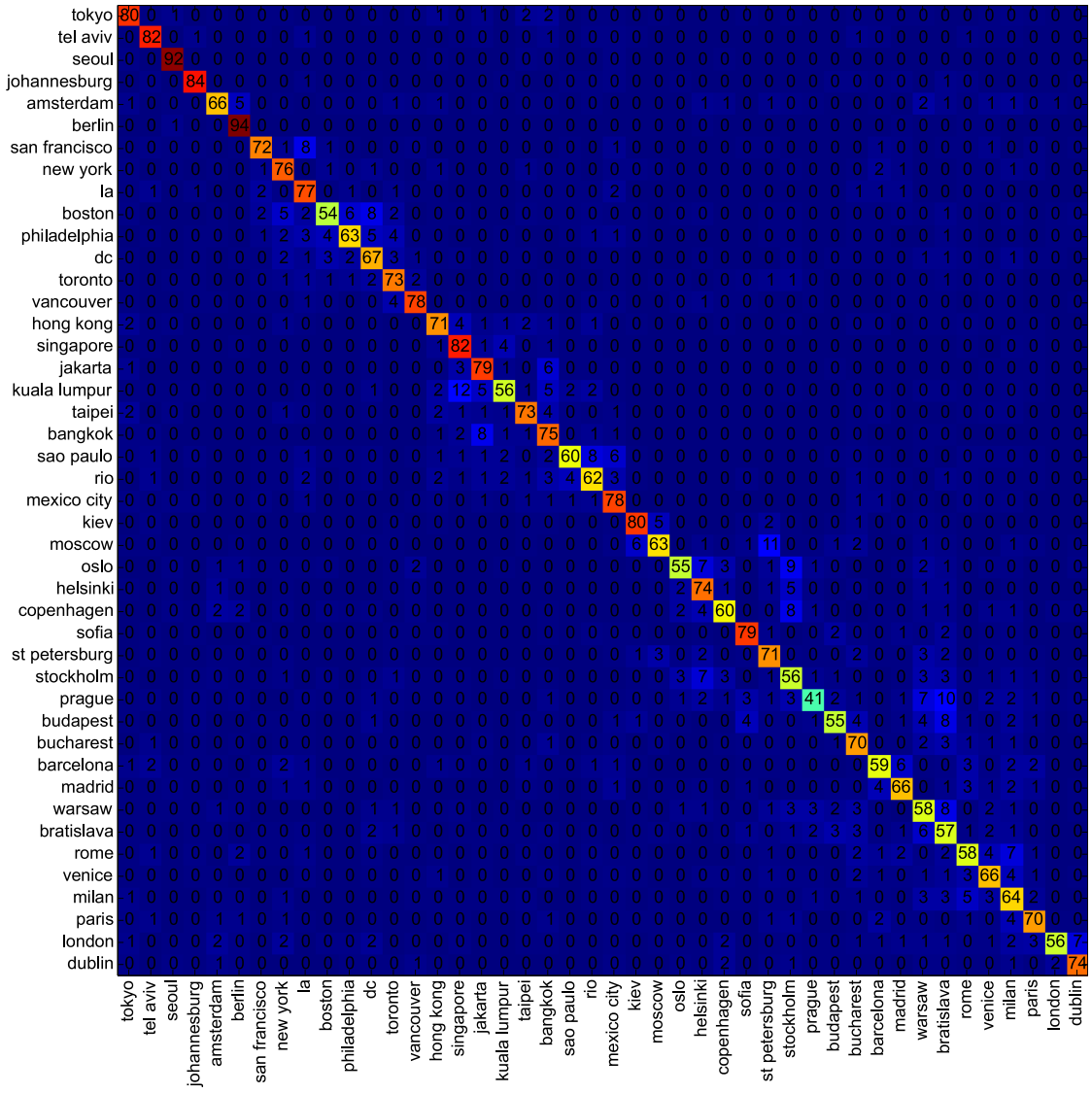


Figure 3.4: **StreetViewCities confusion matrix.** StreetViewCities is a new dataset built from millions of StreetView photos. The GoogLeNet model obtains 69% accuracy on the test set.

		AlexNet/GoogLeNet Normal			FoveaNet Foveal+Normal			BubbLeNet Bubble+Normal
Dataset	Architecture	Normal	Foveal Random	Foveal Best	Normal	Foveal Random	Foveal Best	Bubble Best
CUB-200-2011	AlexNet	0.673*	0.018	0.053	0.561	0.263	0.319	0.481
CUB-200-2011	GoogLeNet	0.810†	0.035	0.147	0.737	0.403	0.459	0.655
Food-101	AlexNet	0.689*	0.034	0.058	0.646	0.341	0.392	0.391
Food-101	GoogLeNet	0.764†	0.042	0.081	0.782	0.451	0.594	0.561
ILSVRC2012	AlexNet	0.574‡	0.009	0.027	0.445	0.210	0.275	0.277
ILSVRC2012	GoogLeNet	0.688§	0.021	0.072	0.435	0.149	0.214	0.235
MIT Indoor67	AlexNet	0.714¶	0.033	0.033	0.688	0.382	0.444	0.450
MIT Indoor67	GoogLeNet	0.714†	0.043	0.067	0.702	0.398	0.533	0.584
MIT Places205	AlexNet	0.506‖	0.017	0.025	0.472	0.272	0.312	0.256
MIT Places205	GoogLeNet	0.504†	0.035	0.025	0.495	0.275	0.346	0.311
StreetViewCities	AlexNet	0.692*	0.027	0.021	0.678	0.272	0.326	0.300
StreetViewCities	GoogLeNet	0.690†	0.037	0.036	0.708	0.328	0.397	0.400
InstagramSeasons	AlexNet	0.716*	0.500	0.530	0.721	0.596	0.585	0.618
InstagramSeasons	GoogLeNet	0.716†	0.510	0.495	0.703	0.634	0.634	0.659

Table 3.2: **Raw classification performance.** A synopsis of classifier performance for different types of input (normal, foveal, and bubble) and each stage of fine-tuning. *Foveal random* indicates that at test time, a single, randomly-sampled focal location is used to process the input that the classifier sees. *Foveal best* and *bubble best* indicate that the network is allowed to search for the focal location that produces the lowest entropy prediction distribution. *Fine-tuned from [32] reference CaffeNet. †Fine-tuned from [32] reference GoogLeNet. ‡Original [32] reference CaffeNet. §Original [32] reference GoogLeNet. ¶Fine-tuned from [94] Hybrid-CNN. ‖Original [94] Places205-CNN.

ent types of input for each dataset—for instance, testing how well a network trained only on normal images performs on bubble images. These classification numbers are in Table 3.2 (using two base network architectures, AlexNet and GoogLeNet [71]). All of the accuracies for normal images in AlexNet/GoogLeNet are competitive with state-of-the-art results.[3] For all datasets, results were computed on the test set, with the exception of ILSVRC12 which was computed on the validation set. This table highlights several inter-

[3] In terms of Indoor67 and Food-101, my networks exceed prior published methods [39] [6]. Baseline networks from which I fine-tuned were provided by [32] and [94]. Therefore, I replicate their reported performance on ILSVRC12 and Places205. My classifier does fall slightly short for CUB-200-2011, but it does come very close to the best reported result of 68.29% [92] (no parts oracle-ft setting).

esting properties of our bubble training method:

Normal networks don't do well on foveal images. If a AlexNet/GoogLeNet (normal-image-trained) classifier is given a foveal image, it will nearly randomly guess, even if the focal location is optimized to produce more confident predictions (column "foveal best"). This motivates the use of fine-tuning to adapt the network to a configuration that can predict both normal and foveal/bubble images.

Foveal images can be classified with a FoveaNet. The FoveaNet numbers show that it is indeed possible to produce a network that has non-trivial performance on images with such restricted content. Many of my datasets have FoveaNets that can perform almost half as well as a AlexNet/GoogLeNet applied to the full normal images (an exception being ILSVRC12). Note that datasets such as Indoor67, Places205, and Food-101 are constructed in such a way that looking at any random location in the photo will often give some clue as to the correct class label. However, datasets such as StreetViewCities often have sections such as the sky that are common across all classes. In addition, by optimizing the focal location to minimize prediction entropy ("foveal best"), foveal performance for each FoveaNet improves significantly, indicating that this "saliency landscape" is something that we can explore using SGD and backpropagation to drive the focal location.

Bubble images can be classified with a BubbLeNet. BubbLeNet shows the performance of the classifier used to mine patches after the entire background has been removed. This is the classifier I use for the other evaluations. The performance of classifying optimized bubble images is significantly above chance. This shows that this method can indeed classify images based on finding single

discriminative bubbles. Interestingly, in some cases bubble image performance exceeds that of the foveal images (e.g., for CUB-200-2011). Adding foveal images also sometimes boosts classification performance for normal images (e.g., Food-101 with GoogLeNet).

3.4.2 Qualitative Results

To get a visual sense of what kinds of elements our algorithm discovers, I ran the patch discovery and clustering algorithm to generate visual elements for each dataset. Figure 3.5 shows a few patches from five clusters of each dataset. For CUB-200-2011, often the best place to look was the birds' heads. However, for some species such as the Geococcyx it is better to look at the body.

For some ILSVRC12 classes it is difficult to actually see the object anywhere in the image. For one such class (hockey puck), the patch selection method does highlight, however, that it is important to look near the ice by the players, rather than elsewhere, such as the players' helmets. Some classes, such as the wine-cellar in Indoor67, have multiple important components. Figure 3.5 shows one component highlighting racks of bottled wine, and another component highlighting wine barrels. Most of these high-scoring patch classes are easily identifiable as a particular element. Like Doersch et al. [15], we find metal railings in Paris as a distinctive element, as well as interesting types of features in other cities (e.g., motorbikes in Jakarta).

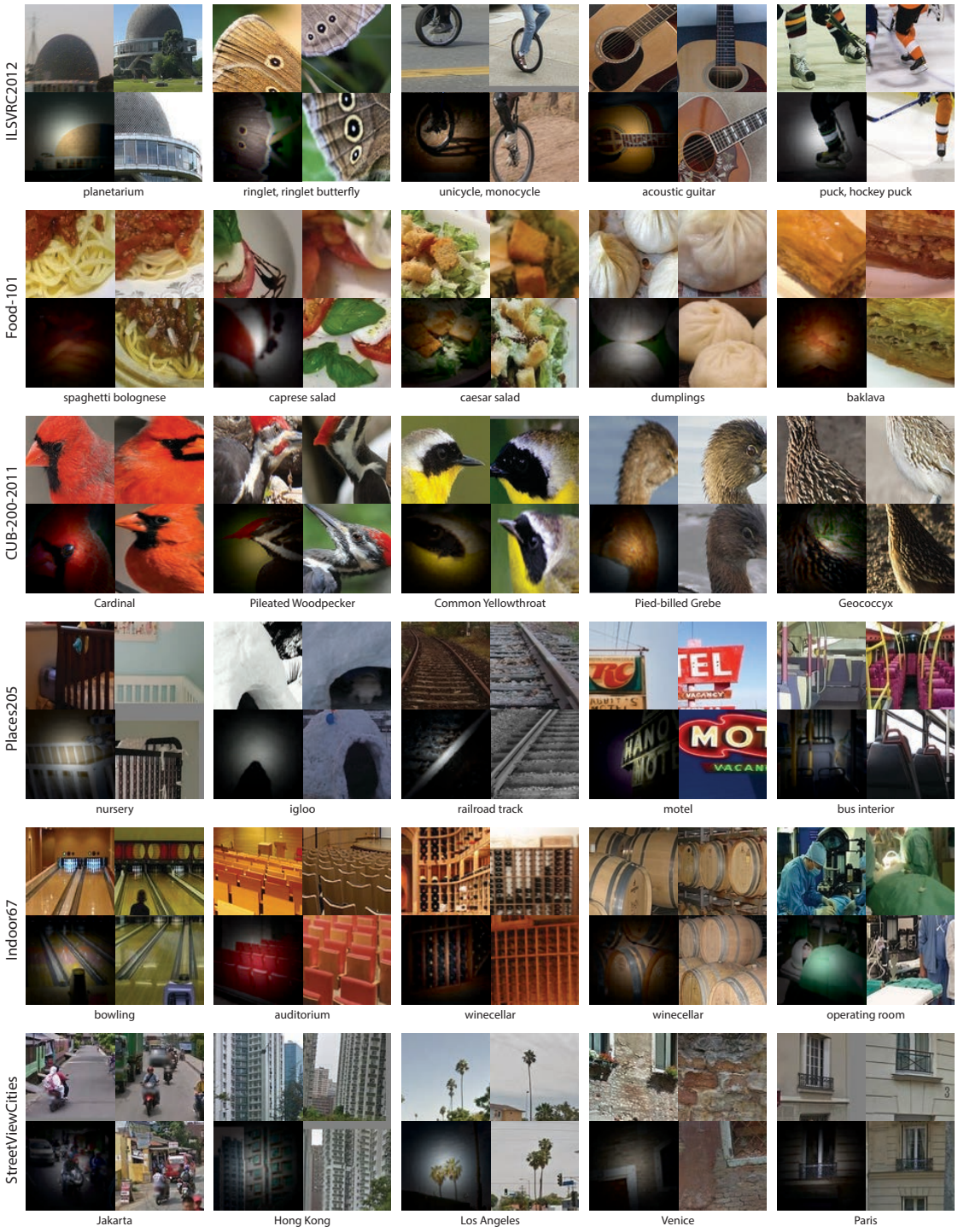


Figure 3.5: **Extracted patch clusters.** Patches extracted from clusters with low entropy. I show one example in the lower left of each cluster with a mask indicating the size of the Gaussian bubble used during discovery. The majority class label is shown below each cluster.

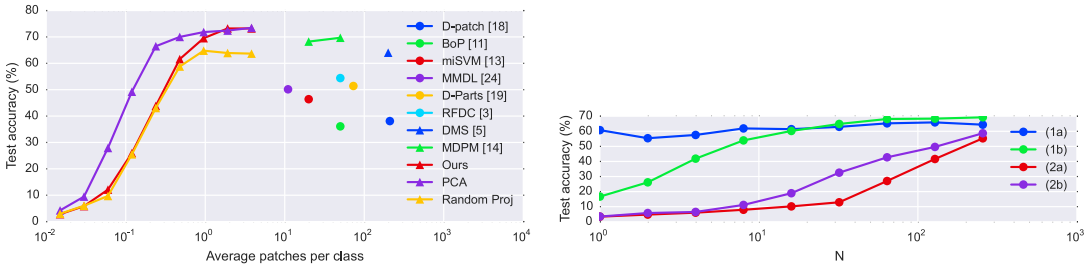


Figure 3.6: **Indoor67 evaluation.** *Top*: The patch selection method allows us to select a very small set of semantic elements as a representation, yet still retain good classification performance.*Bottom*: I validate two alternate BubbleBank design choices. (1) I take the BubbleBank feature vector (a) or PCA applied to the penultimate layer of the CNN (b) and preserve the top N features of each descriptor while setting the remaining to zero. For example, $N = 1$ would produce a feature vector with exactly 1 non-zero located in the position of the component with largest magnitude. (2) I use only the top N ranked bubble features given by (a) a random ranking or (b) the method's ranking.

3.4.3 Discriminability

While the main goal is to extract a human interpretable representation of categories, I also wish to evaluate the method quantitatively—how good of a representation of the data does this method produce? To facilitate comparison with prior work [65, 13, 39, 38, 83, 69, 6] I evaluate using Indoor67. There are often two goals at hand. One is to show that the patch mining method itself produces a set of mid-level features that are well-suited for learning a classifier. However, the baseline CNN produced by fine-tuning outperforms all prior work. Perhaps more interesting is how well the mid-level representation acts as a method of compression. For instance, how well can it encode the 67 classes in Indoor67 with as few visual elements as possible?

Figure 3.6 shows the trade-off between representation size and discrim-

inability for a range of methods, including my own (red line). To create an image encoding with my method, I use a method inspired by the BubbleBank algorithm [11]: I take the top K scoring bubble clusters produced by my algorithm, and, for a given input image run each of the classifiers for these clusters on the input image. These K classifier outputs are then stacked into a vector, and a new SVM is learned for categorizing based on this representation. Compared to bag-of-words methods, this approach detects each visual element once on the entire image, and also takes into account any spatial information encoded in the classifier.

Figure 3.6 (top) plots the performance of my method as a function of number of patches per class, and similarly plots the results of other published methods on this dataset. The results of running PCA or random projections on the penultimate CNN vectors are plotted for comparison (note that these representations are not sparse). As the figure shows, my method requires only a couple of patches per scene class to exceed the performance of the 20-50 patches per class used in prior work (the closest competitor is the MDPM algorithm, which also uses deep learning [39]). This suggests that (a) scenes in Indoor67 can often be classified accurately with only a couple canonical elements per scene, and (b) the invariances baked into the CNN are what make the good performance possible. Despite these invariances, I find that the visual elements are still understandable. Figure 3.5 shows several examples of clusters for different datasets.

Figure 3.6 (bottom) shows the effect of (1) sparsifying BubbleBank feature vectors rather than using PCA on the penultimate CNN layer and (2) using my ranking function to select which clusters are to be included in the BubbleBank

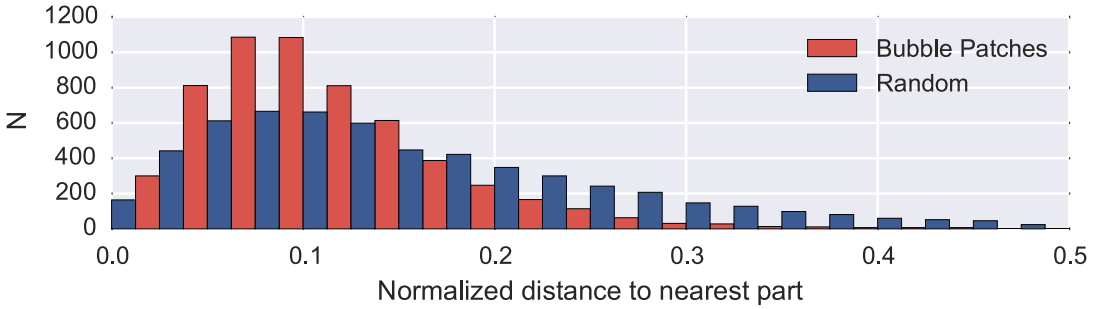


Figure 3.7: **Part localization.** Normalized distance to closed part for the bubble location that maximizes correct class label on CUB-200-2011. The bubble method chooses locations closer to part annotations more often than random guessing.

feature vector to build more concise representations. (1) shows that only a few non-zeros in this vector are needed for good classification suggesting the representations are much sparser compared to dimensionality-reduced vectors from PCA and (2) shows that incorporating clusters according to my ranking leads to better classifiers than selecting clusters at random.

3.4.4 Localization

I perform two experiments to evaluate the localization power of my method. In the first experiment, I measure how closely the most discriminative part per photo in the CUB-200-2011 dataset matches to a part annotation. If a element is close to a part, that suggests we are correctly localizing near an object of interest. I measure this by normalizing image coordinates to $[0, 1] \times [0, 1]$ and computing the Euclidean distance between the centroid of the optimal bubble and its nearest visible part. I compare this to random guessing. The histogram in Fig-

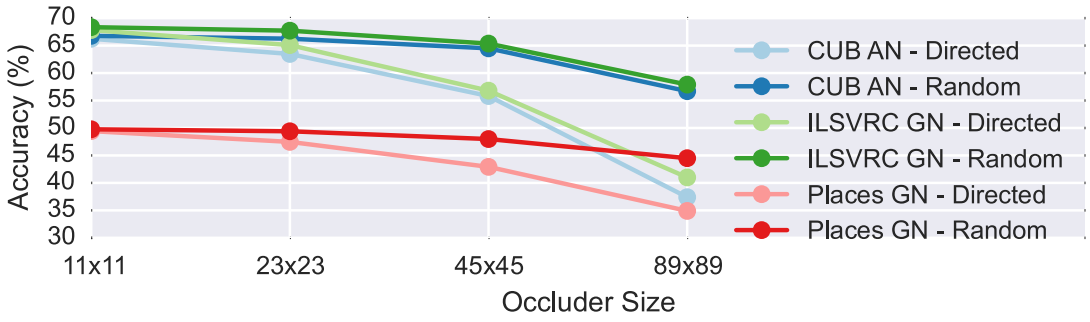


Figure 3.8: **Occlusion analysis.** Given a AlexNet/GoogLeNet classifier, I analyze how much the classification score is impacted by occluding parts at random and as guided by our discriminative bubbles on three datasets. First a discriminative bubble is mined for each photo and then an occluding box is placed at its position. The photo's classification is reevaluated.

ure 3.7 shows that the distributions of element-to-part distances clearly differ where random guessing has a median distance of 0.140 and our method has a median distance of 0.091.

The second experiment measures the performance of the original AlexNet/GoogLeNet classifiers applied to unaltered images and to images that have had a grey occluder box inserted as in [90]. Bubble images can be seen as a additive method; the occluder box is a subtractive method. In this experiment, I select one location for this occluder per image based on the most discriminative part detected by our bubble method. Figure 3.8 compares the performance of several AlexNet/GoogLeNet classifiers on bubble-guided occluders and random occluders. The performance drops significantly, and much more if the occluder is guided by our bubble method than by random guessing, suggesting the bubble method is locating salient features.

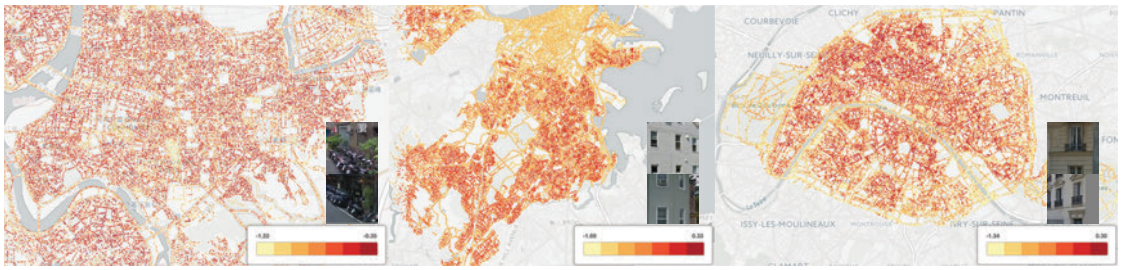


Figure 3.9: Three cities with discriminative bubble cluster activations. Left to right: motorbikes in Jakarta, houses with siding in Boston, and windows with metal railings in Paris.

3.4.5 Exploring Visual Data

The ultimate goal of this method is as a tool for visual discovery. To conclude, I use our technique to explore visual elements of our datasets across time and space.

How do cities differ across the world? Doersch, et al. presented a discriminative framework for comparing visual style across cities in the world. I performed a similar analysis with several orders of magnitude more data and a few clusters are shown in Figure 3.5. Additionally, Figure 3.9 shows the distributions of three separate clusters across the cities that they are predominantly associated with. For example, in Jakarta, motorbikes are often found parked on the side of small roads, but not often in parks. In Boston, houses with siding are found in residential regions, not commercial regions. In Paris, metal window railings are more often found near the center of the city versus the edges.

How do people change over a year? There are many interesting questions about the appearance of people over time, relating to fashion, weather, and so on. As a simple experiment, I took photos of people from NYC from the InstagramPeo-

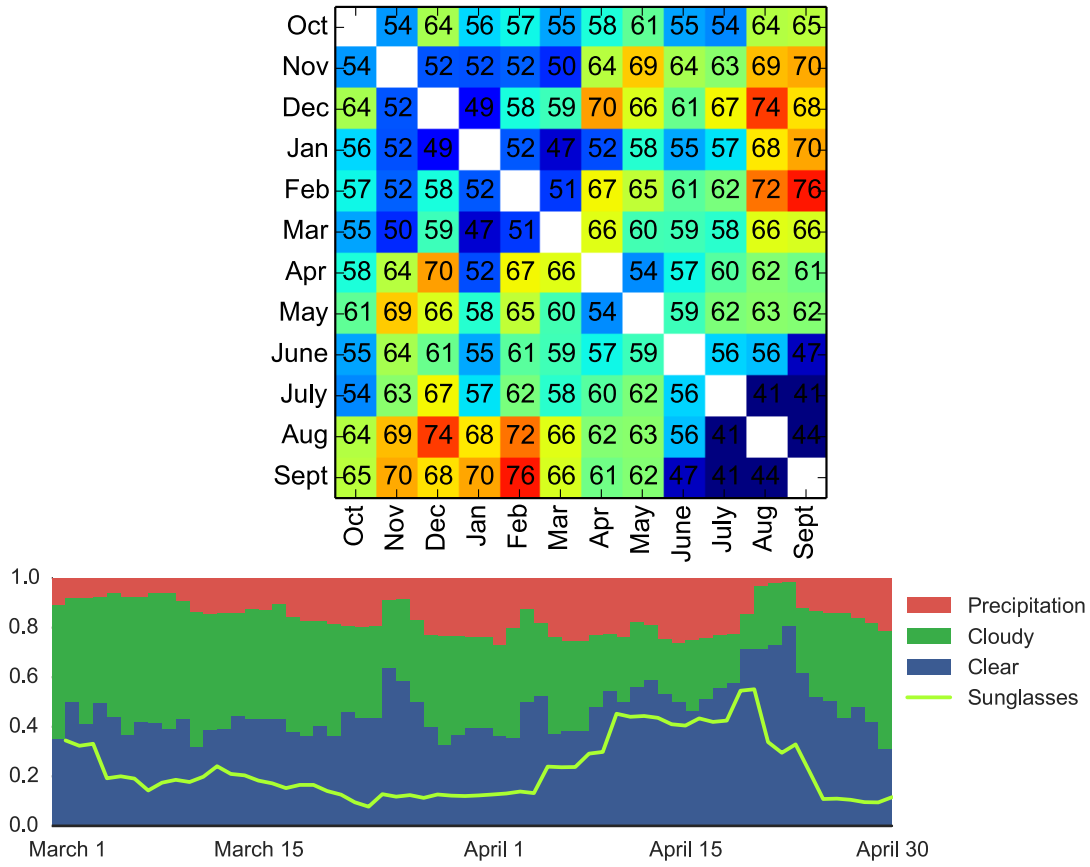


Figure 3.10: **Finding temporal correlations.** *Top*: NYC one-versus-one month classification accuracies. *Bottom*: I found sunglasses to be a discriminative element that makes April in NYC visually distinct from March and correlated them to weather reports.

ple dataset and learned a bubble image classifier that predicts the month when the photo was taken. Rather than train an n-way classifier, I opted to train a shared CNN to produce an intermediate representation on which a large set of one-versus-one logistic regressors could be trained; the entire network was fine-tuned end-to-end. The results are shown in Figure 3.10 as a matrix of one-to-one test accuracies. Some observations: adjacent months are hard to differentiate, and there are clusters of months corresponding to similar seasons (summer and winter). Furthermore, by examining the off-diagonal, I find that the classifier

can differentiate between summer and winter.

However, two adjacent months—March and April—are much more distinguishable than other pairs. What has the network learned? I ran our algorithm on the task of NYC March vs. April. It was clear from the extracted patches that clothing had become lighter, but the top-ranked cluster was sunglasses. By running the detector for this cluster on the 2014 March+April portion of our dataset, I recorded a sunglasses score (percent photos with thresholded score) for each photo. I also collected weather condition reports from wunderground.com and categorized them: clear, cloudy, and precipitation. I found a strong correlation between the occurrence of sunglasses and the weather condition. Namely, the Pearson's r of the "sunglasses" signal to the "clear weather" signal is 0.487 and of the "sunglasses" signal to the "cloudy weather" signal is -0.593 (see Figure 3.10). This illustrates an end-to-end use of my algorithm to explore trends in a dataset.

How do people change over space? Style varies across space as well as time. Can we discover variations in personal fashion between different locales? To explore this question, I ran the method to classify the NYC and London subsets of `InstagramPeople`. Figure 3.11 shows several top-ranked clusters of patches. Accuracy is 60.8%. Sports caps, and stripes and certain text on shirts are more discriminative for NYC, while collared and textured shirts and dark, shiny jackets are more discriminative for London. Some clusters are specialized for color, some hair, and others clothing. However, despite high purity, some clusters remain difficult to interpret suggesting more is required to guarantee interpretability than enforcing spatial compactness alone.

Limitations. These initial experiments required that I use some intuition to

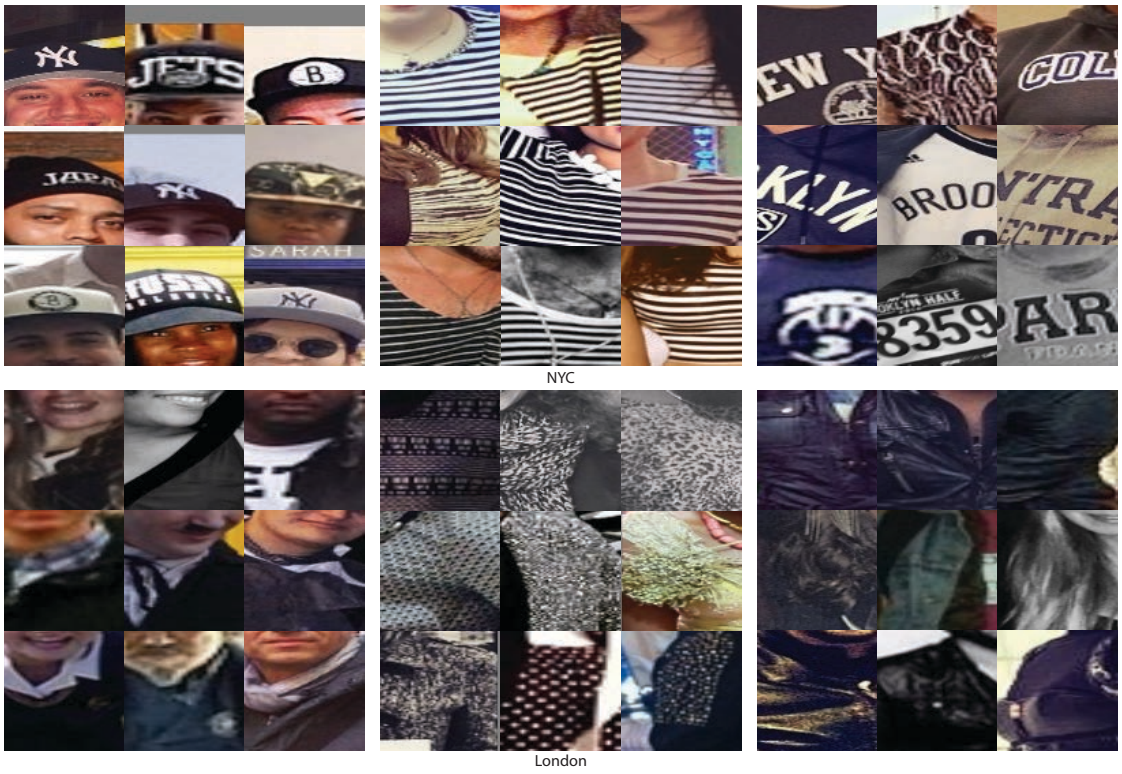


Figure 3.11: **NYC vs. London patches.** I used my method to find what makes people in NYC visually distinct from London. I show clusters correlated with NYC (top) and London (bottom).

figure out what signals might be correlated. An automated discovery system might be able to search through a diverse body of data to find correlations. I also presented additional qualitative results in Figure 3.9 highlighting spatial trends in StreetView photos for which I do not have a way to automatically validate. Finally, the high invariance of the CNN features can still cause some clusters to be **too** invariant and appear noisy.

3.5 Limitations

The assumption that each class in the dataset can be described by small, localized content could be violated, in which case this method will be unable to find discriminative patches. That is, if the entire image needs to be seen at once to make a correct classification, then this method will find no single patch that is adequate. This failure mode has not been observed in my experiments. The second limitation is that the patch mining routine selects only the most discriminative patch per image. If two "concepts" are correlated in the imagery, and one is selected to be a discriminative patch, then it is likely that the second will never be revealed as its own discriminative patch. Finally, this form of automated visual vocabulary offers little control to an analyst. For example, an analyst may have prior domain knowledge that a certain visual attribute is indeed discriminative. This methodology does not allow them to integrate that knowledge.

3.6 Conclusions

This chapter presented a simple, scalable, new method for discovering visual elements in large datasets that directly builds on CNNs via a new form of modified training data. I believe techniques such as the one presented here can start to open the door to new kinds of visual discovery. In Chapter 4, I build off of the experiments in this chapter to explore fashion in diverse populations around the world and across time. It would also be intriguing to build on this preliminary work to explore design, art, and other visual arenas to identify trends and influences. There are also a number of other interesting areas for future work. One

key problem is that humans still have to interpret the clusters—this one represents hats, another represents a certain pattern, etc. Doing this kind of labeling automatically—perhaps using text or other cues found on the web, would further enable automatic interpretation of discovered elements.

CHAPTER 4

MEASURING HUMAN TRENDS WITH SOCIAL MEDIA IMAGES

In this age of big data, one of the most compelling opportunities is to use all available information to measure the world in ways that were never before possible. Large amounts of data—for instance, OCRed scans of centuries worth of books—coupled with even basic tools for exploring this data, can yield powerful tools for scientists to study our history, culture, and behavior [45]. These opportunities are magnified by the increasing power of machine learning, and by the massive scale at which humans are generating cultural artifacts on social media. For instance, by applying natural language processing to millions of Twitter messages, we can discover relationships between time of day and mood that leverage sample sizes much larger than any traditional study might hope to accomplish [27].

To date, most of these new kinds of big data analyses have been limited to structured data like user interactions on social networks or textual data like books and tweets. However, a tremendous cache of visual information about our world is locked in images, and particularly in images of people, including the billions of photos uploaded to photo-sharing services each day. Imagine a future anthropologist with access to trillions of photos of people, taken over centuries across the world, and to the right tools to analyze these photos and derive insight. What kinds of new questions can be asked and answered? This problem area of *automatic visual discovery* is still new, but is starting to gain attention in vision and graphics [15, 82, 24]. This work takes a step towards this vision by analyzing geo-spatial trends in style across millions of images from social media.

In this chapter, I focus on clothing as it is a critical aspect of both the visual world, and our daily lives. Individuals make fashion choices based on geography, weather, culture, and personal preference. Understanding trends in fashion is valuable for many applications including costume design in movies and games, and analytics for fashion designers and manufacturers. This kind of analysis currently is done manually by artists, for example, by collecting and inspecting images from relevant locations and times using historical photographs.

I aim to extract meaningful insight about the geo-spatial and temporal distributions of clothing, fashion, and style around the world through social media analysis *at scale*. I do so through a combination of (1) millions of photos spanning the world across years—photos uploaded to social media websites by everyday users—and (2) powerful machine learning methods based on deep learning. The results of these algorithms enable automated and manual analyses that can be used to derive meaningful insight in the form of trend reports, map-based visualizations, and artist reference material. This approach also demonstrates the utility of machine learning tools that have vastly simplified the process of making real world measurements of visual trends.

The main challenges are in collecting training data at scale and applying proper analysis and calibration to handle classifier errors. I first downloaded hundreds of millions of images from social media sites, detected people in them, and built a training set of labeled clothing attributes using crowdsourcing for a small percentage of the total photo collection. These clothing attributes include binary attributes such as *wearing-hat*, as well as multi-class attributes such as *clothing-category* (shirt, sweater, etc). I then trained discriminative classifiers to recognize these attributes in new imagery, and applied these classifiers to

millions of images of people. The results immediately allow us to measure trends—such as "are people wearing more hats over time?"—at sufficient scale to achieve statistical significance.

A byproduct of this model is an internal representation that can be interpreted as an embedding of people images in a high-dimensional space shown in Figure 4.1. By clustering images in this space, I can go beyond basic fashion attributes to automatically identify visually distinctive style clusters. I perform further automated and manual exploration of these clusterings to derive insight about finer-grained styles (e.g., black shirts with text). Further, I use these clusters to identify representative style for cities around the world. I show a potential use of these representative styles to assist artists in illustrating people in a given city.

This work has the following contributions. I introduce:

1. a large-scale, annotated dataset of people containing 27k images, each with 12 clothing attributes,
2. a methodology for analyzing millions of photos to derive visual insight,
3. the first-of-its-kind analysis of global and per-city fashion choices and trends using unsupervised clustering techniques,
4. and a proof-of-concept application to assist artists by providing them with representative style references for cities.

Figure 4.1: t-SNE visualization of learned feature space. The style embedding is 1024 dimensions and is visualized in 2 dimensions here. Several combinations of attributes are highlighted. For example, white shirts with text, plaid shirts, black v-neck shirts, etc. are all styles discoverable in this embedding.

4.1 Related Work

I now discuss the rich history of clothing and fashion understanding in the computer graphics and vision literature as well as work that derives insight from social media in the field of natural language processing.

Visual Discovery. We are beginning to see work in graphics and vision that mines world-wide imagery to (1) discover visual trends and (2) measure aspects of the world over space and time. Much of this work has looked at *places*. Doersch et al. pioneered the idea of *computational geography* and explore the distinctive visual characteristics of cities through Street View imagery [15], driven by weakly supervised methods for finding discriminative elements [65, 13]. Such visual characteristics can also be correlated to other properties of cities, such as perceived safety [2, 50]. These correlations can then be used to automatically predict such properties across a city simply from ground level images. Beyond analysis of cities, computer vision techniques have been used to answer questions such as "is there snow in this photo?" or "are there clouds in this webcam?" which in turn can be used at scale to estimate maps of snow or cloud cover over large regions of the world [91, 82, 49].

Others have used large image collections to study people, including explorations of the relationship between facial appearance and geolocation [30] and time [59]. Ginosar et al. analyzed tens of thousands of face portraits scanned from a century's worth of American high school yearbooks, and used this data in tasks such as automatically tracking the growth in smiles in portraits over time and mining for hairstyles [24]. My work shares a similar overall motivation, but I examine a much larger and spatially broader collection of millions of

images of people across the globe. I also create annotations that enable analysis of clothing styles and not just faces.

Visual Understanding of Clothing. Clothing and fashion has played an important role in computer vision research over the last several years, as it combines a unique blend of attribute-level understanding and social context. I build on previous work that describes people in terms of clothing and other fashion attributes, such as "wearing glasses," or "short sleeves," and recognizes these attributes in new images [9, 7, 5, 93]. Beyond classifying attributes, other work also produces full pixel-level clothing segmentations [86, 85, 88], or recognizes specific (or similar) products rather than general attributes [12, 80, 34]. Other work categorizes clothing in terms of explicit, named styles. Kiapor et al. used a game to analyze coarse fashion styles such as "hipster" and "goth" to build intra-style classifiers (i.e. is this photo goth or hipster?) and inter-style ranking functions (i.e. how hipster is this person's fashion?) [35]. In my work, I learn coarse grained attributes from images around the world, but use this to organize images visually so as to perform more fine-grained clustering.

Clothing Trends. Temporal and spatial trends related to fashion have been explored in the computer vision literature, but often on small samples sizes and with limited statistical analysis. Hidayati et al. analyze catwalk images from NYC fashion shows to find style trends in high-end fashion [29] Simo-Serra et al. analyzed `chictopia.com` and found a weak positive correlation between fashionability of photos and the wealth of the nation in which they were taken, and found temporal trends such as a sudden increase in popularity for "heels" in Manila [63]. [80] showed that seasonal trends related to styles such as "floral", "pastel", and "neon" could be obtained using their fashion clas-

sifiers with the peak occurring in the springtime. However, these trends were not evaluated for statistical significance, and I argue that without the massive amounts of data I advocate, it is difficult to find such significance.

Clothing Datasets. Historically, fashion datasets have largely been very limited in scale, and biased towards images of the fashion-conscious. For example, a number of researchers makes use of fashion-centric social networking sites such as `fashionista.com` and `chictopia.com` [88, 63, 87, 86, 85]. These sites enable users to upload photos and annotate articles of clothing for the express purpose of modeling their style. Other work makes use of online clothing retailers such as Amazon.com, eBay, and ModShop to build similar datasets [41, 12, 34]. The images of these websites are relatively clean and organized according to tags, facilitating the creation of annotated datasets on the scale of hundreds of thousands of people [9, 5]. However, my goal is to measure spatio-temporal trends over the entire world, and to obtain highly certain statistics, so very large data is key—100K images is not sufficient, once the data is sliced in time and space. Furthermore, these sites are designed to explicitly engage only the most fashion conscious people in the world, not every person in the world. Therefore, one can expect certain types of bias in these approaches if the goal were to analyze all the people of the world. My approach is to build a massive dataset from photos on social media, which can be gathered at larger scale and are more representative of everyday fashions. However, these images are also much more noisy and require annotation. My strategy is to annotate a small subset of such a large dataset manually, and use a machine learning classifier to generalize the result to the rest of the dataset.

4.2 Data

My dataset consists of three key parts: (1) photos, (2) the people in those photos, and (3) clothing attributes of those people.

Photos. The photos used in this analysis were acquired via Instagram, a popular mobile photography social network owned by Facebook. I chose Instagram due to the sheer volume of uploads—an advertised 80 million photos per day[1]—as well as their providing a *public* photo search API. Users can query this API to retrieve images that have been uploaded within a 5 kilometer radius of a specified latitude and longitude and within 5 days of a specified date.[2]

Given that the spatial distribution of global uploads is unknown a priori, what is the best strategy to sample photos? My work aims to conduct two types of experiments. First, a comparison of major population centers of the world, which requires high-density samples in major cities. Second, a nation- or continent-wide analysis of trends. To achieve these goals, I sampled photos according to two strategies. First, to get a good sampling of photos in urban centers, I built a predetermined short list of 44 major cities, each with a lat/long approximating the city center. Each city was sampled and Instagram was queried with that coordinate. The second sampling strategy approximated the global distribution with a second dataset, the Flickr 100M photo dataset [77]. I extracted the approximately 48 million geotags from that dataset, and used these geotags as a geographic distribution from which to sample.

For both strategies, I uniformly sampled a 5-day window from June 2013—the earliest date that Instagram provided results—till June 2016. For each photo

[1] `https://www.instagram.com/press/`
[2] `https://www.instagram.com/developer/`

retrieved, I also stored geolocation and timestamp, the two pieces of information our analysis focuses on.

People. To find people in photos, I ran two state-of-the-art vision methods on downloaded photos, the first to detect and localize faces, and the second to estimate the visibility of the rest of the body. To detect faces, I used the API provided by Face++.[3] The output of this detector is a bounding box for the face, as well as facial landmark keypoints. I use the bounding box to spatially align people in this analysis. To determine body visibility, I use the Deformable Part Model [19, 25] and the provided pedestrian detector trained on VOC2010 [17]. This detector is comprised of a mixture model of human visibility configurations. For example, one component is dedicated to fully visible bodies, one is dedicated to people visible from the waist up, and the last is dedicated to people visible from the shoulders up. I used the detection bounding box as an estimate of the person's visibility.

Given a set of faces and visible bodies in a photo, I pair them up using a simple distance-based heuristic. For each face, I compute a canonical crop based on the position and scale of the face detection, illustrated in Figure 4.2. If the crop extends beyond the visible body bounding box, I discard the detection.

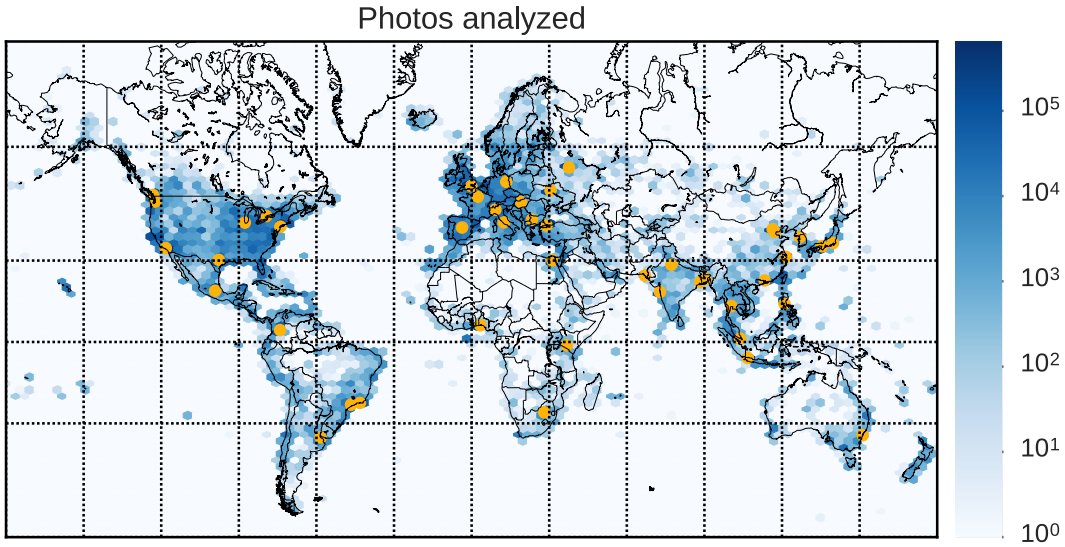
In total, I retrieved and processed several hundred million photos of which 37.7 million had at least one successful pedestrian detection at any of the three visibility levels. For this work, I chose to select detections with at least the face and torso visible. I do not analyze the lower part of the body since the legs are often occluded. After generating canonical crops from the face detections and filtering on face and torso visibility, I was left with 14.5 million images of

[3] `http://www.faceplusplus.com/`

Figure 4.2: Given an approximate person detection and a precise face localization, I generate a crop with canonical position and scaling to capture the head and torso of the person, shown left. Additional examples from our dataset are shown on the right.

people. Distributions of these people at both the city-level and at the world-level are shown in Figure 4.3.

Clothing annotations. In addition to photos and people, I wanted to collect fashion and style related annotations. For example, what kind of pattern is this shirt? What is the neckline? There were several clothing attributes, such as type, pattern, and color that I knew a priori and wished to study. I surveyed prior work in the area of clothing attribute recognition and arrived at a consolidated list of attributes which I believed were frequent and identifiable in our imagery. I started with several of the attributes presented in [9] and made a few adjustments to more clearly articulate collar type as in [12]. Finally, I added a



Austin	Bangkok	Beijing	Berlin
Bogotá	Budapest	Buenos Aires	Cairo
Chicago	Delhi	Dhaka	Guangzhou
Istanbul	Jakarta	Johannesburg	Karachi
Kiev	Kolkata	Lagos	London
Los Angeles	Madrid	Manila	Mexico City
Milan	Moscow	Mumbai	Nairobi
New York City	Osaka	Paris	Rio de Janeiro
Rome	São Paulo	Seattle	Seoul
Shanghai	Singapore	Sofia	Sydney
Tianjin	Tokyo	Toronto	Vancouver

Figure 4.3: Geo-spatial distribution of images collected from Instagram. Images were collected according to two distributions: a set of the world cities listed (yellow dots), and distributed over the entire Earth (blue).

Attribute count	No	Yes
Wearing Jacket	18078	7113
Collar Presence	16774	7299
Wearing Scarf	23979	1452
Wearing Necktie	24843	827
Wearing Hat	23279	2255
Wearing Glasses	22058	3401
Multiple Layers	15921	8829

Table 4.1: Number of MTurk annotations collected for binary attributes indicating the presence of jackets, collars, scarves, necktie, hat, glasses, and multiple layers of clothing.

Clothing Category	Sleeve Length	Neckline Shape	Clothing Pattern	Major Color
Shirt (4666)	Long sleeve (13410)	Round (9799)	Solid (15933)	Black (6545)
Outerwear (4580)	Short sleeve (7145)	Folded (8119)	Graphics (3832)	White (4461)
T-shirt (4580)	No sleeve (3520)	V-shape (2017)	Striped (1069)	2+ colors (2439)
Dress (2558)			Floral (885)	Blue (2419)
Tank top (1348)			Plaid (532)	Gray (1345)
Suit (1143)			Spotted (241)	Red (1131)
Sweater (874)				Pink (649)
				Green (526)
				Yellow (441)
				Brown (386)
				Purple (170)
				Orange (162)
				Cyan (33)

Table 4.2: Attributes with multiple values include the type of clothing, sleeve length, neckline shape, clothing pattern, and major color. In parentheses are the number of MTurk annotations I collected for each attribute.

few attributes of my own such as *wearing-hats* and *wearing-jackets* as well as an attribute indicating whether or not more than one layer of clothing can be seen. (e.g., A suit jacket over a shirt or an open jacket showing the t-shirt underneath.) The full list of attributes is shown in Tables 4.1 and 4.2.

A 27K person subset of our dataset was labeled with these attributes using Amazon Mechanical Turk. Mechanical Turk users were given a list of 40 pho-

tos, presented one-by-one, and were asked to select one of several predefined labels. Additionally, they were allowed to indicate that either the detection was faulty in one of several ways such as containing more than one person or containing no person at all. Within each set of 40 photos were two photos with known ground truth label. If a worker failed at least five of these sentinels and their sentinel failure rate was above 20%, I prevented them from performing any more work for our tasks. Compensation for a single attribute on a set of 40 photos was 0.10 USD. Average wage was approximately 4 USD / hour. Each image was labeled by 5 separate users and a different set of 5 workers labeled each attribute. A label was accepted if 3 / 5 users agreed, otherwise it was not included. Approximately 4K USD was spent on these annotations.

Dataset bias and limitations. Any dataset has inherent bias, and it is important to be cognizant of these biases when deriving conclusions from the data. Instagram is a mobile social network, so participants are more likely to be in regions where broadband mobile Internet is available. State-imposed censorship will also affect results in certain regions such as mainland China where Instagram is occasionally blocked. Competitors vary by region and therefore some people might be more likely to upload to one of those competitors than to Instagram. People of certain ages might be more or less likely to upload to any social network. Cultural norms can affect who, when, where, and what people photograph. The face detector I use is an off-the-shelf component for which no reported statistics regarding bias for age, gender, and race are made available. The person detector has also not been evaluated to determine bias in these three factors either. These factors impact who will and will not be properly added to our dataset, which could introduce bias. As vision methods mature, analyses for such bias will become increasingly important.

4.3 Methodology

To make measurements across the millions of photos in our dataset, I wish to generalize labeled data from the 27K labeled images to the entire dataset. For instance, to measure instances of glasses across the world, I wish to use all of the *wearing-glasses* annotations as training data and predict the *wearing-glasses* attribute in millions of unlabeled images. To do so I use Convolutional Neural Networks [37] (CNNs), a machine learning method. CNNs map an input (e.g., the pixels of an image), to an output (e.g., a binary label representing whether or not the person is wearing glasses), automatically learning a rich hierarchy of internal image features given training data. Next, I describe the architecture of our particular CNN, how I trained it using the labeled attribute dataset, and how I evaluated the classifier's performance on the clothing attribute prediction task. I then discuss how and when it is appropriate to generate predictions from this CNN across millions of photos and use these predictions to estimate real world statistics.

4.3.1 Learning attributes

A key design decision in creating a CNN is defining its network architecture, or how the various types of layers are composed. The final layer of a typical CNN outputs a multi-dimensional (e.g., 1024-D) feature vector, which is followed by a "linear layer" (matrix product) operation that outputs one or more prediction scores (e.g., values indicating the probability of a set of attributes of interest, such as *wearing-glasses*). The base architecture for my CNN is the "GoogLeNet" architecture [72]. While several excellent alternatives exist (such as VGG-16 and

VGG-19 [64]), GoogLeNet offers a good tradeoff between performance on tasks such as image classification [57] and speed. As with other small datasets (fewer than 1 million training examples), the attribute prediction dataset makes it difficult to train such a CNN from scratch without overfitting. Instead, I start with a CNN pretrained on image classification (on ImageNet ILSVRC2012) and fine-tune the CNN on my data. After ImageNet training, I discard the last linear layer of the network to expose the 1024-dimensional feature output and then append several $1024 \times N_i$ linear layers in parallel where N_i is the number of class labels for attribute i.

Training details. I train the CNN with the following procedure: for each attribute, I use stratified sampling to first select an attribute label and then a particular example with that label. This is to counteract the implicit imbalance within each attribute (e.g., not wearing a hat is more common than wearing a hat). I do this 32 times to build a single mini-batch. Using this mini-batch, I apply the forward and backward passes of the CNN to compute the parameter gradient with respect to the cross-entropy loss. I do this for each attribute, accumulating the gradients. Then I use SGD with momentum = 0.9, learning rate = 10^{-2}, and weight decay = 10^{-4} to update the parameters. I fine-trained the CNN for 6,000 iterations ($\times$14 attributes) after which the mean class accuracy on the validation set stopped increasing. 80% of the labeled data was used for training, 10% was used as validation to determine when to stop training as well as for additional analysis in this section, and 10% was used for a final evaluation of the attribute classification task.

Table 4.3 summarizes the accuracy and mean class accuracy for each attribute of our learned classifiers on the held-out test set. Figure 4.4 shows a set

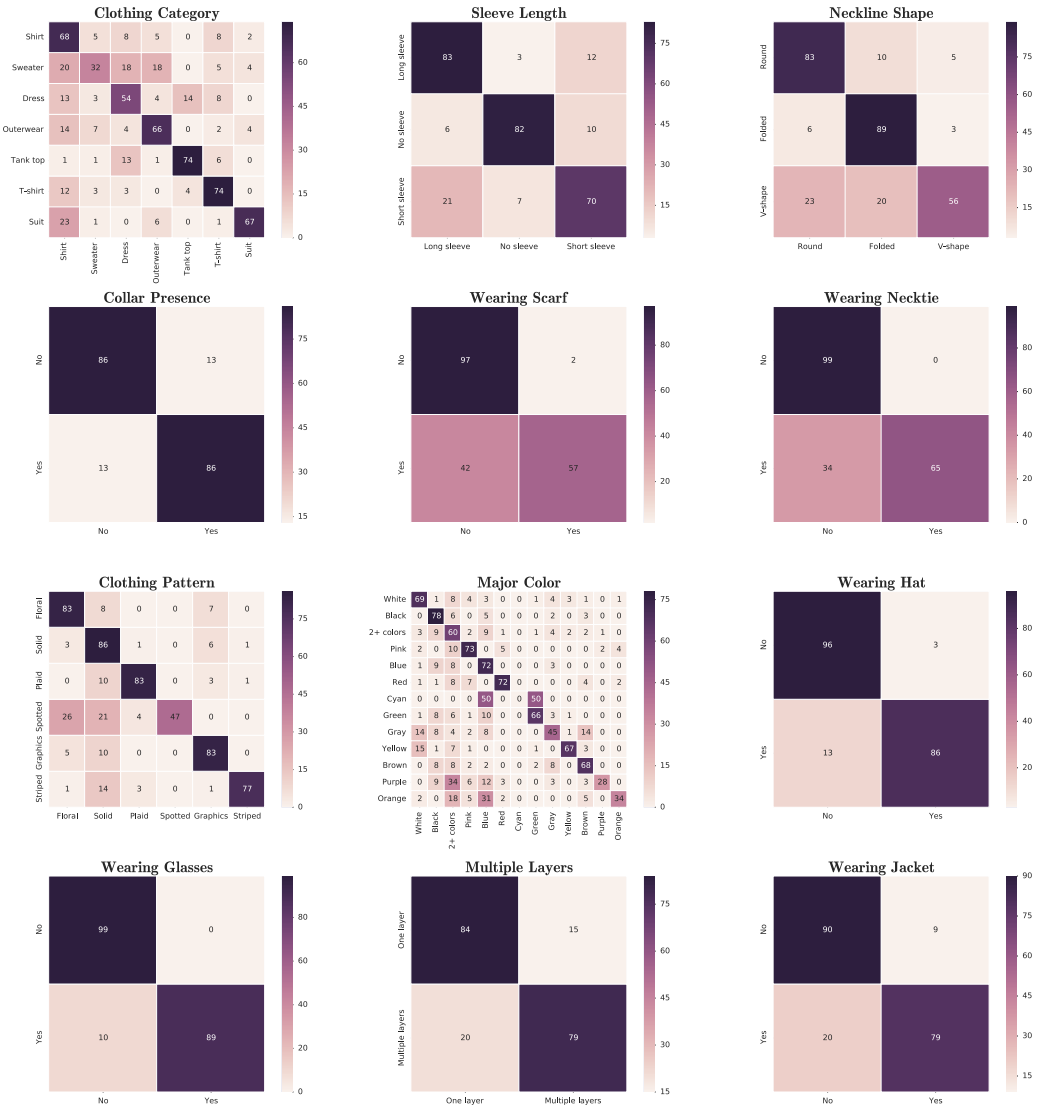


Figure 4.4: Confusion matrices for each of the clothing attributes. Given class labels are vertical and predicted class labels are horizontal. Each row has been normalized such that a cell indicates the probability of a classification given some true label.

Attribute	Our CNN (acc. / mean class. acc.)	Random guessing	Majority guessing
wearing jacket	0.869 / 0.848	0.5 / 0.5	0.698 / 0.5
clothing category	0.661 / 0.627	0.142 / 0.142	0.241 / 0.142
sleeve length	0.794 / 0.788	0.333 / 0.333	0.573 / 0.333
neckline shape	0.831 / 0.766	0.333 / 0.333	0.481 / 0.333
collar presence	0.869 / 0.868	0.5 / 0.5	0.701 / 0.5
wearing scarf	0.944 / 0.772	0.5 / 0.5	0.926 / 0.5
wearing necktie	0.979 / 0.826	0.5 / 0.5	0.956 / 0.5
clothing pattern	0.853 / 0.772	0.166 / 0.166	0.717 / 0.166
major color	0.688 / 0.568	0.077 / 0.077	0.197 / 0.077
wearing hat	0.959 / 0.917	0.5 / 0.5	0.904 / 0.5
wearing glasses	0.982 / 0.945	0.5 / 0.5	0.863 / 0.5
multiple layers	0.830 / 0.823	0.5 / 0.5	0.619 / 0.5

Table 4.3: Test set performance for the trained CNN. Since each attribute has an unbalanced set of test examples, I report both accuracy and mean classification accuracy. Included are baseline performance scores for random guessing and majority class guessing for reference.

of confusion matrices, one per attribute, illustrating what attributes tend to get confused for one another (e.g., spotted shirts can be mistaken for floral shirts). While the classifiers perform quite well in general (e.g., *wearing-hat* has an accuracy over 90%), even state-of-the-art CNNs make mistakes, and so all of the classifiers have an inherent error level. I now discuss how I take this error into account.

4.3.2 Measuring attributes at scale

My aim is not only to build a classifier for these pre-determined clothing attributes and maximize some performance criterion on a test set, but to take that learned classifier, apply it to a much larger corpus of photos spanning years worth of the world's photos, and provide a tool for discovering interesting

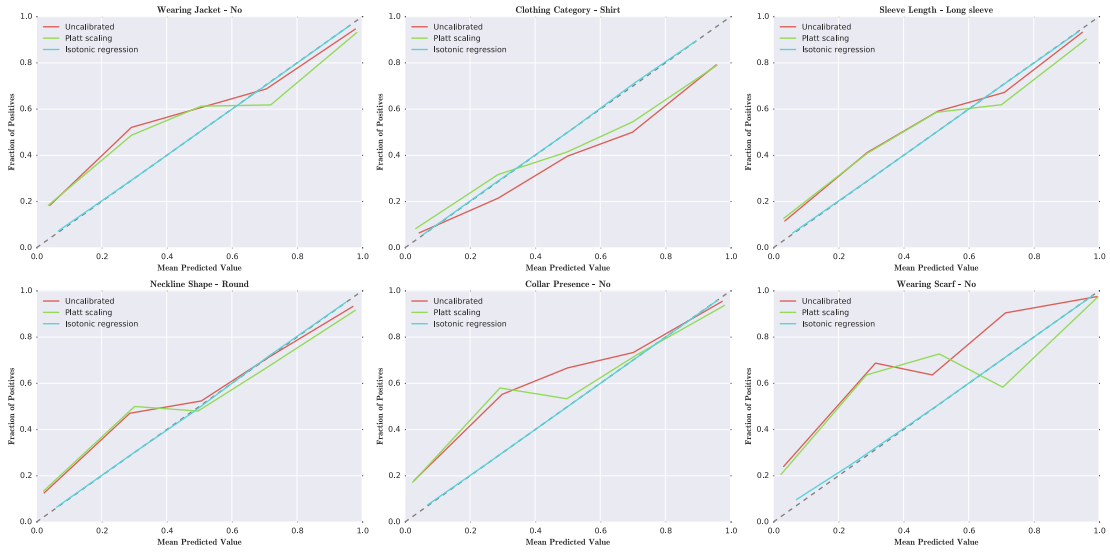


Figure 4.5: Reliability curves for a few of the clothing attributes. A well-calibrated reliability curve has fraction of positives equal to mean predicted value. Neural networks tend to produce well-calibrated probabilities, but some minor improvement is made using Isotonic regression for additional calibration.

trends. However, the classifiers are not perfect, and any real-world measurement is going to have some noise. While it is inevitable that a classifier will have some non-zero variance, the hope is that the estimation is unbiased, that is, as I increase the sample size, the computed statistic, such as the detected percentage of people wearing hats in photos will converge on the true percentage of people wearing hats. However, by examining the confusion matrices in Figure 4.4 it becomes clear that this will not be the case. For example, the proportion of images featuring people wearing neckties is much less than 50%. For this attribute, the classifier predicts **No** correctly 99% of the time, but **Yes** correctly only 65% of the time. Therefore, the estimate of the percentage of people wearing neckties in photos will be biased towards **No**.

One way to mitigate this bias is to calibrate the scores of a classifier using a function that models the posterior probability of the classification being correct given the classification score. There are several methods for this type of calibration, including Platt scaling (an application of logistic regression) [53] and isotonic regression. Others have noted that neural networks tend not to require calibration as severely as classifiers such as the SVM and boosted decision trees [52]. Nevertheless, I found that my networks benefited from calibration by applying isotonic regression to our validation set. Figure 4.5 shows the generalization performance of calibration for several attribute labels. Half of the validation set was used for training the regressor and the other half was used to generate reliability curves. A properly calibrated curve should be very close to the identity function ($y = x$). Isotonic regression typically requires more data to avoid overfitting, yet is more flexible when the calibration required is not sigmoidal, an underlying assumption of Platt scaling. The generalization shows that these estimates do not exhibit severe overfitting and therefore I opt to use isotonic regression trained on the entire validation set to calibrate the scores.

4.3.3 Visually consistent style clusters

The predicted attributes are immediately useful in plotting and analyzing trends, such as "how is the proportion of people wearing black in Los Angeles changing over time?" as I demonstrate in Section 4.4. However, I would like to analyze clothing styles beyond these attributes in a number of ways:

- Identify common, visually consistent combinations of these basic attributes (e.g., *blue sweater with jacket and wool hat*).

- Identify styles that appear more frequently in one city versus another or more frequently during particular periods of time.
- Identify finer-grained, visually coherent versions of these elements (e.g., sports jerseys in a particular style).

To achieve these goals, I use clustering to identify recurring *visual themes* in the embedded space of people. The idea is that by using supervision from attributes, I can learn how to organize people visually according to fashion, and clustering in the 1024-D embedded feature space will reveal such visual themes. Once I identify these clusters, I can further characterize cities and times in terms of these clusters—for instance, I might discover that some clusters are heavily correlated with one particular city (e.g., "What makes Parisians look like Parisians?"), as I explore in Section 4.4.

I ran clustering on a subset of the full dataset, for efficiency and to achieve balance between different regions of the world and time. Examples were divided into bins by city and by week (e.g., Paris in week 26 of 2015.) For bins with fewer than N examples, I selected all examples, and for bins with more than N examples, I randomly selected N examples. $N = 4000$ in these experiments to retrieve approximately 5.4M examples in total.

For each cropped person image in this set, I compute its 1024-D CNN feature vector and L_2 normalize this vector. I run PCA on these normalized vectors, and project onto the top principal components that retain 90% of the variance in the vectors (in our case, 165 dimensions). To cluster these vectors, I used a Gaussian mixture model (GMM) of 400 components with diagonal covariance matrices. Each person is assigned to the mixture component (cluster) which maximizes the posterior probability. The people assigned to a cluster are then sorted by

their Euclidean distance from the cluster center. Although more clusters would increase the likelihood of the data under the model it would also create smaller clusters so I selected 400 as a compromise between cluster size and maximizing likelihood.

The resulting 400 clusters have sizes ranging from 1,001 to 115,165. Each cluster tends to represent some combination of our chosen attributes. I assign further meaning to our clusters by ranking them and ranking the people within each cluster. Further exploration of this clustering is presented in Section 4.4.

4.4 Exploratory Analysis and Applications

A key goal of this work is to take automatic predictions from machine learning tools and derive statistics from which we can find trends. Some of these may be common sense trends that validate the approach, while others may be new, unexpected relations. This section describes several ways in which one can explore this data, insights one can derive through this exploratory data analysis, and other graphics applications of the data and analysis.

4.4.1 Analyzing Trends

Color. How does color vary over time? Figure 4.6 shows plots of the frequency of appearance (across the entire dataset) of several colors. White and black, for example, exhibit a highly periodic trend; white is at its maximum (> 20%) in September, whereas black is a nearly reversed, much more common in January. However, if I break these plots down per city, I find that cities in the Southern

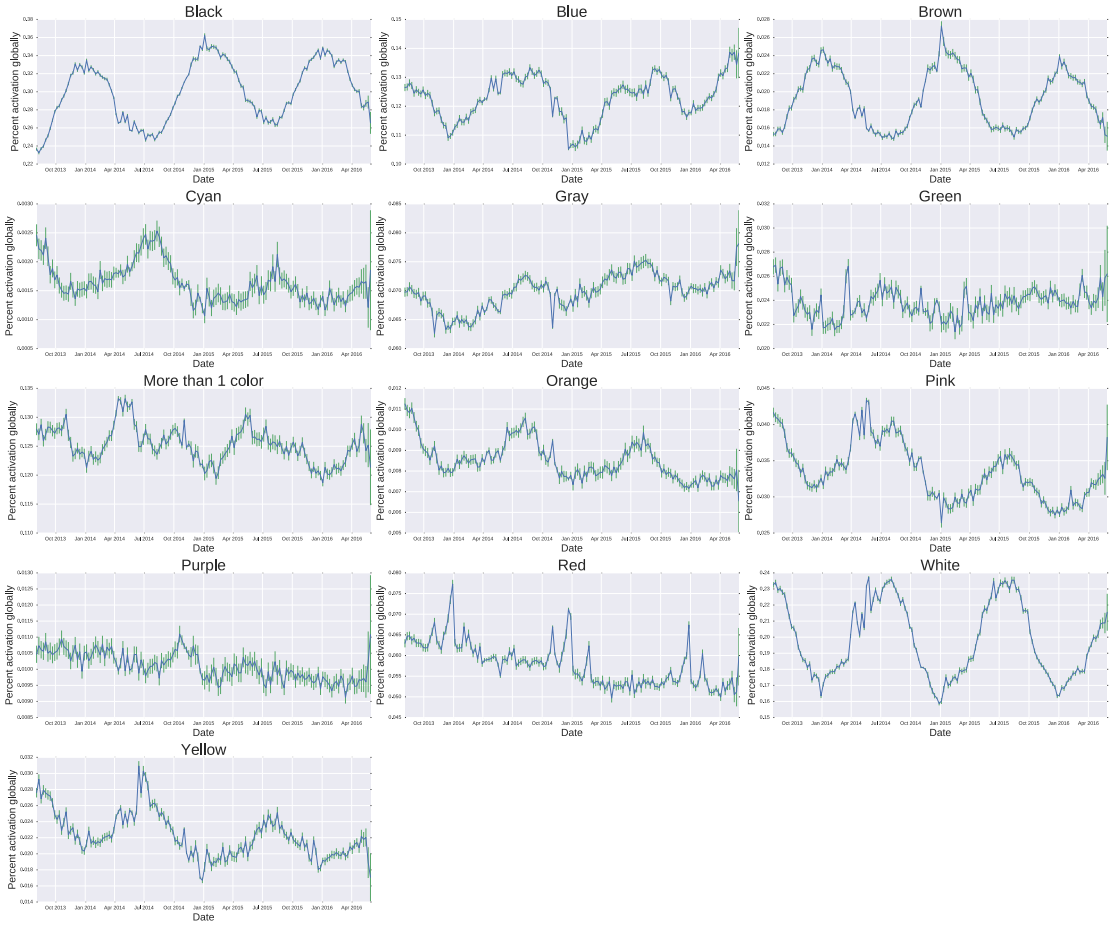


Figure 4.6: Mean major color score for several different colors. The means are computed by aggregating measurements per 1 week intervals and error bars indicate 95% confidence interval. Colors such as brown and black exhibit unified seasonal patterns whereas colors like blue and white are the opposite. Colors such as gray are trending up year over year whereas red, orange and purple are trending down.

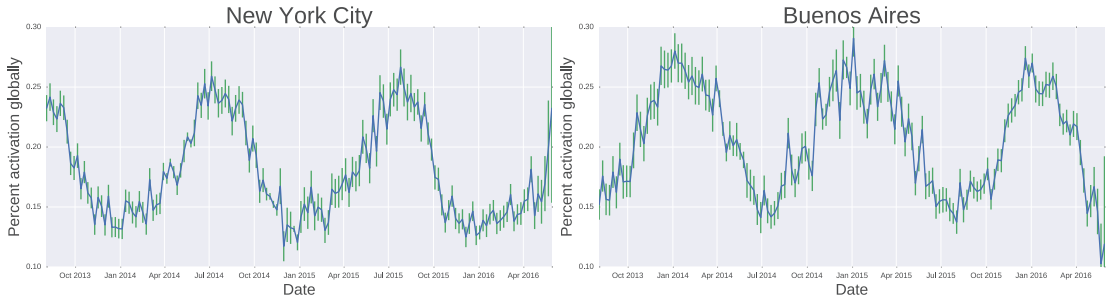


Figure 4.7: Mean major color score for the color white for two cities highlighting flipped behavior between Northern Hemisphere and Southern Hemisphere. Mean is computed by aggregating measurements per 1 week intervals and error bars indicate 95% confidence interval.

Hemisphere flip this pattern, suggesting a correlation between season and color. An example with two cities is shown in Figure 4.7. Figure 4.8 shows all pairs of cities correlated by their white color temporal profiles, with cities reordered by latitude to highlight this seasonal effect—cities in each hemisphere tend to be highly correlated with one another, while cities in opposite hemispheres are highly negatively correlated. In the United States, there is an oft-cited rule that one should not wear white after Labor Day, with claims that this rule extended from practical seasonal choices when weather is cooler. Using this method, I can actually extract meaningful data that supports the claim that there is a significant decrease in white clothing that begins mid-September, shortly after Labor Day, shown in Figure 4.6.

Other colors exhibit interesting, but more subtle trends. The color red is much less periodic, and appears to have been experiencing a downward trend, down approximately 1% since a few years ago. Whether this trend is a true change in how people around the world wear the color red or a result of chang-

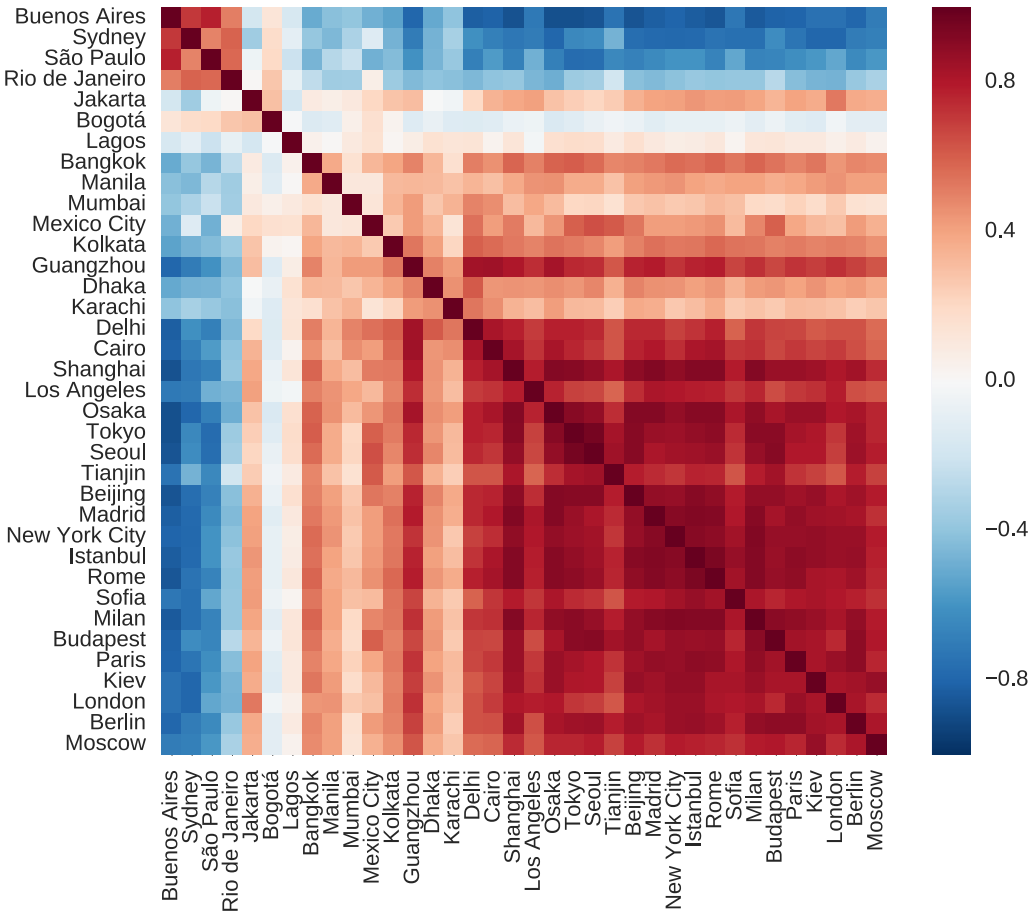


Figure 4.8: Pearson's correlation coefficients for the mean prediction score of the color white across all pairs of major world cities. Labels have been permuted by latitude to highlight the relationship between the northern and southern hemispheres and their relationship to the seasonal trend of the color white.

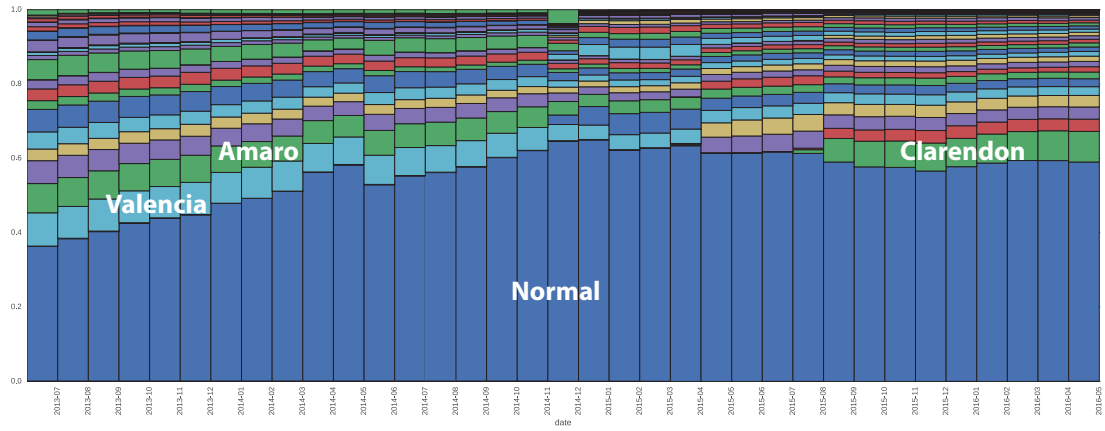


Figure 4.9: Distribution of Instagram filters used in the dataset. More than half of the dataset has no filter applied. Popular filters near the end of the dataset include the filter "Clarendon". Filters popular several years earlier such as "Valencia" have declined significantly over the years.

ing camera technology and image filtering choices is unknown. For example, Doersch, et al. [14] found anomalies in internet photography related to chromatic aberration and took additional steps to prevent their learned neural network from using this information in its regression tasks. I performed additional analysis in two ways. Instagram is a platform where users may manipulate their photos via filters. These filters modify the color composition of the photo and may add decorations such as borders. Figure 4.9 shows the popularity of certain filters across time for our dataset. Early in 2013, many more photos had filters applied, but that has decreased as more people now favor no filter or "normal". Furthermore, earlier, filters designed to make photos look vintage such as "Valencia" were popular whereas now filters that improve photos in a more traditional sense by brightening colors and sharpening edges such as "Clarendon" have increased in popularity. Filters like "Valencia" emphasize reds whereas filters like "Clarendon" emphasize blue-green. A second visualization shown

Figure 4.10: Average image for each month in the dataset with per-pixel mean subtracted and scaled by per-pixel standard deviation. Top row: 2013, Bottom row: 2016. Over time photos uploaded to Instagram have been progressively changing in mean color from reddish to greenish.

in Figure 4.10 is a set of average images, one for each month, across the entire dataset. This illustrates that photos are becoming progressively more green versus red over time. Producing a similar visualization in Figure 4.11 using only images with no Instagram filter applied reveals that much of this trend is due to filter usage, although there is still a slight trend from red to green.

Returning to the temporal analysis of the color red shown in Figure 4.6, beyond these long term trends, there are also two events that appear each year, one small event near the end of each October and a much larger event near the

Figure 4.11: Average image for each month in the "filter=Normal" subset with per-pixel mean subtracted and scaled by per-pixel standard deviation. Top row: 2013, Bottom row: 2016. Over time photos uploaded to Instagram have been progressively changing in mean color from reddish to greenish despite considering only those with no filter applied.

end of each December. I examined the visual clusters and found one that contained images with red clothing and had spikes at the same two times of the year. This cluster was not just red, but red hats. An assortment of winter knit beanies and red jacket hoods were found, but what stood out were a large assortment of Santa hats as well as an unexpected assortment of red Halloween costumes with red hats or hoods. It appears that red hats are uncommon enough that when they do appear they are highly correlated with the exceptional sort of costumes worn on these two holidays. Finally, the Chinese New Year sees a

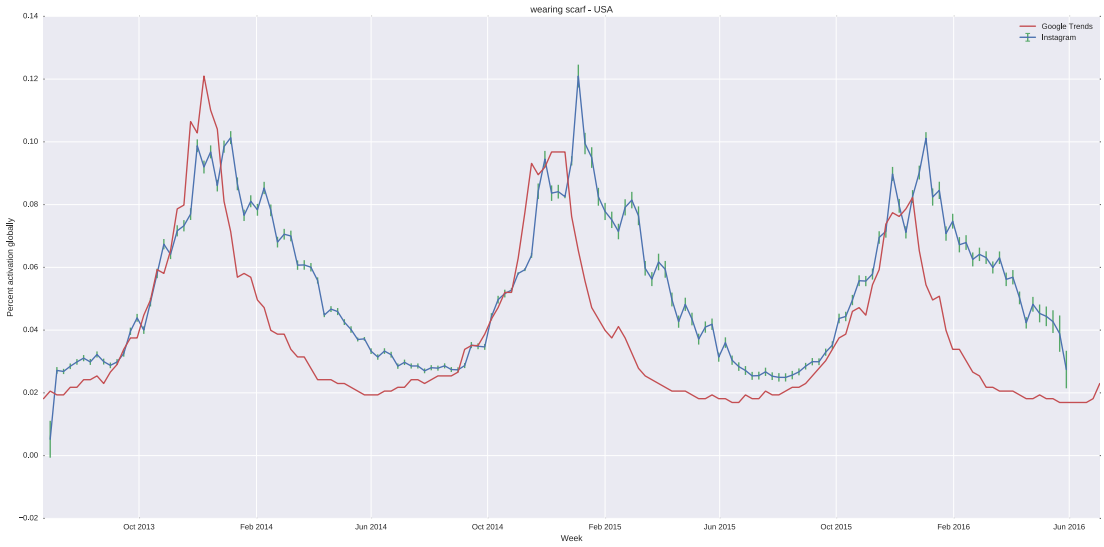


Figure 4.12: Scarf trends in the USA. The Instagram-based measurements are compared to Google Trends. Instagram measurements are derived from mean scarf score aggregated by week with error bars indicating 95% confidence interval. Google Trends signal is per-week "q=scarf, geo=US" query volume normalized by total query volume.

similar spike in the color red in 2015 and 2016. I did not observe a similar spike in 2014.

Clothing types. I can also explore more functional attributes. For example, we know people wear scarves in cold weather, so one would expect to see a seasonal trend. Figure 4.12 shows such a trend for scarves in the United States. However, I can also compare with a secondary source of information to estimate this signal, Google Trends.[4] Google Trends provides historical trend data for Google Search queries. By comparing these two signals, I can derive three insights. The first is that the shapes of the signals are similar. Both feature a sudden attack and decay rather than a purely sinusoidal shape. The second is that

[4] https://www.google.com/trends/explore#q=scarf&geo=US

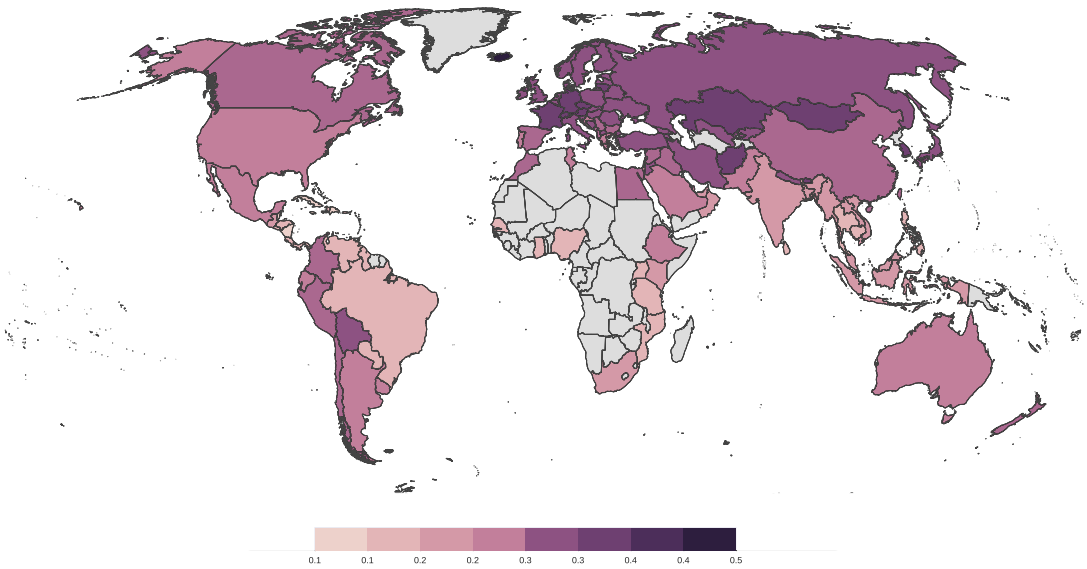


Figure 4.13: Mean jacket score per country. Countries with ≥ 1000 photos included. Countries with < 1000 photos shown in gray.

the onset of the signal derived from Google Trends tends to come prior to the onset from the Instagram photos. One plausible explanation is that people are searching for scarves on Google in order to purchase them and Instagram allows us to measure when they are actually wearing them. The third piece of insight is that while Google Trends suggests that scarves are in a significant downward annual trend, the Instagram results suggest that they are in a significant *upward* annual trend. There are several possible explanations for this effect such as people still having the scarves that they purchased last season, but the important insight is that these two signals tell two different stories. Both of these sources of data together give us a much better sense of what is happening in the world.

So far I have focused on temporal trends, but I can also explore interesting geographic trends. Can we validate the attribute classifications using prior knowledge of climate and weather-related clothing? For example, where in the

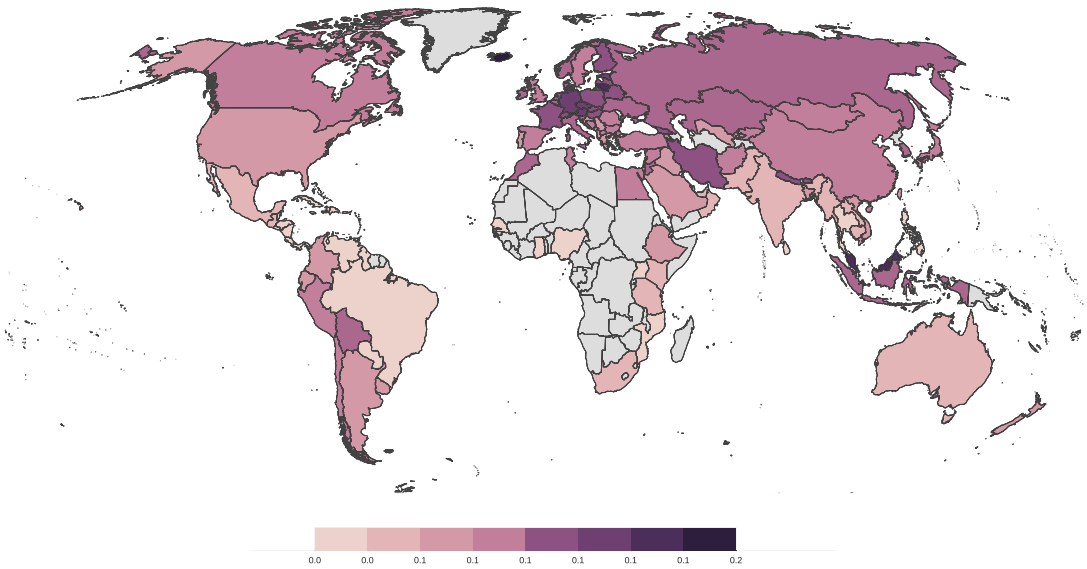


Figure 4.14: Mean scarf score per country. Countries with ≥ 1000 photos included. Countries with < 1000 photos shown in gray.

world do people wear jackets? Figure 4.13 shows the percentage of photos per country that contain people wearing jackets. This accords with intuition. Countries near the north tend to wear jackets more often in their photos. In South America, as you go further west, more jackets are to be found. A similar pattern appears for scarves in Figure 4.14. One explanation is that climate is also related to elevation and winter clothes will be worn more often at higher elevation (in this case the Andes) than at lower elevations.

But what about a more unexpected geographic trend? Where in the world are people wearing hats? Figure 4.15 shows the percentages of photos per country that contain people wearing hats. Once again, people wear hats in colder places, but what is interesting is that in Oman, hats are apparently extremely popular. Several examples of images that were highly ranked as wearing hats in Oman are shown in Figure 4.16. In particular, the kuma and massar are pop-

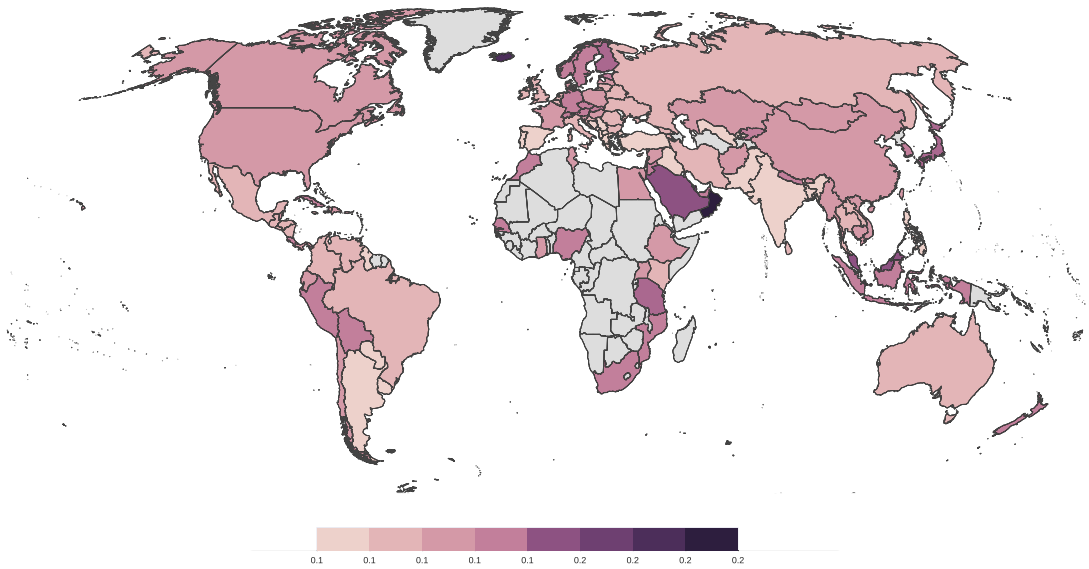


Figure 4.15: Mean hat score per country. Countries with ≥ 1000 photos included. Countries with < 1000 photos shown in gray.

Figure 4.16: Several top-ranked examples of hats in Oman. While not strictly hats, it's clear that the classifier has learned to identify more generally, head coverings.

ular in Oman since they are an important element of the men's national dress.

4.4.2 Visualizing Styles

The annotations for this dataset were motivated by prior knowledge of what might be interesting to measure. However, it's unclear a priori whether or not

Figure 4.17: Images closest to the centers of several mixture components in our GMM. Clusters are presented in no particular order.

certain combinations of these attributes might be even more interesting. A very straightforward method for visualizing consistent pockets of this visual clothing embedding is to use an off-the-shelf clustering method such as the Gaussian Mixture Model (GMM). I use approximately 5.4M photos to build the GMM and then for each novel photo, I record the centroid to which it is closest. In turn, I can analyze the proportion of each cluster across space and across time to mine for styles that are well localized in one or both. Figure 4.17 shows several such clusters, in no particular order.

This raw visualization of clusters is not particularly easy to explore as there are hundreds of these to study and understand. Therefore, it's useful to define a ranking function to help surface the most interesting signals. One option is to rank the clusters according to entropy over the different cities in the dataset. Figure 4.18 shows the top clusters for such a ranking. This ranking helps provide a sense of how geography affects clothing as cities in Europe tend to have a higher percentage of winter clothing than other countries.

Finally, we can perform a similar ranking, but consider both cities as well as months in the distributions we rank by entropy. Figures 4.19, 4.20, 4.21, 4.22, 4.23, 4.24 illustrate some of the top ranked clusters. (Clusters that represent similar concepts such as winter clothing are suppressed for this visualization, although each tends to encode a slightly different variation.) Some articles of clothing such as the hijab are popular in locations with a specific cultural connection such as Cairo and Jakarta in Figure 4.19. Figures 4.20 and 4.21 show an example where climate is connected to clothing, where the latter breaks the strict seasonal dependency by appearing throughout the entire year in Singapore. Figure 4.22 shows that clusters can capture styles that happen to be ex-

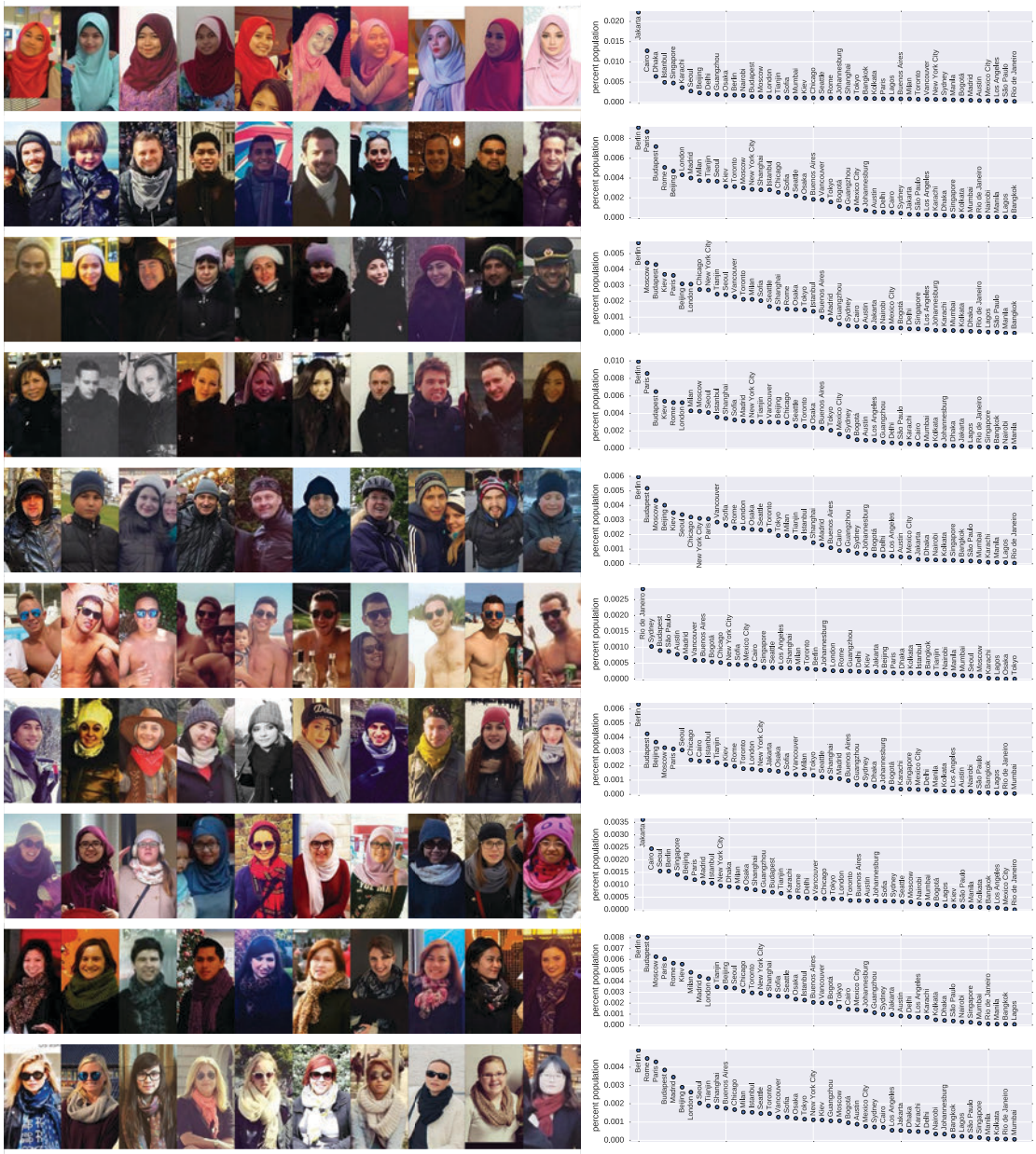


Figure 4.18: Top ranked GMM clusters according to entropy across cities.

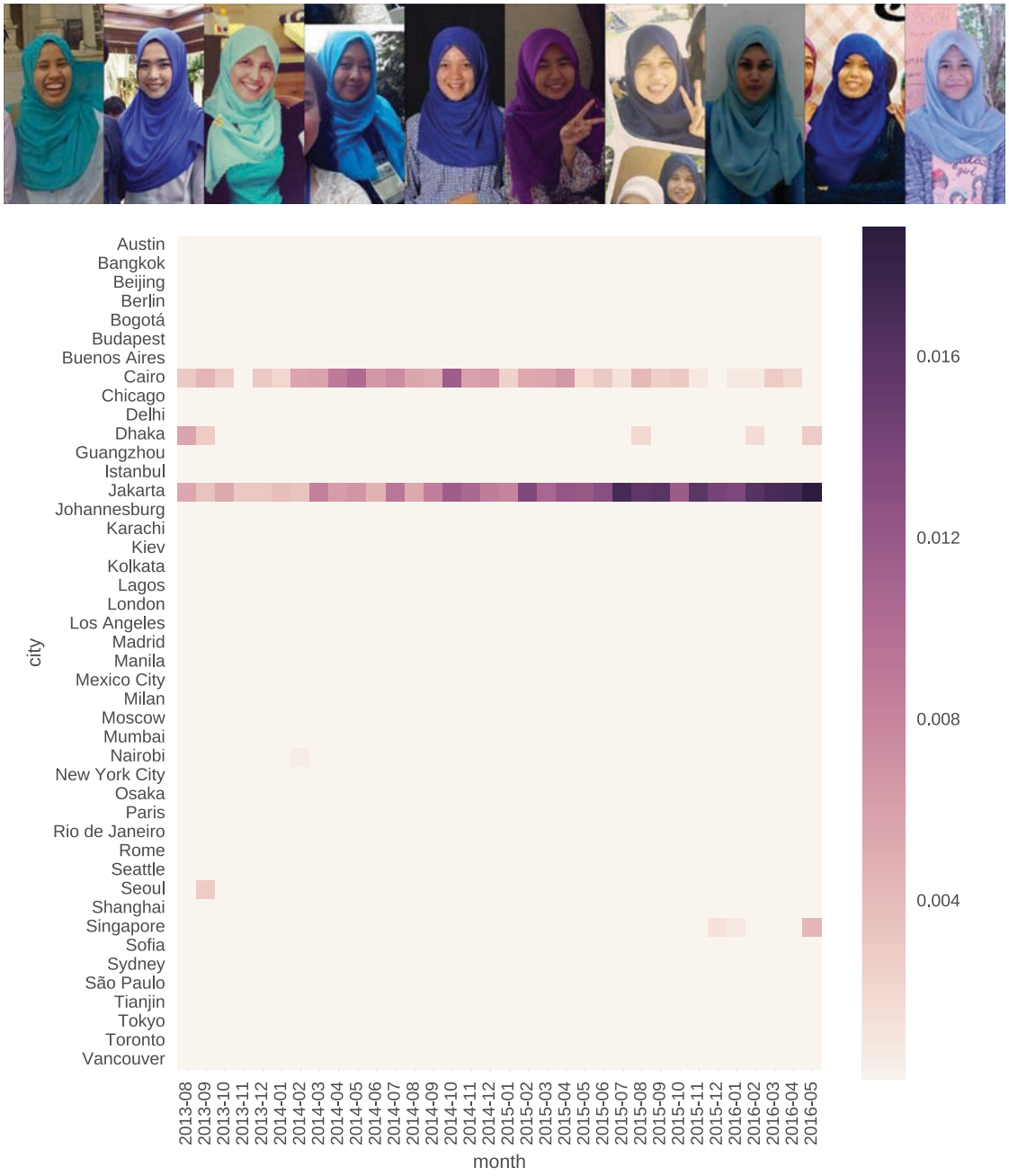


Figure 4.19: High-ranking spatio-temporal cluster illustrating that hijabs are a popular garment in Cairo and Jakarta.

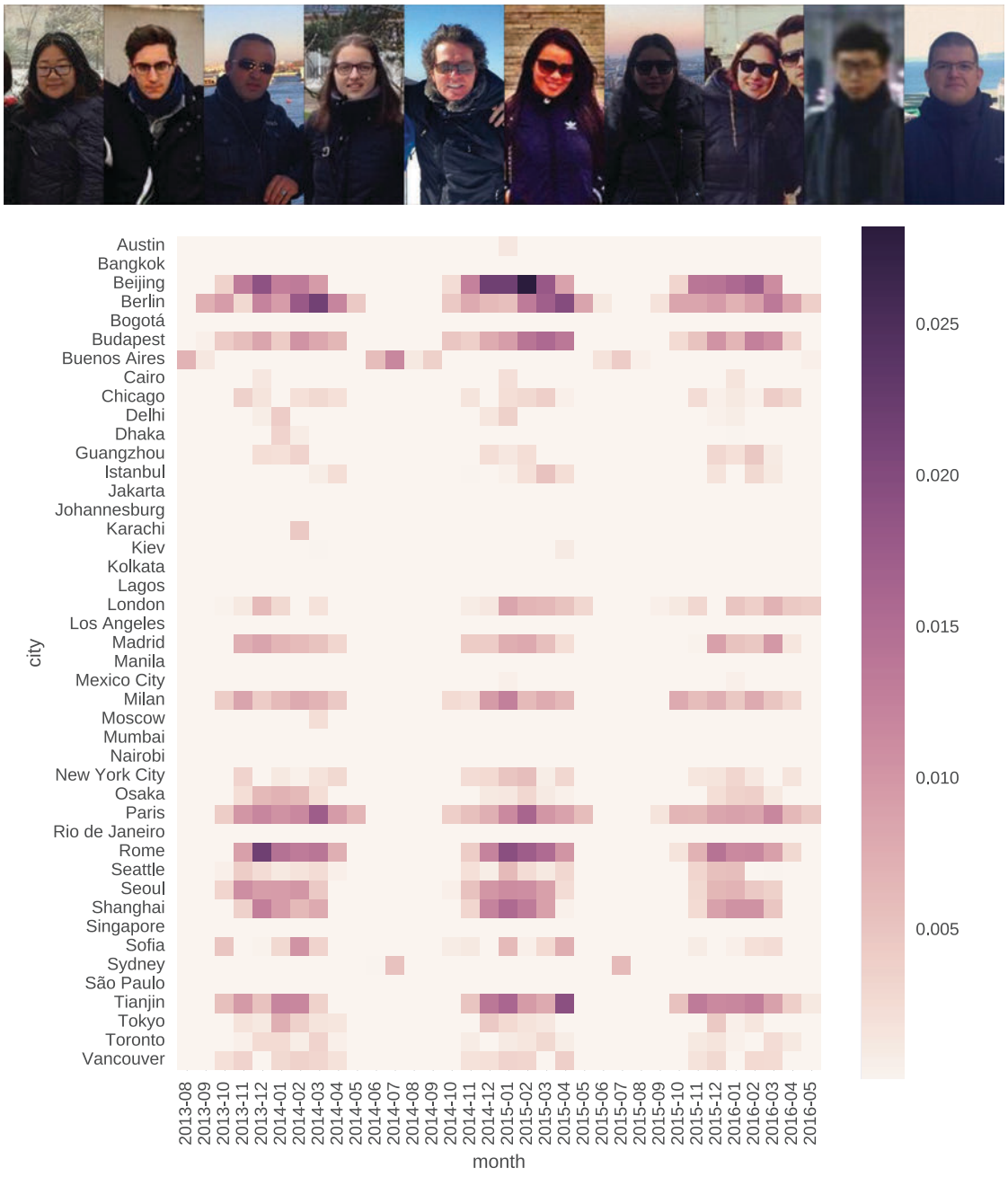


Figure 4.20: High-ranking spatio-temporal cluster illustrating that dark jackets and glasses are a popular combination during the winter in several cities that typically experience cold winters.

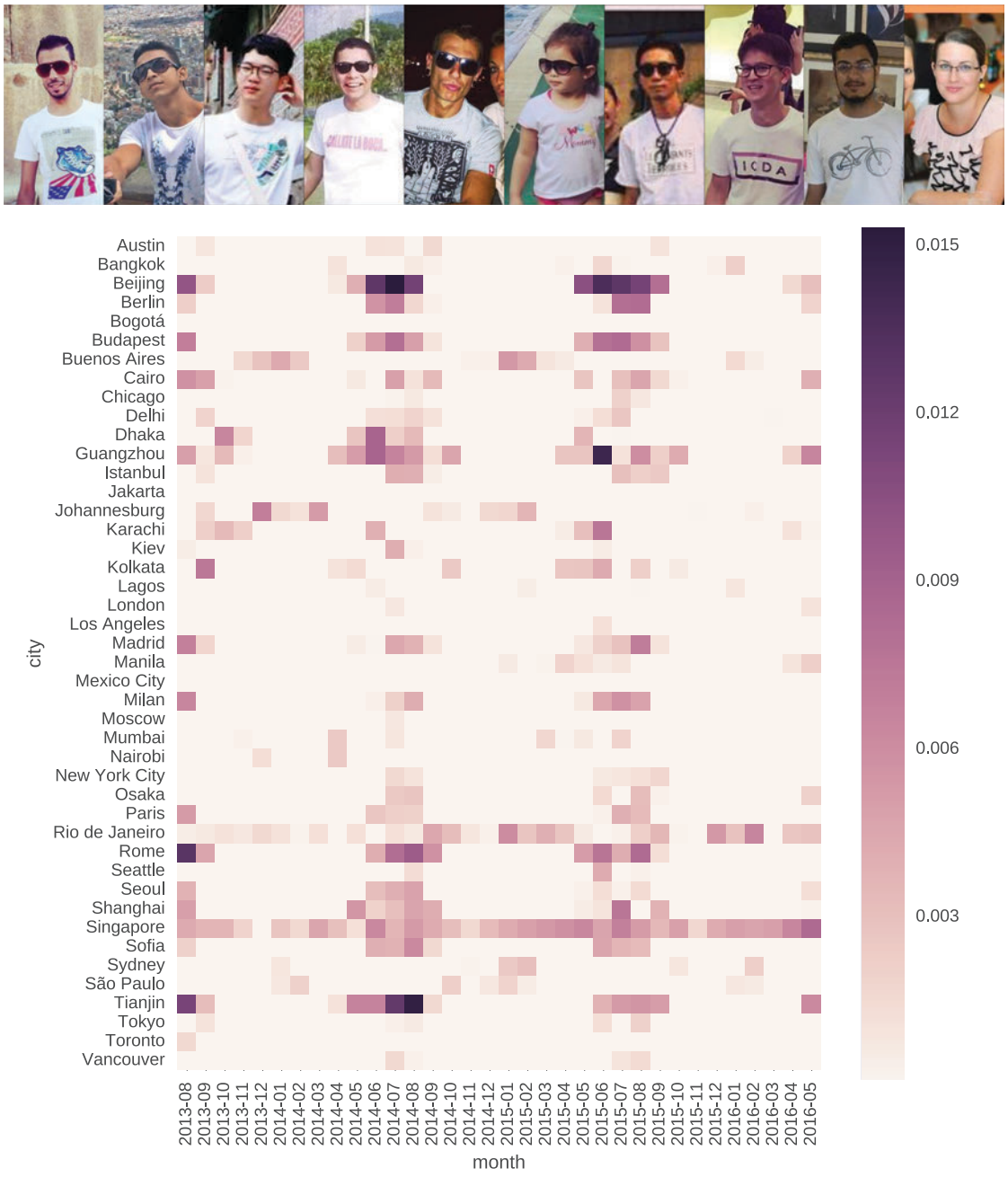


Figure 4.21: High-ranking spatio-temporal cluster illustrating that white t-shirts with graphic designs and glasses are a popular combination during summer months, but popular in Singapore throughout the entire year.

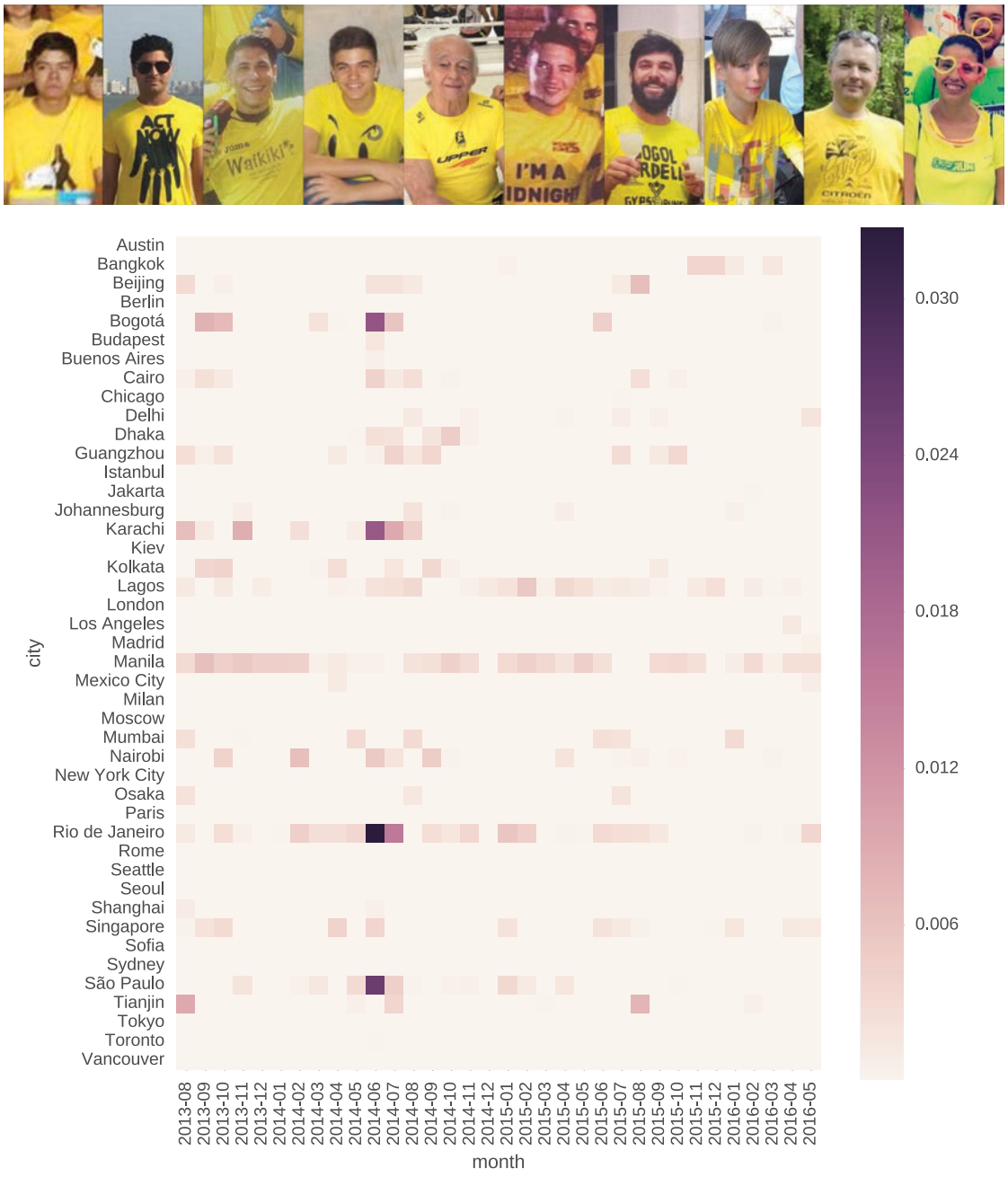


Figure 4.22: High-ranking spatio-temporal cluster illustrating that yellow t-shirts with graphic patterns were very popular for a very short period of time, June-July 2014, in specifically Bogotá, Karachi, Rio de Janeiro, and São Paulo.

Figure 4.23: High-ranking spatio-temporal cluster illustrating that blue polo shirts are popular in Cairo and throughout the Indian subcontinent.

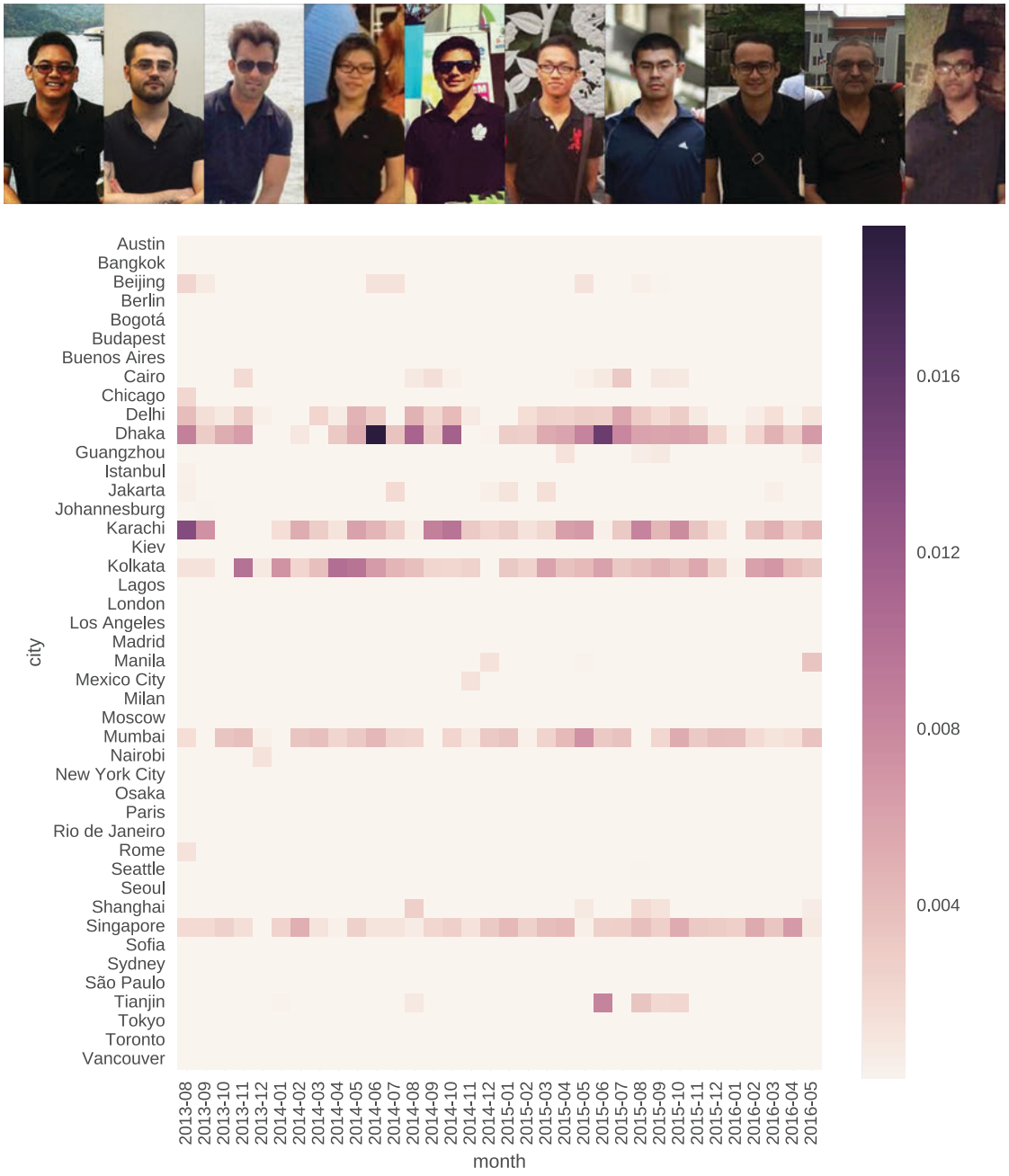


Figure 4.24: High-ranking spatio-temporal cluster illustrating that black polo shirts with glasses are popular in Singapore and throughout the Indian subcontinent.

tremely popular at a very specific time and place. In this case, June and July 2014 capture the 2014 World Cup and the contents of these clusters contain many examples of soccer fans wearing team colors. Finally, Figures 4.23 and 4.24 show that very similar styles can be popular throughout a very large region such as the Indian subcontinent, yet for one reason or another, one is also popular in Cairo, but not Singapore and the other vice versa.

4.4.3 Artist Reference Material Application

Artists and illustrators often have to design outfits for crowds in movies and videos such that they represent the style of some real-world locale. I claim that automated analysis of internet photos can be used to enable such a large scale understanding of how they should design their crowds by more clearly organizing reference photos than if they browsed a random stream.

But are these automatically organized collections of photos better than just a large list of random internet photos? I conducted a case study with two fashion illustrators with prior undergraduate training in fashion design. They did not have much prior knowledge of certain regions of the world.

I designed the following 2-phase experiment. First I selected Mumbai as a city for them to illustrate. Both illustrators had not visited India and were not trained in Indian fashion. In both phases, each illustrator was given a similar number of photos (about 100) from Instagram to be used as reference material and was instructed to draw 15 figure sketches they believed captured the core elements of style that they found in these photos. In the first phase, these photos were 100 random photos collected from Instagram as geotagged in Mumbai

(*the before condition*). In the second phase they were given style clusters from Mumbai which were organized into 16 top-ranked clusters, each with 6 photos (*the after condition*). The illustrators were given three hours to study the random baseline reference material and complete the set of 15 illustrations. One day later they were given three hours to study the clustered reference material and complete the second set of illustrations. Even though the analysis pipeline was only done on torsos, the illustrators were shown in both the before and after conditions the complete photo so that they could illustrate the full body costume.

Their illustrations for Mumbai are shown in Figure 4.25: Row 1 (Artist 1, before), Row 2 (Artist 1, after), Row 3 (Artist 2, before), Row 4 (Artist 2, after). First I note that both artists added a sari, a typical clothing item in the Indian subcontinent, to their sketch after seeing our style clusters. They also added plaid shirts for men. A similar before and after for Jakarta made the illustrators draw more hijabs in the after condition shown in Figure 4.26.

To quantitatively validate our results, I surveyed a set of 15 participants who had lived (for at least a month) in Mumbai in the last 5 years. Each was shown a pair of images from the same artist, randomly flipping between the before and after conditions being presented first. Participants were given a brief web survey asking which image they believed more accurately captured the style of clothing in Mumbai. Out of 15 participants, 9 chose Row 2, and 12 chose Row 4, the *after* sketches based on the style clusters. While this result validates the use of the representative style as a potentially useful tool for artists, more exploration is needed to understand the nuances of the subject choices, how crowds are interpreted etc. In the future I plan to do further exploration and

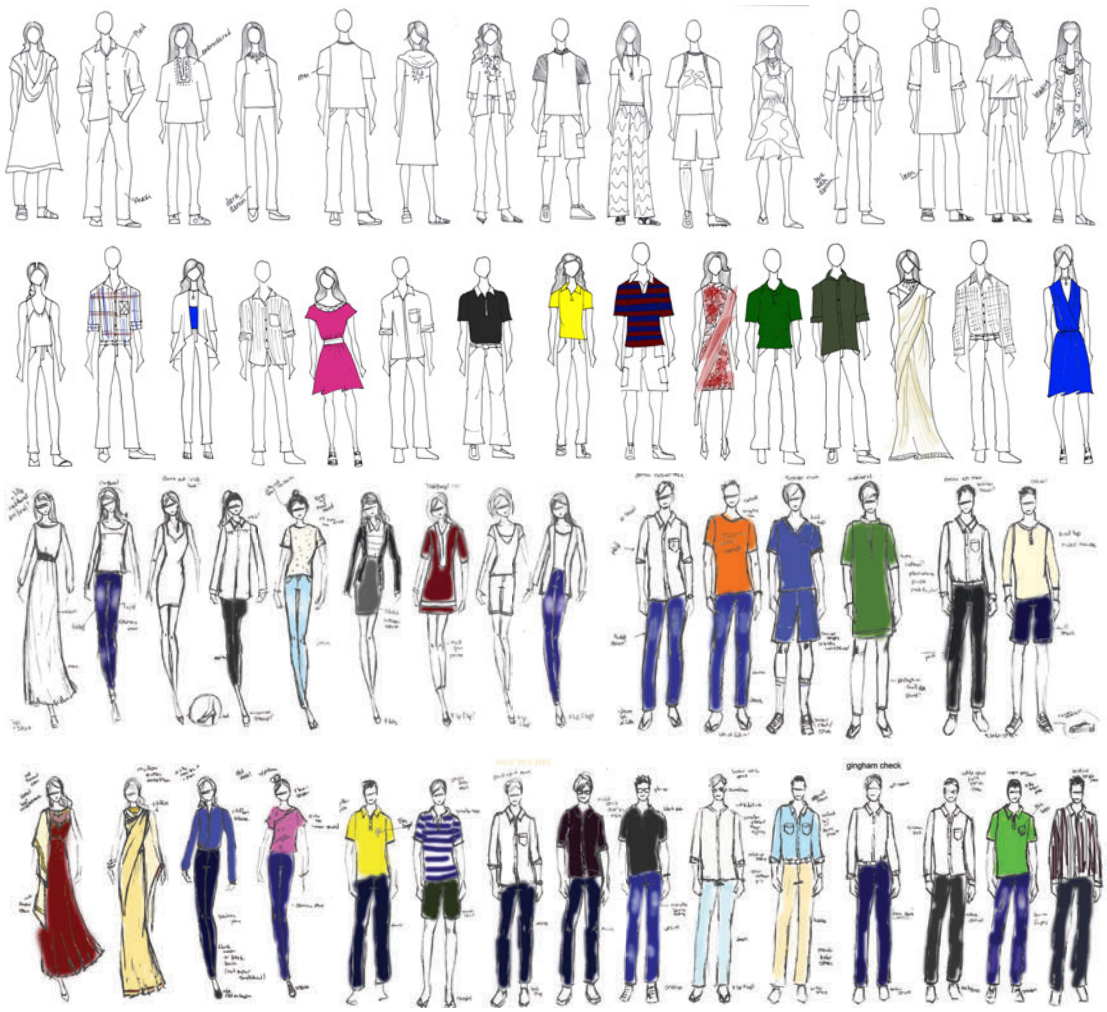


Figure 4.25: Fashion sketches of Mumbai used in study. Row 1: Mumbai, Artist 1, random reference material, Row 2: Mumbai, Artist 1, using style clusters, Row 3: Mumbai, Artist 2, random reference material, Row 4: Mumbai, Artist 2, using style clusters.

Figure 4.26: Fashion sketches of Jakarta used in study. Row 1 left: Jakarta, Artist 1, random reference material, Row 1 right: Jakarta, Artist 1, using style clusters, Row 2 left: Jakarta, Artist 2, random reference material, Row 2 right: Jakarta, Artist 2, using style clusters.

follow up questionnaires for more cities.

4.5 Limitations

CNN-based embeddings are a very powerful approach for organizing imagery in a high-dimensional feature space when given a set. Furthermore, unsupervised methods such as GMMs can be used to further explore this space. How-

ever, there is a limit to the granularity of styles that are obtained. One useful extension would be to incorporate active learning where users are asked to rank similarity of styles. Another useful extension would be to apply weakly-supervised learning using the spatio-temporal metadata directly in the embedding objective function rather than use it for post hoc analysis. Finally, this technique is limited to analyzing the upper body of the person. As computer vision techniques for pose estimation mature, it would be interesting to revisit this design to normalize articulated pose for full body analysis.

4.6 Future Work

There are many areas for future work. I would like to more thoroughly study dataset bias. This current dataset measures a particular population, in a particular context (Instagram users and the photos they take). I plan to extend to additional image sources and compare trends across them, but also to study aspects such as: can we distinguish between the (posed) subject of a photo and people who may be incidentally in the background? Does that make a difference in the measurements? Can we identify and take into account differences between tourists and locales? Are there certain types of people that tend to be missing from such data? Do our face detection algorithms themselves exhibit bias? I would also like to take other difficult areas of computer vision, such as pose estimation, and characterize them in a way that makes them amenable to measurement at scale.

Ultimately, I would like to apply automatic data exploration algorithms to derive, explain, and analyze the significance of insights from machine learn-

ing (e.g., see the "Automatic Statistician" [42]). It would be also be interesting to combine visual data with other types of information, such as temperature, weather, and textual information on social media to explore new types of as-yet unseen connections. The synthesis of big data, machine learning, and automated analysis algorithms, would make for a very powerful tool in computer graphics and vision, and more broadly in visual discovery of fashion and many other areas.

CHAPTER 5

CONCLUSIONS

Visual vocabularies form the entities from which large-scale image collections can be analyzed. This dissertation presented three different approaches to crafting visual vocabularies for describing imagery in three different domains. The first representation extracted spatio-temporal segments for a time-varying 3D reconstruction application. These segments enabled a rendering application to visualize the world from a user-specified viewpoint and time. They were also validated as being useful for timestamping photography. The second representation consisted of discriminative patches mined from large scale image corpora using a combination of state-of-the-art image classification methods and an experimental protocol derived from the human visual perception literature. The final representation allowed a domain specialist to specify a subset of a visual vocabulary, supervised machine learning was used to learn a high-dimensional embedding, and unsupervised machine learning was used to further analyze this embedding. A large-scale case study on millions of examples of people in internet photos was used to study clothing fashion trends across the world.

This body of work primarily builds what I called visual vocabularies, but in reality, what was analyzed included people, places, and things; more like *visual nouns*. However, I presented no attempt at understanding the *visual verbs*; essentially the field of action recognition. Action recognition is certainly an important field in the computer vision community, but the input is often video rather than static photos. I believe that not only will this prompt further developments in computer vision for video, but that social media platforms will become increasingly more video-centric versus photo-centric as more users take

advantage of even higher bandwidth mobile internet and more compact smartphone cameras. It's intriguing to imagine that as action recognition improves in accuracy, we will begin to see interesting applications of fine-grained action recognition. Similar to how we can analyze photos of clothing to find trends and patterns across the globe, perhaps we can also see and *measure* variations in human posture and gait around the world.

Finally, much of this work is motivated by building descriptions that can be conveyed to analysts. For example, in Chapter 3, a set of patches was extracted, clustered, and so on, with the intent that it be shown to a human to convey meaning. This is where the analogy to a vocabulary is built; these are the units that have to speak to the analyst. However, the analogy could be taken to the point where we use the plethora of machine learning techniques built for the natural language processing community to do further reasoning. Imagine replacing word2vec [46] with a "visualvocab2vec" and leveraging existing topic modeling strategies. Perhaps a knowledge base of relations that connect the visual vocabulary entities could be built automatically from the web to improve reasoning. (i.e., A visual Knowledge Vault [16])

I believe that ultimately, it will be a combination of (1) continual improvements in core computer vision algorithms such as detection, recognition, segmentation, and reconstruction, (2) large visual web data, (3) visual vocabularies, and (4) unified frameworks to learn relations between these entities that will enable the large-scale, automated visual understanding of people, places, and things from the web.

BIBLIOGRAPHY

[1] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. Building Rome in a day. In *ICCV*, 2009.

[2] S.M. Arietta, A.A. Efros, R. Ramamoorthi, and M. Agrawala. City forensics: Using visual elements to predict non-visual city attributes. *IEEE Trans. Visualization and Computer Graphics*, 20(12):2624–2633, Dec 2014.

[3] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. *CoRR*, abs/1412.7755, 2014.

[4] Luca Ballan, Gabriel J. Brostow, Jens Puwein, and Marc Pollefeys. Unstructured video-based rendering: Interactive exploration of casually captured videos. In *SIGGRAPH*, 2010.

[5] Lukas Bossard, Matthias Dantone, Christian Leistner, Christian Wengert, Till Quack, and Luc Van Gool. Apparel classification with style. In *Proc. Asian Conf. on Computer Vision*, 2013.

[6] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *ECCV*, 2014.

[7] Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Describing people: Poselet-based attribute classification. In *ICCV*, 2011.

[8] Christoph Bregler, Aaron Hertzmann, and Henning Biermann. Recovering non-rigid 3d shape from image streams. In *CVPR*, pages 690–696, 2000.

[9] Huizhong Chen, Andrew Gallagher, and Bernd Girod. Describing clothing by semantic attributes. In *ECCV*, 2012.

[10] Tali Dekel Basha, Yael Moses, and Shai Avidan. Photo sequencing. In *ECCV*, 2012.

[11] Jia Deng, J. Krause, and Li Fei-Fei. Fine-grained crowdsourcing for fine-grained recognition. In *CVPR*, pages 580–587, June 2013.

[12] Wei Di, C. Wah, A. Bhardwaj, R. Piramuthu, and N. Sundaresan. Style finder: Fine-grained clothing style detection and retrieval. In *Proc. CVPR Workshops*, 2013.

[13] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Mid-level visual element discovery as discriminative mode seeking. In *NIPS*, 2013.

[14] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015.

[15] Carl Doersch, Saurabh Singh, Abhinav Gupta, Josef Sivic, and Alexei A. Efros. What makes Paris look like Paris? *SIGGRAPH*, 31(4):101:1–101:9, 2012.

[16] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *KDD*, 2014.

[17] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results. http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html.

[18] Mani Golparvar Fard, Feniosky Peña-Mora, and Silvio Savarese. Monitoring changes of 3d building elements from unordered photo collections. In *ICCV Workshops*, 2011.

[19] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 32, 2010.

[20] Jan-Michael Frahm, Pierre Georgel, David Gallup, Tim Johnson, Rahul Raguram, Changchang Wu, Yi-Hung Jen, Enrique Dunn, Brian Clipp, Svetlana Lazebnik, and Marc Pollefeys. Building Rome on a cloudless day. In *ECCV*, 2010.

[21] Yasutaka Furukawa, Brian Curless, Steven M. Seitz, and Richard Szeliski. Reconstructing building interiors from images. In *ICCV*, 2009.

[22] Yasutaka Furukawa, Brian Curless, Steven M. Seitz, and Richard Szeliski. Towards Internet-scale multi-view stereo. In *CVPR*, 2010.

[23] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *PAMI*, 32(8):1362–1376, 2009.

[24] Shiry Ginosar, Kate Rakelly, Sarah Sachs, Brian Yin, and Alexei A. Efros. A Century of Portraits: A visual historical record of american high school yearbooks. In *ICCV Workshops*, December 2015.

[25] R. B. Girshick, P. F. Felzenszwalb, and D. McAllester. Discriminatively trained deformable part models, release 5. http://people.cs.uchicago.edu/~rbg/latent-release5/, 2012.

[26] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[27] Scott A. Golder and Michael W. Macy. Diurnal and seasonal mood vary with work, sleep, and daylength across diverse cultures. *Science*, 333(6051):1878–1881, 2011.

[28] Frédéric Gosselin and Philippe G. Schyns. Bubbles: a technique to reveal the use of information in recognition tasks. *Vision Research*, 41(17):2261 – 2271, 2001.

[29] Shintami C. Hidayati, Kai-Lung Hua, Wen-Huang Cheng, and Shih-Wei Sun. What are the fashion trends in new york? In *Proc. Int. Conf. on Multimedia*, 2014.

[30] Mohammad T Islam, Connor Greenwell, Richard Souvenir, and Nathan Jacobs. Large-scale geo-facial image analysis. *EURASIP J. on Image and Video Processing*, 2015(1), 2015.

[31] Nathan Jacobs, Nathaniel Roman, and Robert Pless. Consistent temporal variations in many outdoor scenes. In *CVPR*, 2007.

[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. *CoRR*, 2014.

[33] M. Juneja, A. Vedaldi, C. V. Jawahar, and A. Zisserman. Blocks that shout: Distinctive parts for scene classification. In *CVPR*, 2013.

[34] M. Hadi Kiapour, Xufeng Han, Svetlana Lazebnik, Alexander C. Berg, and Tamara L. Berg. Where to Buy It: Matching street clothing photos in online shops. In *ICCV*, 2015.

[35] M. Hadi Kiapour, Kota Yamaguchi, Alexander C. Berg, and Tamara L. Berg. Hipster wars: Discovering elements of fashion styles. In *ECCV*, pages 472–488, Cham, 2014. Springer International Publishing.

[36] Bryan Klingner, David Martin, and James Roseborough. Street view motion-from-structure-from-motion. In *ICCV*, 2013.

[37] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

[38] Quannan Li, Jiajun Wu, and Zhuowen Tu. Harvesting mid-level visual concepts from large-scale internet images. In *CVPR*, 2013.

[39] Yao Li, Lingqiao Liu, Chunhua Shen, and Anton van den Hengel. Mid-level deep pattern mining. In *CVPR*, 2015.

[40] Yunpeng Li, Noah Snavely, Daniel Huttenlocher, and Pascal Fua. Worldwide pose estimation using 3D point clouds. In *ECCV*, 2012.

[41] Si Liu, Zheng Song, Meng Wang, Changsheng Xu, Hanqing Lu, and Shuicheng Yan. Street-to-shop: Cross-scenario clothing retrieval via parts alignment and auxiliary set. In *Proc. Int. Conf. on Multimedia*, 2012.

[42] James Robert Lloyd, David K. Duvenaud, Roger B. Grosse, Joshua B. Tenenbaum, and Zoubin Ghahramani. Automatic construction and natural-language description of nonparametric regression models. In *AAAI*, 2014.

[43] Kevin Matzen and Noah Snavely. Scene chronology. In *ECCV*, 2014.

[44] Kevin Matzen and Noah Snavely. Bubblenet: Foveated imaging for visual discovery. In *ICCV*, 2015.

[45] Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, The Google Books Team, Joseph P. Pickett, Dale Holberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. Quantitative analysis of culture using millions of digitized books. *Science*, 2010.

[46] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their composition-

ality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *NIPS*, pages 3111–3119. Curran Associates, Inc., 2013.

[47] Volodymyr Mnih, Nicolas Heess, Alex Graves, and koray kavukcuoglu. Recurrent models of visual attention. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *NIPS*, pages 2204–2212. Curran Associates, Inc., 2014.

[48] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *Int. Conf. on Computer Vision Theory and Application*, 2009.

[49] Calvin Murdock, Nathan Jacobs, and Robert Pless. Building dynamic cloud maps from the ground up. In *ICCV*, 2015.

[50] Nikhil Naik, Jade Philipoom, Ramesh Raskar, and César Hidalgo. Streetscore: Predicting the perceived safety of one million streetscapes. In *Proc. CVPR Workshops*, 2014.

[51] Kai Ni, Hailin Jin, and Frank Dellaert. GroupSAC: Efficient consensus in the presence of groupings. In *ICCV*, 2009.

[52] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *ICML*, 2005.

[53] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*. MIT Press, 1999.

[54] T. Pollard and J.L. Mundy. Change detection in a 3-d world. In *CVPR*, pages 1–6, June 2007.

[55] A. Quattoni and A. Torralba. Recognizing indoor scenes. In *CVPR*, pages 413–420, June 2009.

[56] Michael Rubinstein, Ce Liu, Peter Sand, Frédo Durand, and William T. Freeman. Motion denoising with application to time-lapse photography. In *CVPR*, 2011.

[57] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bern-

stein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3), 2015.

[58] 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.

[59] Tawfiq Salem, Scott Workman, Menghua Zhai, and Nathan Jacobs. Analyzing human appearance as a cue for dating images. In *Proc. Winter Conf. on Applications of Comp. Vision*, 2016.

[60] G. Schindler and F. Dellaert. Probabilistic temporal inference on reconstructed 3d scenes. In *CVPR*, pages 1410–1417, June 2010.

[61] Grant Schindler, Frank Dellaert, and Sing Bing Kang. Inferring temporal order of images from 3D structure. In *CVPR*, 2007.

[62] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, 2011.

[63] Edgar Simo-Serra, Sanja Fidler, Francesc Moreno-Noguer, and Raquel Urtasun. Neuroaesthetics in fashion: Modeling the perception of fashionability. In *CVPR*, June 2015.

[64] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[65] Saurabh Singh, Abhinav Gupta, and Alexei A. Efros. Unsupervised discovery of mid-level discriminative patches. In *ECCV*, 2012.

[66] Sudipta Sinha, Drew Steedley, and Richard Szeliski. Piecewise planar stereo for image-based rendering. In *ICCV*, 2009.

[67] Noah" "Snavely. snavely:code:2010: Structure from motion for unordered image collections. `http://phototour.cs.washington.edu/snavely:code:2010/`.

[68] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Modeling the world from Internet photo collections. *IJCV*, 80(2):189–210, November 2008.

[69] Jian Sun and J. Ponce. Learning discriminative part detectors for image classification and cosegmentation. In *ICCV*, pages 3400–3407, Dec 2013.

[70] Kalyan Sunkavalli, Wojciech Matusik, Hanspeter Pfister, and Szymon Rusinkiewicz. Factored Time-Lapse Video. In *SIGGRAPH*, 2007.

[71] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[72] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[73] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

[74] A. Taneja, L. Ballan, and M. Pollefeys. Image based detection of geometric changes in urban environments. In *ICCV*, Nov 2011.

[75] Aparna Taneja, Luca Ballan, and Marc Pollefeys. City-scale change detection in cadastral 3d models using images. In *CVPR*, June 2013.

[76] Yichuan Tang, Nitish Srivastava, and Ruslan R Salakhutdinov. Learning generative models with visual attention. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *NIPS*, pages 1808–1816. Curran Associates, Inc., 2014.

[77] Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. The new data and new challenges in multimedia research. *CoRR*, 2015.

[78] Ali Osman Ulusoy, Octavian Biris, and Joseph L. Mundy. Dynamic probabilistic volumetric models. In *ICCV*, 2013.

[79] Sundar Vedula, Simon Baker, Peter Rander, Robert T. Collins, and Takeo Kanade. Three-dimensional scene flow. *PAMI*, 27(3), 2005.

[80] S. Vittayakorn, K. Yamaguchi, A.C. Berg, and T.L. Berg. Runway to Realway: Visual analysis of fashion. In *Proc. Winter Conf. on Applications of Comp. Vision*, pages 951–958, Jan 2015.

[81] 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.

[82] Jingya Wang, Mohammed Korayem, and David J. Crandall. Observing the natural world with flickr. In *ICCV Workshops*, 2013.

[83] Xinggang Wang, Baoyuan Wang, Xiang Bai, Wenyu Liu, and Zhuowen Tu. Max-margin multiple-instance dictionary learning. In *ICML*. ICML, June 2013.

[84] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044, 2015.

[85] K. Yamaguchi, M.H. Kiapour, and T.L. Berg. Paper doll parsing: Retrieving similar styles to parse clothing items. In *ICCV*, Dec 2013.

[86] K. Yamaguchi, M.H. Kiapour, L.E. Ortiz, and T.L. Berg. Parsing clothing in fashion photographs. In *CVPR*, pages 3570–3577, June 2012.

[87] Kota Yamaguchi, Takayuki Okatani, Kyoko Sudo, Kazuhiko Murasaki, and Yukinobu Taniguchi. Mix and match: Joint model for clothing and attribute recognition. In *Proc. British Machine Vision Conference (BMVC)*, 2015.

[88] Wei Yang, Ping Luo, and Liang Lin. Clothing co-parsing by joint image segmentation and labeling. In *CVPR*, pages 3182–3189, June 2014.

[89] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014.

[90] Matthew Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[91] Haipeng Zhang, Mohammed Korayem, David J. Crandall, and Gretchen LeBuhn. Mining photo-sharing websites to study ecological phenomena. In *WWW*, 2012.

[92] Ning Zhang, Jeff Donahue, Ross Girshick, and Trevor Darrell. Part-based r-cnns for fine-grained category detection. In *ECCV*, pages 834–849, Cham, 2014. Springer International Publishing.

[93] Ning Zhang, Manohar Paluri, Marc'Aurelio Rantazo, Trevor Darrell, and Lubomir Bourdev. Panda: Pose aligned networks for deep attribute modeling. In *CVPR*, 2014.

[94] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning Deep Features for Scene Recognition using Places Database. *NIPS*, 2014.

[95] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. *CoRR*, abs/1412.6856, 2014.